ISSN 1346-7328
国総研報告第 42 号
平成 23 年 3 月 14 日

国土技術政策総合研究所報告

RESEARCH REPORT of
National Institute for Land and Infrastructure Management

No.42 March 14, 2011

国土交通省版・景観シミュレーション・システム Ver.2.09 のアーキテクチャ

高度情報化研究センター住宅情報システム研究官
小林　英之

The Architecture of Landscape Simulation System Ver.2.09, provided by MLIT

Hideyuki Kobayashi
Research Coordinator for Housing Information System
Research Center for Advanced Information Technology

国土交通省　国土技術政策総合研究所

National Institute for Land and Infrastructure Management
Ministry of Land, Infrastructure, Transport and Tourism, Japan

国土技術政策総合研究所報告
第42号　2011年3月14日

Research Report of NILIM
No.42 March 14,　2011

国土交通省版・景観シミュレーション・システム Ver.2.09 のアーキテクチャ

小林　英之[*1]

The Architecture of Landscape Simulation System Ver.2.09, provided by MLIT

Hideyuki Kobayashi[*1]

概要：

平成５年以来開発、改良されてきた景観シミュレーション・システムの目的別・言語別バージョンに関して、平成21年度に、住宅情報システム研究官が、整理を行い、Ver.2.09としてとりまとめた統合版のアーキテクチャに関して、当初からの開発の経緯、バージョン統合の考え方、及び全体構成と各部の動作原理について報告する。

キーワード：景観　シミュレーション　アーキテクチャ

Synopsis：

The research coordinator for housing information system coordinated the systems for landscape simulation developed by research institutes and local branch offices of the ministry since 1993 to be integrated into only one version 2.09 in fy 2009, and describes its history since 1993, approaches for integration, and architecture of the integrated version.

＊１：高度情報化研究センター住宅情報システム研究官
Research Coordinator for Housing Information System
Research Center for Advanced Information Technology

URL: http://sim.nlim.go.jp

はしがき

本報告が対象とする国土交通省版・景観シミュレーション・システムは、平成５～８年度に、建設省総合技術開発プロジェクト「美しい景観の創造技術の開発」の一環として、建設省建築研究所・土木研究所の共同により基本的な部分が開発され、建設省がライセンスを有するオープン・ソースのフリーウェアとして公開・配布されてきました。その後の各種現場への適用に加え官民共同研究、国際共同研究等を通じて改良を加えられてきました。平成 13 年に国土技術政策総合研究所が設立されました後は、主として高度情報化研究センターと、環境研究部により更に改良が加えられ、立体視、ネットワーク配信、GIS 連携などの機能が付け加えられてきました。しかしながら、これらの過程で多くの枝分かれバージョンが形成され、言語別・機能別に異なる実行形式(.exe)を使い分けるような状態となっていました。

平成 21 年度に、住宅情報システム研究官が、従来の各種枝分かれバージョンを整理・調整した上で、土木・建築・都市・住宅等の各分野に共通する機能を統合した基幹部分と、分野別の専門的機能を選択的に追加するプラグインから構成するアーキテクチャに統一し、更に言語に依存するプログラムを全て外部テキストに分離し、同一の実行形式を用いて様々な言語で使用できる Ver.2.09 としてとりまとめました。この統合バージョンにおいては、バージョンの枝分かれを生じることなしに、新たな機能を基幹部分とは独立したプラグインの形で柔軟に付け加えることが可能となっています。

本資料は、今後オープン・ソースに対してシステムの改良を行い、あるいはソースコードの一部を活用して別システムに機能の移植を行おうとするプログラマを主な対象として、この Ver.2.09 に至る開発過程を報告すると共に、その全体構成を解説し、添付の CD-ROM に全ソースコードを収録したものです。

なお、この報告が対象とするシステムは、サーバー機能を含む、まちづくり・コミュニケーション・システムの一部を成す、クライアント側の PC 上で動作する部分です。

国土交通省国土技術政策総合研究所

国土交通省版・景観シミュレーション・システムの概要

主な表示機能

・画像の視点を解析し、三次元モデルを前景画像と背景画像の間に合成表示する。
・三次元モデルを任意視点から透視図として表示する。
・テクスチャ、シェーディング、ワイヤーフレームの表示モードで表示する。
・平面図、立面図（東・西・南・北）の表示を行う。
・緯度経度季節時刻から光源を自動設定する。
・表示画像を画像ファイルとして保存する。またプリンタに出力する。
・移動経路を設定し、アニメーションを表示する。これを.avi 形式の動画として保存する。
・地形や市街地の中に存在する設計対象物に関して、可視範囲を解析する。
・築後年数を指定し、経年変化を表示する。
・立体表示ができるディスプレイ、プロジェクタに対してステレオ表示を行う。
・指定したオブジェクト、全地物の影の表示を行う。
・各種条件による表示をシーンとして記録し、LSS-S 形式でファイル保存する。

主なモデリング機能

・任意の平面図形を点列から生成する。
・既存の平面図形に穴をあける。
・平面図形に高さを与えて立体を生成する。
・断面群または断面と折れ線から掃引体を生成する。
・道路・河川の断面形状と中心線軌跡から立体形状を生成する。
・外部関数を用いて基本的な原始図形をパラメトリックに生成する。
・ユーザー定義による外部関数で、パラメトリックな応用図形を生成する。
・データベースに登録された三次元図形（点景など）を配置する（単体・リニア・エリア)。
・面・立体・同色面に対して、カラー、テクスチャ、マテリアルの設定を行う。
・コンバータにより、数値地図、ステレオ空中写真解析結果、既存 GIS データ等から地形、周辺市街地を作成する。また、電子納品成果、CAD データ等を利用する。
・地形を加工する（図形演算による切断、法面生成等)。

連携機能その他

・景観データベースにより既存データを登録・検索・利用する。
・市街地生成結果を動的に表示する。
・WEB ブラウザと連携し、ネットワーク上にある三次元データを取得して表示する。
・外部関数、およびプラグイン DLL により、機能を追加する。
・メニュー、ダイアログ、ヘルプ、メッセージに用いる言語を動的に切り替える。
・ユーザーが追記可能なヘルプ等、プログラムの変更無しに新たな言語に翻訳移植する。

国土交通省版・景観シミュレータの活用実績と、今後の応用

技術概要

○国土交通省が開発したフリーウェアで、普通のパソコンを使用して簡便に景観検討を実施

○道路・橋梁、河川、都市、ダムその他様々な事業に適用可能です（内部検討、情報公開）

- 景観総プロ(平成５～８年)で基本部分を開発、その後現場で適用しながら安定性・信頼性を高めてきました。
- 一般の事務用パソコンを使用し、CAD、GIS の経験の無いユーザーを想定しています。
- デジカメ画像を用いた簡便な写真合成作業から、本格的な GIS ベースの空間検討まで幅広く利用可能です。
- 景観データベースにより、既存の様々な部品を利用し、手軽に景観検討できます。
- CAD データ、GIS データ、CG ソフト等とのデータ交換機能を向上させました。
- 韓国との共同研究を通じ、国際化対応（日本語依存部分の整理）と現場適用のノウハウ交換をしています。
- 国土技術政策総合研究資料でマニュアルを出版した他、使用方法について、インターネットからも入手可能です。

適用事例

○土木系・建築系のモデル現場における適用

○平成１３年度に公募した１５のまちづくり現場で、計画内容公開＋コミュニケーションを実施

年度	事業	概要
1996	福岡県住宅供給公社峰花台団地建替(住宅)	現場事務所で、再入居予定者を対象に、初めて評価実験実施
1996	福島工事事務所（当時）(橋梁・道路・河川・公園)	周辺部地形まで本格的な３次元データを作成、川原の砂利、周辺樹木、近景・遠景を再現
1997	沖縄北部ダム	空中写真から作成した地形データを利用、亜熱帯植物を景観データベースに拡充
1997	福井駅前再開発	市街地と再開発計画、連続立体交差を三次元で作成し、北陸テクノフェアに出展
1997	三陸国道	周辺地形＋市街地＋施設のデータを作成。斜面にのり面、高架道路、トンネルなどを作成
1998~	福島都心東土地区画整理	現場事務所の担当者による本格的なデータ作成と地元説明への活用
1999~	幕張駅東口土地区画整理	コンサルタントから派遣職員による現況・計画案データの作成
2001	まちづくり・コミュニケーション実験	１５の現場で３次元データを構築し、コミュニケーションを実施
2002	みちのく国営公園	GIS と連動させつつデータ作成中

例１：峰花台団地：シミュレーションと、竣工後の比較（1996）

再入居予定者は、外から眺めた団地の景観よりも、入居予定の住戸からの眺望や、道路・歩道からの視線・プライバシーに関心。色彩は、その後地味なものに変更。評価した再入居予定者の多くは高齢の女性。模型やパースより好評。

設計図通りに入力した、屈曲した団地内通路が、その通りの形状で竣工。駐車場入口の前を平坦に、間を急な坂にした結果であるが、車椅子などの通行には支障が考えられる。景観のみならず、このような図面だけではわかりにくい事項も検討する価値がある。

国土技術政策総合研究所 National Institute for Land and Infrastructure Management

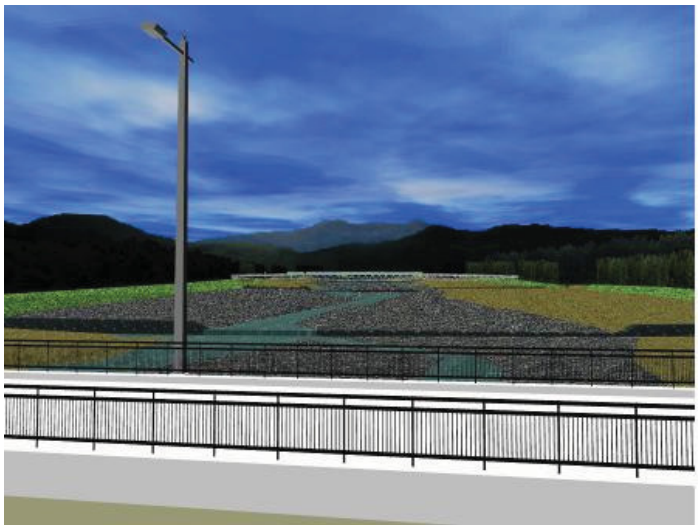
〒305-0804 茨城県つくば市旭１　URL:http//sim.nilim.go.jp/MCS

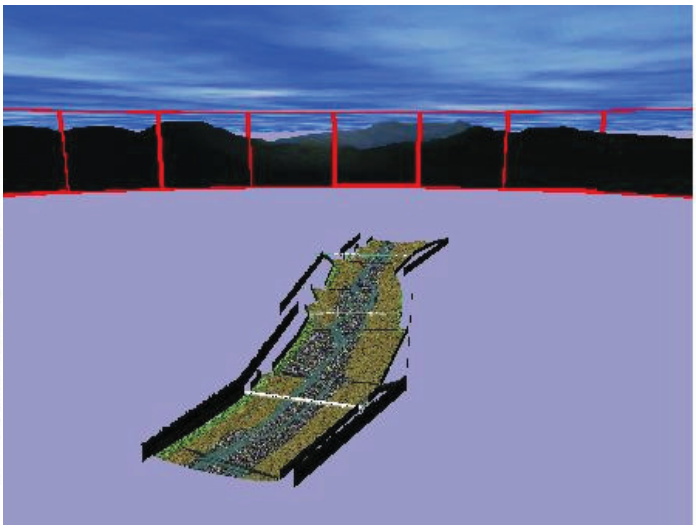
問合せ先：高度情報化研究センター　住宅情報システム研究官　Tel:029-864-4433 E-mail:keikan2@nilim.go.jp

環境研究部　緑化生態研究室　Tel:029-864-2742 E-mail:keikan@nilim.go.jp

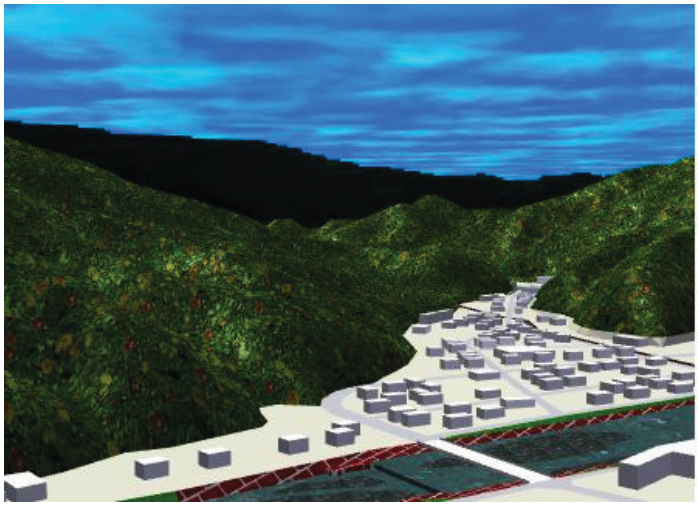
国土交通省版・景観シミュレータの活用実績と、今後の応用

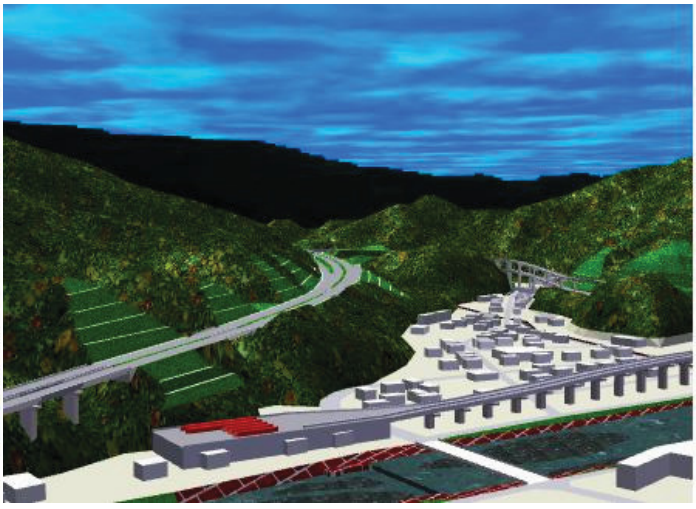
例２：川原の形状・砂利のテクスチャまで再現した。周辺の山や樹木は、屏風にテクスチャを貼り表現。



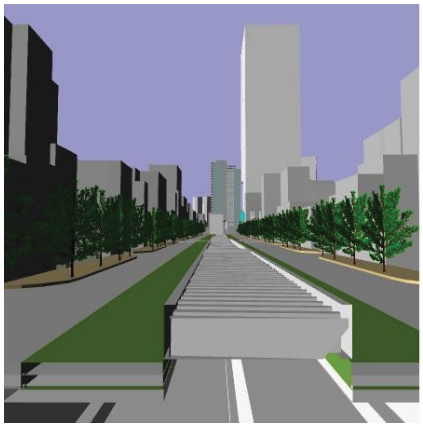


例３：ジャンクション（施工前：現況と、施工後：計画案の比較検討）





例４：市街地再開発・連続立体等（2001：学生等のクリエータが、景観シミュレータのみを用いてモデリング）



ステレオ空中写真から作成した地形＋市街地の上に、再開発建物、及び地下の道路を作成。幾何学的演算を行い、既存の建物・地形から不要となる部分を削除して合成した。結果は、インターネットを通じて配信し、これに関する意見を、テキスト、画像、三次元データの形で受付け、提案としての適切さを審査した上で掲示板に公開する。

国土技術政策総合研究所 National Institute for Land and Infrastructure Management

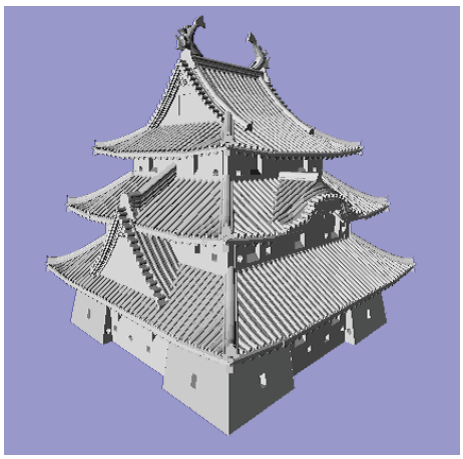
〒305-0804 茨城県つくば市旭１　URL:http//sim.nilim.go.jp/MCS

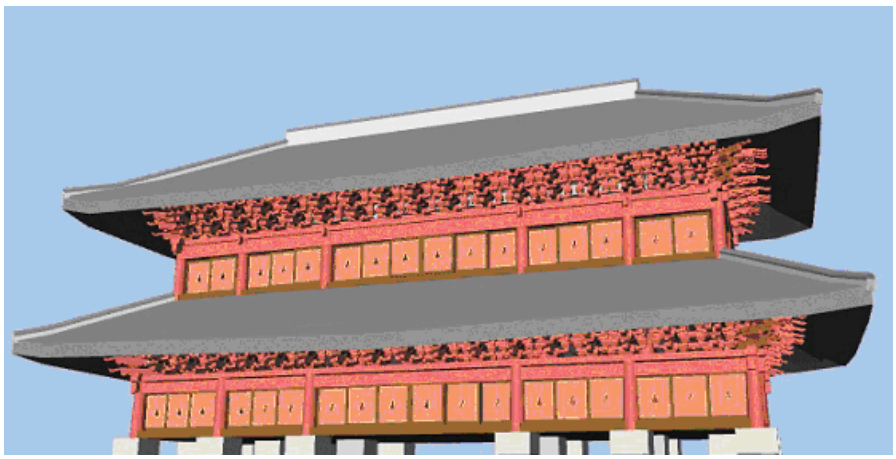
問合せ先：高度情報化研究センター　住宅情報システム研究官　Tel:029-864-4433 E-mail:keikan2@nilim.go.jp

環境研究部　緑化生態研究室　Tel:029-864-2742 E-mail:keikan@nilim.go.jp

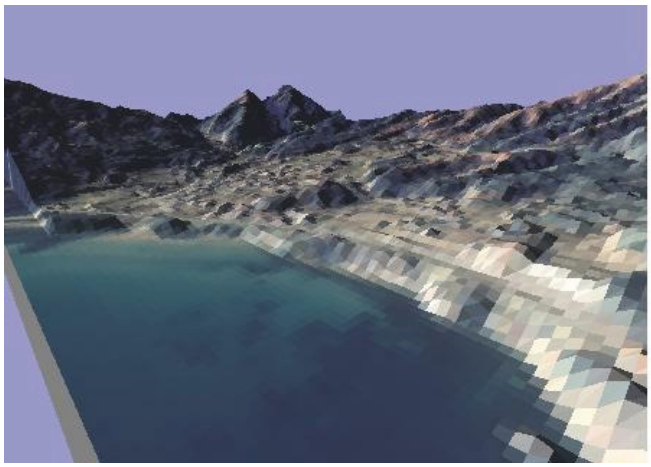
国土交通省版・景観シミュレータの活用実績と、今後の応用

例５：部品から構築して建築物を作成した例





例６：GIS データとの連携、自然の山林を再現した例



今後の応用

○様々の行政手続きの中への応用

○ライセンスの制約がない部品として、大きなシステムの中に組み込んで利用

実務面：各種現場への応用事例を広げ、マニュアル・サンプルデータなどの拡充と普及に努めます。今後は、実際の景観検討が必要となる行政上の手続きとの関連を重視します。また、このために、実際の実務に即した機能・操作性の向上に努めます。

例１：インターネットでのデータ転送による登録や公開（まちづくり・コミュニケーションシステム）
例２：GIS や CAD システムとのデータ交換（dxf, vrml, shape, dem 等）

技術面：ライセンスの制約なしに、より大きなシステムの一部に組み込むことにより、多様な応用範囲が可能です。

例１：建築確認　モデラーにより作成した３次元データと、帳票エディタで作成した申請書データを、インターネットで送付します。単純な検討事項は、入力段階でチェックすることが可能。
例２：都市の歴史　三次元オブジェクト（建物やインフラ）にタイム・スタンプ（建築年、除却年）を付け、景観シミュレータの経年変化機能を利用して、指定した時代に存在したオブジェクトだけを表示する。
例３：再開発ビルの物件広告　オブジェクトに属性として、物件データ（データベースの ID）を付け、参照を行う。また、ユニットを選択して、内部の間取りや眺望を確認します。

国土技術政策総合研究所 National Institute for Land and Infrastructure Management

〒305-0804 茨城県つくば市旭１　URL:http//sim.nilim.go.jp/GE

問合せ先：高度情報化研究センター　住宅情報システム研究官　Tel:029-864-4433 E-mail:keikan2@nilim.go.jp
　　環境研究部　緑化生態研究室　Tel:029-864-2742 E-mail:keikan@nilim.go.jp

データ活用による景観シミュレーションの試み

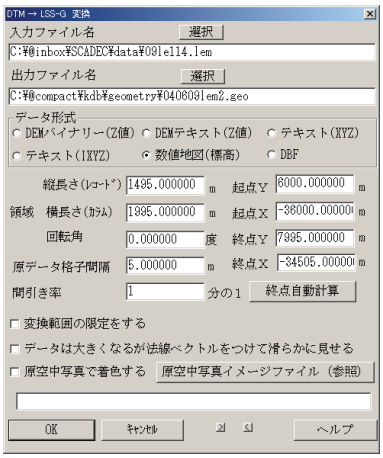
（2008年度　国土技術研究会）

最近、高精度数値地図、様々なGISデータ、電子納品データなど、形状を表現した電子情報が普及しています。三次元データであれば、コンバータで変換することにより、また二次元データであれば、それを紙図面に代わる作業下図として活用することにより、事業後の景観を視覚的に確認し、プレゼンテーションすることが手早くローコストで実行できるようになってきました。

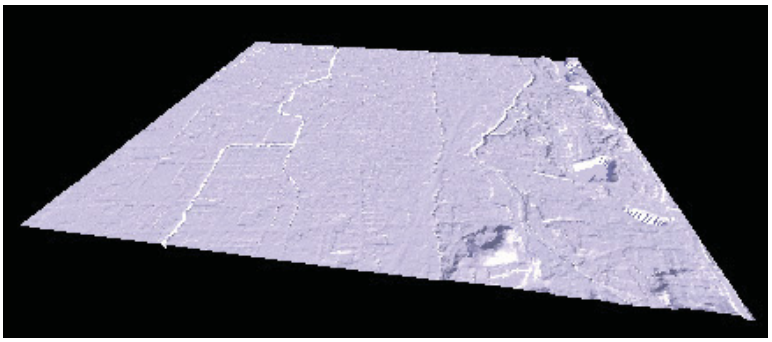
本展示では、数年来、ファイル・コンバータ等を改良して、既存データの景観シミュレーションへの活用を試みたいくつかの実例を表示し、具体的な作業方法や、工数、得られる画像の品質等について解説します。

1．地形に関する既存データの利用

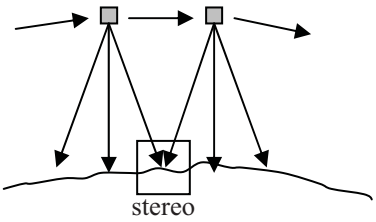
（1）遠景の山などは、50mメッシュ数値地図で十分ですが、市街地周辺等では、5mメッシュ数値地図が有用です。

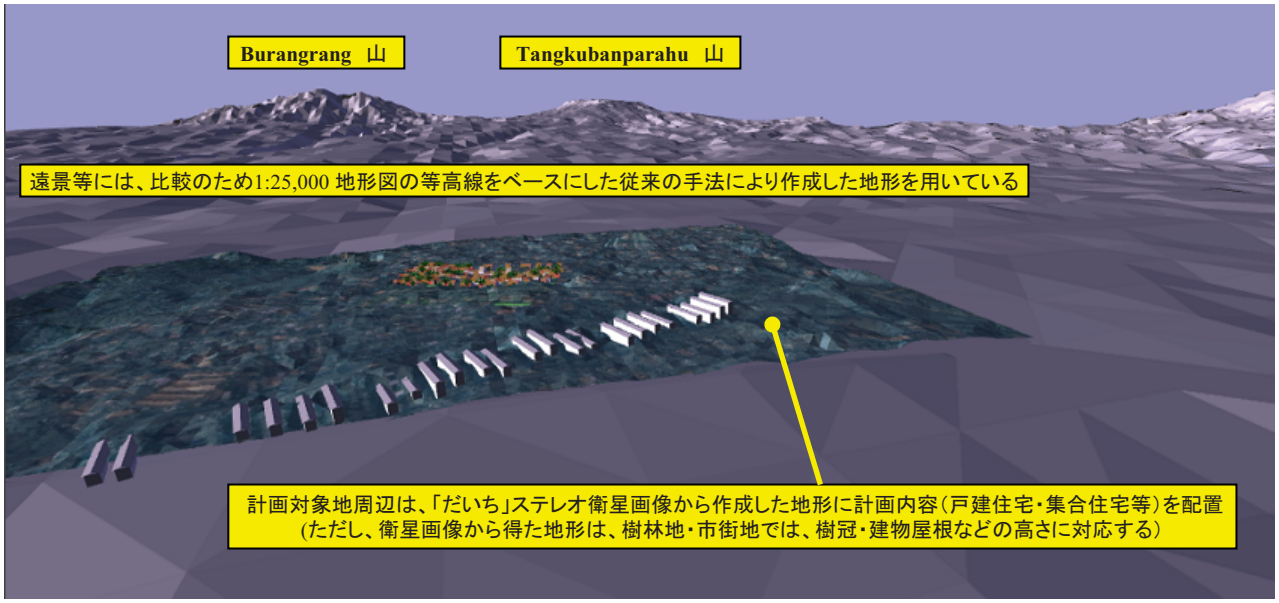


格子点の高さデータから、コンバータで地表面を作成し、構造物を乗せたり、切土・盛土の図形処理を行います。



（2）既存測量データの得難い途上国では、「だいち」衛星により撮影されたステレオのライブラリ画像から、7．5mメッシュ精度の地形データ（インドネシア・バンドン市の場合）がローコストで得られます。





国土技術政策総合研究所 National Institute for Land and Infrastructure Management

〒305-0804　茨城県つくば市旭１ URL:http://sim2.nilim.go.jp/GE E-mail:keikan2@nilim.go.jp
問い合わせ先：高度情報化研究センター　住宅情報システム研究官 Tel 029-864-4433

<A B S T R A C T>

The Architecture of Landscape Simulation System Ver.2.09, provided by MLIT

By: Kobayashi, Hideyuki. DR.Eng.
Research Coordinator for Housing Information System,
Research Center for Advanced Information Technology,
Tel +81-298-64-4433(direct) Fax +81-298-64-0565
e-mail keikan2@nilim.go.jp URL : http://sim.nilim.go.jp

0. Outline: basic purpose, features and functions

This system is a kind of open source 3D graphic system, developed by Building Research Institute, and Public Works Research Institute, under Ministry of Construction (1993-2000) at that time, and succeeded by the National Institute for Land and Infrastructure Management, Ministry of Land, Infrastructure, Transportation and Tourism which was newly organized through the process of restructuring the previous ministries and related institution (2001-now).

The system was developed for the purpose to promote local branch offices and their stuffs to perform CG simulation and presentation without no heavy training, and provision of special budget, in order to enhance the visual design and evaluation of regional development projects and related public works.

0-1. Outline of the system

The system consists of following components:

(1) Data Base of 3 categories:

- Past examples of projects (provided by public sector, browsed by yuu.exe)
- Building materials (commercial base, browsed by zai.exe)
- Elemental components of landscape (non commercial base, browsed by yuu.exe)

(2) Sim.exe : main load module for modeling and browsing:

- Various viewpoint setting, walk-through, etc. of 3D landscape data.
- Modeling and editing 3D data
- Analysis of image data and synthesis of photo and 3D model
- Time dependent materials (deteriorate / mature)

(3) Maju.exe : city planning & build up process simulation

- Lot subdivision pattern must be prepared as the stage of events.
- The simulation continues hundreds of years.
- Houses and buildings appears and disappears with probable length of life.

- Types of houses which newly appear follow the statistical probability.
- Shape of houses which newly appear follow the city planning scheme.
- Sim.exe is used as an output device like three dimensional printer.

(4) File converters

- Dxf format
- Minicad text format
- DEM format
- VRML format
- SXF format (official data format for archiving designed and constructed land & infra)
- Other formats used for simulations: urban fire spread process, etc.

(5) User-defined parametric components

XXX_D.exe (GUI dialogue)

XXX.exe (3D model generator)

(6) Plugin dll

Sim.exe dynamically loads the plugin dll-s when the user selects it from the list (plugin.tab).

All the library functions and static variables used by sim.exe are exported to the developed dll through sim.lib, so that the plugin dll-s can perform as a part of sim.exe with full access to the system variables and status.

Such examples are provided, as a result of re-structuring the derived versions of sim.exe:

Land.dll: To manipulate and modify the free form of the land

ParkRoad.dll: To design foot path in the parks

Tunnel.dll: To append a kind of pipe in a mountain to achieve short-cut road.

Nori.dll: To evaluate new artificial slopes that accompany to the constructed road.

0-2. License

- Copyright(C) Ministry of Land, Infrastructure, Transport and Tourism
- Free ware, and source code is disclosed (open source)
- Re-sale without additional value is eliminated
- International portability

Japanese, Korean and Indonesian are available (English is under construction)

0-3. Conditions for installation

The platform is PC installed with Windows 2000, XP, VISTA and 7.

Disk space: 70MB (minimum)

Memory: 16MB (minimum)

Graphic Display: 800*600(minimum), 32768 colors (minimum)

Recommended OpenGL accelerator

(In short, any recent models of PSs and Laptops are useful for this system, and the requirements depend on the size and quantity of data that users wish to handle.)

0-4. Users

Target user group of the development was construction-related staffs of local branch offices of the ministry, local governments, relevant corporations, consultants, architects, planners and designers, with educational background of senior high school, with self training less than 3 months. They will not operate for daily use, but sometimes require the system for their work

0-5. Major functions and characteristics

- File load/save convert of 3d model, scene
- Analysis of viewpoint of image data, accurate montage with 3D data
- Basic 3d modeling (primitives, typical elements, user defined parametric...)
- Various plotting functions on ground
- Earth work (cut and pile, calculation of amount of soil)
- Sweep functions (use section and orbit: to create road, river, etc.)
- View point setting, walk through
- Analysis of visible area
- On line simulation of build up process (evaluation of rules and regulations)

0-6. Purpose of this report

This purpose of this report describes the inside of the system, to support the further development of additional functions, in the form of appended external functions or appended plug-in dll-s. However, this report will also be useful to those who wish to develop any similar systems for different purposes (tourism, agriculture or transportation etc.). Because all the source code is open, anyone can take some part of the program of this system for the purpose to develop own system. This report may also be useful to those trials.

0-7. Contents of this report

1. History
2. Basic geometry and data structure
3. Own file formats
4. Structure of source codes
5. External functions
6. Plugin dll
7. Graphics process
8. File converters

1. Process of Development

The system was initially developed through an Integrated Technology Development Project, titled as "Technology Development for Landscape Amenities", undertaken by the Ministry of Construction (1993-96). Among the 6 topics, the fifth topic titled "Development of Landscape Simulator and Landscape Database" was assigned to the Building Research Institute (BRI) and Public Works Research Institute (PWRI), under the ministry. Unlike to the usual case, the task was not shared among the two institutes according to the "objects" (namely, building and social infra), but according to the "functions". The development of the system and software was undertaken by BRI, while the database was undertaken by PWRI.

1-1. Background

The reason why MOC started this project was the citizen's emerging consciousness on the impact of development projects to the environment and landscape.

Previously, design and program was decided mostly considering the technical and economical conditions. However, it has become frequent that several citizens and opinion leaders protest and criticize the negative impact of the development. In order to conquer the situation, technical staffs of local branch offices of the Ministry had better to acquire the ability of evaluating the impact of any project they are engaged in.

However, most technical staffs for civil engineering in our country have no educational background on designing and drawing, but only calculating the mechanical problems. Landscape-related matters are mostly considered by the consultants, who

have almost similar educational back-ground, and sometimes assisted by designers and architects.

Therefore, they seem to need smart tools to visualize the impact of their design to the existing landscape. This had been done through perspective drawing or rather expensive CG simulation done by external experts and accordingly the re-evaluation session was not enough frequent.

In 1992, preliminary research & survey on existing needs and technologies was done by the team. At that time, existing systems were mostly English based ones, and required fairly expensive graphic workstations, and the systems themselves were also very heavy (general purpose) and expensive. It seemed almost impossible to apply those systems into every branch offices and let them to operate.

While, the performance of PC, especially the graphic processing, seemed to start advance rapidly in the near future, and conditions for developing appropriate system is becoming ripe.

By considering those conditions, the team decided to develop original simulation system specially designed for MOC projects, as free-ware (to be distributed without license constraint, and maintenance free). In order to adapt to the rapid growth of hardware, which seemed difficult to forecast, the system would be multi-platform, written mostly by using ANSI-C language.

1-2. Initial development

The initial prototype (man-machine interface and skeleton functions) was elaborated on INDY provided newly by SGI, comprising merely skeleton GUI operations and simple photo montage function, while data structure was designed for the next step.

The second version realized major viewer functions and time-related alteration simulation (deterioration of materials after completion, seasonal change of trees and forest, lighting conditions depend on climate , time and weather, etc.). At the same time, the team started to convert into the PC version utilizing the OpenGL functions started to be provided with WindowsNT 3.51.

The third version utilized the GIS data obtained from stereo air-photo and full 3D functions (automatic move of viewpoint, cut and fill of slopes, etc) were added.

In the final year (1996), the team made the installer with sample data, and started to distribute, even though there were so many defects.

1-3. Testing and Debugging

In the end of the project period, the major functions originally designed were achieved, however, stability was very poor. The only viewer functions was sustainable, but modeling and editing functions could not be continuously operated. Therefore, after the

project completion, requesting budgets from the local branch offices (for survey and design), the team continued to improve the system, until 1998. While this process, several requested functions were added, like advanced editing functions, newly requested networking functions, etc.

1-4. Application to the development site

The first application was tried in the bridge project in Fukushima prefecture in 1995, on INDY, and re-construction of Flat in Fukuoka prefecture, on PC. In former case, the 3D data were prepared by the team (not by the end user), while the second case, the data was prepared by the newly experiencing local soft-house, chosen through open tendering, showing that the cost for preparing the visual media is almost half as providing model or perspective drawing.

After 1997, the system was distributed to many local offices through inter-net and CD-ROM. Some of local staffs elaborated very complicated data and reported (complained about) many bugs and troubles.

1-5. International co-operation

In 1996, Korean governmental Rural Research Institute started to develop landscape simulation system, and dispatched survey team to Japan. Realizing that there is no regal and technical constraint on transferring and translating Japanese free-ware, we jointly proposed international joint research. This was adopted in the Japan-Korea agreement on co-operation in Science and Technology, as "Application of Landscape Technologies for Regional Development".

In the August 1997, translation of Japanese latest version into Korean-Hanguel system was successfully achieved. This was achieved through a composite team within 3 days (8 translator, programmer, civil engineer etc.). Japanese side will also expect advanced functions through role sharing, and exchange of experience of actual application on the site.

1-6. Co-operation with private sectors

Since 1997, private sectors are promoted to develop application functions jointly. In this case, especially, the companies which have already developed some kinds of software or data are invited to jointly develop file converters to achieve that the privately developed software could be utilized with free-ware jointly. If seen from the user of the free-ware, some kinds of data will become easier to make, by purchasing private products. However this choice is not or should not be compulsory, but alternative. If seen from the private sector, the potential target user will be appended.

Through this scheme, 3D Modeling software, and GIS data are connected to the LSS system through developing file converters jointly.

1-7. advanced research and development

After the completion of the project, several related research and development are undertaken, to utilize the basic graphical functions:

(1) Simulation of social and urban phenomena:

To simulate the change of buildings, according to the lot subdivision pattern and building regulations (applied to the land re-adjustment projects).

(2) Project simulation

To simulate expected progress of the project, and to feed back the forecasted result to the initial designing stage.

The results of those related projects were realized in the forms of separate software, which generate the 3D data to be transferred and visualized by the LSS system running somewhere within network, dynamically. The transferred 3D data can be treated as usual 3D object data in the local environment of LSS system.

2. Application to landscape design

2-1. Processing digital images taken from sites

Digital photo data is very useful for creating a scene. The simplest way is to use as background (like wall paper). The simulator has a function to analyze the rendering-related parameters (view-point, view-center, focal length, etc.) from more than five viewable points which are given xyz coordinates by the operator. After that, the simulator will render any 3D object in the position just same as the camera (fig 1.).

The system also utilizes the image data for texture. For example, in order to model a proposed urban renewal project, the texture data taken from the buildings surrounding area, by using digital camera, can relief the data entry work, and makes it possible for less skilled local staffs to operate.

2-2. Modeling and database

The team assumed that those who will create some complicated shape are already good at operating some CAD system. File converter will be more suitable for them.

For less skilled operators, the system provided 3 types of database, with various plotting functions (single, linear, area). The database is consist of three kinds, namely, (1) past and popular examples of buildings and public works, (2) commercial landscape materials, and (3) popular components of landscape, like trees, street furniture, popular buildings, mobiles, etc.

Database (1) will be supplemented by the local branch offices of the ministry.

Database (2) will be provided by the makers and providers for the commercial purpose.

The team organized the landscape material committee to classify the many kinds of components and prepared the example core data set during the project period, along

with software for maintaining the data.
Database (3) will be provided by research institutes.

Beside these three types of 3D data said above, which are rather fixed in shape, the system also provides parametric components, described as ANSI-C functions, to make geometrical data more compact, and to make the system more flexible. If a user elaborates some new functions, and GUI dialog to give parameters, then he or she can add it to the system. For example, if a function which gives 3D shape of a clock, based on the parametric data of time, e.g. CLOCK(TIME) and the function is referred as :
G1=FILE(CLOCK,"2:30");
Then, a 3D shape of a clock showing the time of "half past two" will appear. The GUI part is a dialog box for user to edit the time.

Simple road or river can also be a parametric component, described by section shape (file) and orbit shape (file).

After creating the shape, some boolean calculation is done with existing shape of land surface.

2-3. Editing texture data

At the start point of development in 1993, we assumed that the texture data will be a kind of database for building materials, provided by the supply side. Therefore, we tried to collect texture data of popular building materials (timber, concrete, trees, grass, etc.), and provided them usually in the form of tiling data (1m * 1m). When users try to apply them to the 3D model, the system can automatically calculate the texture coordinates of each vertex of polygons, in repetition.

Recently digital photos with high quality/resolution are becoming popular, therefore preparing texture data is becoming far easier. Therefore, users can also easily make, for example, a texture mapped 3D data of an existing office building. In such case, functions to make a orthogonal texture, which will be applied only once on a whole wide face.

Since 1997, we are co-operating Fujitsu company who has developed Real-Modeler system now on sale (c.a. 200 k-yen). Actually, the system realized almost similar function of our system to analyze the viewpoint and camera angle of a photo (image data). For example, an object in a photo could be assumed an "cube", the system abstract edges from the image, and semi-automatically calculate the dimension of the object and camera position. After that, the system subdivide the image to each seen surface of the cube, and output the orthogonal texture image data.
However, the purpose (object) of the system was very general, designed by computer programmers (not by designers or planners). In order to make it easier to make a 3D

data of existing buildings, we co-operated to make a file converter, and we requested them to enhance the system to become easier and more practical for our field. By using this, a 3D model of simple building can be obtained in a few minutes. This can help to make the data of existing landscape, and also useful to collect the local components to be assembled in the future image.

2-4. Plotting objects and editing the scene

We assumed that most users don't want to make primitive models by hand, but like to choose components from data base and plot them in the scene. Therefore, we tried to enhance the plotting functions.

Repetitive plotting is done by only one actual 3D data, appending link matrixes defining different coordinates, causing less bulky data size.

Automatic plotting along a line, spline curve, and certain area is possible, with desired density. Size and direction of objects can be randomized. By using those functions, e.g. street trees can be easily plotted.

Geographical Survey Institute, our ministry of construction, examined and authorized the technology for stereo air photogrammetric, provided by c.a. 8 private companies in 1993, and after that, we can get DEM format data from each service in the same data format. We made a file converter from DEM to lss-g format, which is recognized as "ground" data. Therefore, we can plot any objects on the orthogonal view window, and the height can also be automatically determined by the system, in relation to the ground data.

2-5. Dialectic presentation

The most popular way of presentation is done in the meeting room on development site, inviting citizens' participation. Video projector (LCD) connected to the note-book type or desk-top type computer shows the future landscape on the screen.

Current request of the users is "dialog" type session. If someone requests to change some components of the project, planners are requested to change the 3D presentation immediately, to evaluate it. On contrary, conventional usage of CG presentation had tend to postpone the corrected and altered design as planners' homework.

Usually, the most interesting aspect of participants' is not the physical shape of the project, but economical impact. Therefore, we have to prepare the contents carefully, in order that the presentation of alternative plan will not give to much impression of economical difference (price of rent, amount of compensation, etc.), along with adequately arranged questioner.

Sometimes, evaluation session in the meeting room together tends to influenced by the loudly speaking persons. In such case, monitoring the evaluation of silent majority

should be arranged. For the time being, said questioner will help this(SD method, etc.). In the future, when PC connected to internet shall become more popular, individual evaluation done by citizens watching personal computers prepared in their home will replace this. In this scope, the simulator has the function to send commands to other simulator through TCP/IP protocol, by sending compact code to realize the simultaneous scene change in remote computers.

However, this function is actually not yet tested on site.

2-6. Evaluation sessions on site

Sometimes, evaluation session in the meeting room together tends to influenced by the loudly speaking persons. In such case, monitoring the evaluation of silent majority should be arranged. For the time being, said questioner will help this (SD method, etc.). In the future, when PC connected to internet shall become more popular, individual evaluation done by citizens watching personal computers prepared in their home will replace this. In this scope, the simulator has the function to send commands to other simulator through TCP/IP protocol, by sending compact code to realize the simultaneous scene change in remote computers.

However, this function is actually not yet tested on site.

2-7. advanced simulation of social phenomena

In case of re-construction of apartment house, or urban renewal, the future landscape will be designed by a few planners and designers in harmonious way, hopefully. This purpose will be fulfilled by simple modeling function, like well arranged popular CAD system. However, in case of land-readjustment project, the future townscape will consist of various individualistic construction activities. For controlling this, “detail plan system” is becoming more and more popular. In such case, manual data entry of individual houses whose behavior is statistical is too exhaustive. Therefore, we provided functions to simulate such kind of social phenomena (individual construction regarding common rules), by using typological building, described as parametric objects realized by external functions (several XXX.exe 's which receive the land conditions and generate the 3D shape data, according to the rules or regulations. By using those functions, in the evaluation session, we can quickly compare the alternative rules and regulations, by watching and comparing the physical statistically forecasted future landscape.

2-8. Evaluation through Networking

In 2001, when NILIM was established, the “Communication System for Town Planning” was developed, which also included the server-side programming. The web-based system is described in the technical note of NILIM No. 134 (Sept. 2003).

In this system, sim.exe was used as a viewer installed in the client side, and used to examine the 3d data provided from the web server. For this purpose, minimum functions are selected and sim.exe was simplified. However, if a user wishes, additional functions are also available at web servers.

The setup was classified into 3,

(1) Minimum: for rapid browsing the 3D data provided from web servers.

(2) Compact: for users who wish to edit the said 3D data.

(3) Full-set: for users who cannot access to the web

Those "purifying" works for the system encountered many troubles, especially on the Windows ME, which seemed more strict and severe for memory management. However, identification of the cause of troubles was easier than previous versions of Windows, because the troubles occurs immediately after the cause appears.]

Stereoscopic presentation of 3D data was also introduced at that time, which is becoming popular today.

3. Examples of application

Most users are civil engineers. The ministry of construction had c.a. 300 local branch offices on site, mainly for public works, while architects are confined in 8 major provincial headquarters, to design and build public buildings.

For urban development and small-scale construction, 47 prefectures and c.a. 3000 local governments will be the users.

3-1. Renewal of housing complex

This case is the earliest trial of simulation in 1996. At that time, the system performed only a simple viewer of the 3D data, provided by using other CAD system (Intergraph's Microstation) through converters (DXF format).

The housing corporation of Fukuoka prefecture planned to re-build the old 4 storied apartment (rental) consists of several blocks built in 1950's, to become a plaza style complex of housing and commercial use. A design competition was done, and an architect in Fukuoka city won the prize. We are afraid of negative impact of evaluation session by using the system still under development, and did the evaluation session after the design was already announced to the inhabitants, through perspective drawings and models.

We tried a tendering session to select a soft-house for data entry in Tsukuba, over 1000 km distant from site, afraid of information leak to the contractor who will participate in the tendering for the construction.

The cost was merely 50% of the cost, which the corporation spent for preparing said drawings and models (more than 3 million yen). The cost covered the data entry of

questioners and simple statistical analysis.

The evaluation session was held on the site office, by preparing Intergraph's TDZ 300 and 21' monitor display (leased). This was the best (fastest) way at that time, and might be the first trial of such simulation on actual construction site in our country.

Many of 25 respondents on site were elderly ladies (20, without background of education for drawing civil engineering or architectural design. We provided a questionnaire on both "design of future home (SD method)" and "the presentation system (simple questions)".

By walking through the future 3D complex, most of them replied that 3D graphics is easier to understand the design, than drawings (birds' eye view) or models, even though the monitor display is too small. At that stage, they were more interested in the choice of units than the design of building itself. Therefore, we were requested to show the scene from several units they are considering choosing.

We came to realize that providing too many adjective-pair in the questioner for sensory evaluation is hard for elderly people to fill over. Based on statistical analysis, we have to prepare smarter and more effective set of items.

After this session, low price graphics card (OpenGL) has become popular, and LCD projector came. Therefore, some of said problems were solved.

3-2. Land re-adjustment project

Fukushima city is now conducting a land re-adjusted project in the inner-city built up area. They provided the local office on site in 1997, and the stuff of the municipal officers is working on site. The mayor was interested in the dialectic CG presentation and provided budget for machines and equipment.

However, same as other local governments, they don't like to provide budget for data entry, but assigned a young officers to be in charge. Their educational background is senior high school (not technical).

The stuffs elaborated to provide the data only by using our still buggy system. At that time, viewer functions had well performed, but modeling functions were not so sustainable, causing frequent memory leakage and memory access violations. GUI was also not so friendly to the users who don't want to prepare CAD system, but like to make up whole complex data only by using our system. The stuffs sent us so many various bug reports and proposal for betterment.

Generally, this type of project last very long (sometimes more than 20 years). In order to cope with such situation, Addressing of re-adjusted lots and data management is essential. In the future, equipment and software will drastically change. Therefore, we have to be careful of initial design for data management.

The stuffs are now entering the data from land map by using digitizer, and convert it for the system. The lot shape data, including city planning conditions, are utilized for the said simulation of statistical social phenomena. For this purpose, more detailed data format for lot conditions was made possible to assign, like a corner lot, long lot which faces 2 streets, inner-block lot which faces only a narrow footpath, etc.

They also requested us to develop a modeler to make 3D model of streets and crossings, according to the technical standard of our ministry, which easily create a model by giving a few parameters. We assumed that the land condition is not necessarily flat.

The project is still ongoing, and they utilized our system to determine the width of the planned streets, and to consider the desired height of buildings along them.
However, against our expectation, the system is mainly utilized among the governmental officers, because the most urgent issue is not negotiation with inhabitants and land owners, but the budget resource sharing among several public bodies, in this severe economical situation. The progress of the project itself is far delayed than initially scheduled.

3-3 Urban-renewal projects

Makuhari area, conducted by Housing and Urban Development Corporation, Chiba city, and local co-operative (three zones) utilized this system to presentation and discussion on planning among related staffs of the corporation and local government and land owners, inhabitants. Following data were provided:

(1) Textured 3D model of existing town, utilizing Real Modeler(TM of Fujitsu co.ltd) which process the image data taken by digital camera. The 3D data of each existing buildings (houses, shops, offices, etc.) were recorded in separate LSS-G file, while total town area were integrated by using plotting function of sim.exe.

(2) 3D model of surrounding area, including land and buildings was provided by Asia Air photo Company as a test case. Browser and converter were worked out, and any desired partial area, chosen by users, can be converted into LSS-G data. This data is useful for checking the landscape view from the planned high-rise housing unit, before making any model for planning (only by setting viewpoint at location of future balcony etc.) This data is also useful as base for constructing 3D model for the project site.

(3) 3D model for plan was elaborated for several stages, including housing blocks, road (under path for railroad, underground bus terminal etc.), and discussed among designers.

Through this experience, it has become clear that before city planning scheme, there are many obstacles and difficulties to prepare the feasible plan, containing financial

negotiation, proposal for expected commercial capitals for the planned building that are rather interested in the economical aspect than 3D shape. Therefore, another research project to combine several non physical (visual) factors with physical planning is now undertaken. In this research, simulation of the process of urban renewal project (many steps of procedure) and the function of site office is elaborated, in the shape of "VIRTUAL SITE OFFICE" on WEB, to which any related people can access and participate into the process. Clients are classified into 4 categories, namely:

a. Project managers (corporation, local government, and related consultants, etc.)
b. Land owners
c. Citizens and inhabitants in the surrounding area
d. Interested private sector and candidates of new comers (future inhabitants)

4. Distribution and acquisition

(1) Download from web server

http://sim.nilim.go.jp/MCS

Anyone can download the installer for the whole system, and also refer to the source codes. Until now, the system is for Japanese, Korean and Indonesian users only.

(2) CD-ROM version

In May 1997, the trial version (2.01) is distributed in the form of attached CD-ROM of a magazine on Landscape Design. In November 1997, the more debugged version (2.03) with textbook for self practice was published from our Building Research Institute, and distributed to anyone who requests. This was revised in July 2000 as version 2.05, and published from Building Research Institute (Building Research Data No.96).

(3) Multi-language approach

In 1997, we co-operated with Korean Rural Research Institute, who would start the similar development project, and translated our system into Korean language to enhance the start line of them. It took 3 days to translate the whole system, by 8 persons, consist of professors, researchers, students, professional programmers and translators. Recently, they started to distribute in the form of CD-ROM.

5. Development of applications

During the first half of 2001, a set of functions were developed and the successive release of the data to the public began, then in the second half of the same year, we undertook the challenge of developing new functions which are not possible by improving existing functions.

(1) Examination function

In the past, when a project body received opinions from citizens through a web-based

dialogue (not including three-dimensional functions), the officials in charge often examined their contents, manually selected and edited the contents that should be released, and prepared a public release page. And some regional governments provided completely unrestricted billboards for people to present their views. The function that has been developed so that examinations are performed more systematically selects examiners from a list to examine the opinions and automatically displays only proposals that are approved. A proposal can include images and three-dimensional data.

(2) Increasing display speed

The earlier Landscape Simulator was improved with priority on the conformity and stability of data, primarily during editing, storage, and re-reading in operations, with almost nothing done to speed up data transfer or display processing operations. But the evolution of computer hardware has rapidly improved display performance. In this case, if large scale city data is created in such detail that it takes a long time to display it, it is highly likely that when a member of the public tries to view it, the waiting time will psychologically discourage the person from continuing. Therefore, we, and worked, to prepare a special viewer to be used only for viewing and capable of high speed display.

(3) Town planning problems and discovering system challenges

We are examining the “town planning” problems unique to each of the 15 model sites (urban redevelopment: 6, land readjustment: 5, continuous grade-separated crossing 2, height restrictions: 1, urban recovery: 1), and the contents of the proposals and opinions that have been submitted, and have established a research committee that is now studying future improvement policies etc. Although we requested submissions of model projects from throughout Japan, most have come from south-western Japan. And the hypothetical members of the public that would be the object of communication using the system (its regional range) vary between the projects. At the same time, we are now preparing and modifying contents and discovering new challenges by clarifying (1) public benefits of a project and (2) interests of local landowners in the project, to (3) discuss future dreams that can be achieved anew through the success of the project.

6. Re-structuring process toward completion

During the application into the practical use in the field, functions are added upon requests of the users in many fields, including urban development, water resource management, dam site, bridge design, park etc. The system was also applied to international cooperation, including evaluation of Sea Level Rising at coastal cities, designing alternative future of housing complexes regarding the elimination of increasing energy consumption, recovery of large scale urban disasters etc. Those

activities resulted versions with different additional functions and languages.

Between 2008 and 09, efforts were made to re-organize those versions, and resulted the unified ver.2.09 that is precisely described in this report.

In short, this new version does not contain any new attractive/useful functions. But the most significant feature is the new architecture.

(1) Core part (sim.exe) is consist of basic functions that covers all the field of public works.

(2) Core part can dynamically change the language, and only one executable is enough to support any language.

(3) Core part can be translated to new language at text-base works (without programmers)

(4) Core part can dispatch external functions and plugin dll-s that are provided for specific needs of some field of public works

(5) The selected language is informed from the core part to the external functions and plugin dll-s, so that they can change the language in synchronic way.

In this restructuring process, major useful functions were collected to new core part, including various graphic features (shadow, stereo sight, rapid viewpoint move, etc.).

Also, specific functions available that are specific to some fields of construction were organized in the form of 5 plugin dll-s as mentioned above.

This restructuring will have the impact to the long term future development of the system.

(1) Improvement of the core part will pay attention of the “completeness”

(2) Innovative development will occur in the form of developing plugin dll-s.

(3) Transfer to the new language will be performed by the literal translators.

7. Conclusion: panorama

We are now at the stage where the release of systems and the data that has been prepared has started and opinions are being received from the general public. Additional features that are successfully developed through this process are being added as they are ready. Viewing three-dimensional contents and moving pictures etc. is still extremely troublesome for members of the general public, most of who are still connected by telephone lines. But broadband is spreading rapidly, so that solving the speed problem is only a question of time.

People are being urged to move to new industries from old industries where unemployment is high. In such conditions, those corporations that have received past orders to develop conventional software and prepare data for this project, plus young

newly established venture companies (3 companies), have handled important parts of this development work. Specifically, the preparation of the Apple version, development of the public release function based on the GIS data + VR technologies, and development of server functions based on ASP, or so-called "cloud computing".

The results of a survey of the latest technologies done before development started revealed those practical and potentially useful technologies that would improve three-dimensional display, stereoscopic display, and various service functions existed not only in the private sector, but also at universities and the National Institute of Advanced Industrial Science and Technology. But because many of these researchers and developers lack their own networks outside their institutions, and therefore, do not have the experience and channels necessary to release products, they did not go beyond obtaining patents and publishing papers after the trial manufacture and demonstration stages. They also lacked established procedures for selling technology. Because the Ministry of Land, Infrastructure, and Transport has a large number of facilities where it tests various kinds of intelligent technologies, it may be asked to actively discover and evaluate such technologies that are awaiting practical application, then purchase them and introduce them in the field to refine them.

In the long term, diachronic communication will also be important, which enables the communication of 3D data from the carpenters that constructed a house to another carpenters who will reform the house after 30 years, or to a real-estate dealer who will evaluate the house after 100 years.

Notes:

1) Japan Association for Building Research Promotion (tell: 03-3453-1281) is distributing the Building Research Document No. 96, Mature City Simulator 1.0 + Landscape Simulator 2.05 User's Manual (July 2000, accompanied by 2 CDs). Back numbers include, "Building Research Document No. 92 "MOC Version of the Scenery Simulator Personal Training Handbook" Nov. 1997) and Civil Engineering Research Document, "Landscape Simulator" etc. A Korean language version prepared through joint research with the Korean Rural Research Institute is also available.

2) URLs related to town planning communication systems can be accessed from http://sim.nilim.go.jp/.

Note 3) it has been announced and demonstrated at the following places.

・Twenty-first Century Future Exposition (Kobe: July 20 to September 2)

・Children's Science Festival 2001 (Tsukuba, October 7 to 8)

・Civil Engineering Day (Tsukuba, November 18)

・Ministry of Land, Infrastructure, and Transport, Technology Conference (Tokyo Shinagawa TOC, November 20 and 21)

・Civil Engineering Exhibit of Life and Technology (Tokushima, December, 7 and 8)

Example 1: Future image of Fujimi-city

Example 2: Present Topography and the City of Hiroshima Crossing Project

目　次

図版一覧・・(掲載頁)

リスト一覧・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・(掲載頁)

表一覧・・（掲載頁）

１．ヒストリー

はじめに

あるソフトウェアの機能や適用例を題材とした論文、紹介記事、技術資料等は多いが、開発過程そのものを扱ったものは少ない。少ない多くは、着手段階での構想・計画や、開発成果の社会的普及を目的とした機能追加歴等に関するものであり、様々の失敗経験を含む開発過程を遡って系統的に記録したヒストリーや、完成段階における諸問題を扱った論考は極めて少ない。

筆者は、15 年来、日中韓にまたがる十数名のプログラマと共に開発に関与してきた景観シミュレータの枝分かれバージョンを統合する作業を 2008-09 年度に行った。本章では、その間の開発の経緯と、整理統合及び完成に向けたチャレンジについて報告する。

１－１．国土交通省版・景観シミュレータ開発過程

バブル崩壊後の 1993～96 年度に実施された建設省総合技術開発プロジェクト「美しい景観の創造技術の開発」の１課題として、建築・都市・土木各分野共通のソフトとして建築研究所と土木研究所の共同で開発が着手され、1996 年からフリーウェアとして WEB 公開された、草分け的な景観検討のための三次元 CG システムである。最初の現場投入は 1996 年の、福岡市内の峰花台団地の建替（中層→高層）で、現場にマシンを持ち込んで事業対象者に自由な視点移動による景観シミュレーションを行った例としては国内初である。

開発に先立って、1992 年に、技術状況を把握するための予備調査を実施し、内外の最新技術とその動向を把握した。1992 年に、ロサンゼルス暴動に際して、復興市街地の三次元ＣＧが作成され、デモンストレーションが行われた。当時、約 500 万円のグラフィック・ワークステーションに、約 300 万円のソフトゥエアを乗せた環境の上で、三次元的に再現された市街地の中を自由に移動する作業環境は、従来の駒落としビデオなどにより作製された固定的なアニメーション（視点や移動経路を変更することはできず、設計内容自体も編集不可能）とは一線を画するものであった。将来のＰＣの高性能化が予想される中で、今後の景観評価・設計検討の手段として有効と考えられた。実際に、現在では１０万円未満のハードにフリーウェアを用いることで同等以上の作業環境が実現している。

開発開始当初(1993)の理念は①マルチプラットフォーム、②オープン・ソース、③フリーウェア、④オブジェクト指向であり、開発にあたっては、ユーザーに触れる外観から作り始め、各種機能を次第に奥の方に作り込んでいく順序とした。

VRML 登場以前の、初期のデータ形式と、基本的な操作（ヒューマン・インターフェース）を現在まで全く変えておらず、Windows VISTA に対応した約 46 万ステップの最新版においても、15 年前のデータとの互換を保っている。平成 13 年に、国総研に移管されてからの、ダウンロードサイトのアクセス数は、35,656 程度である。

（１）プロトタイプ段階(Ver.1.00, 1995)

グラフィック・ワークステーション(UNIX)を用いビューワとしての機能に限定した。暫定データ形式を定め、CAD で作成・変換した構造物三次元データをコンバートし、背景写真と合成する。「視点抽出」機能により背景写真を解析し、同じ視点から構造物の透視図を作り位置精度を追求した。構造物のカラー編集機能を用意した。

（２）モデリング機能の萌芽(Ver.2.03, 1997)

WindowsNT3.5、95 をプラットフォームに加えた。ファイル保存機能と、直方体、球、多角錐､型鋼､階段など様々の基本的な立体図形をパラメトリックに生成する外部関数(.exe)によりモデリング機能を提供した。外部関数はユーザーが追加可能とした。更に景観データベースから、樹木や点景等の基本的な部品を選択し配置できるようにした。全三次元で福岡市内の団地建替えの地元説明、三陸国道で高規格道路の内部検討に使用した。韓国農漁村研究院との共同研究を開始した。

（３）市街地自動生成(Ver.2.05, 1999)

都市計画条件を設定して市街地を自動生成する機能、商店等の写真からテクスチャ付き立体を作成する機能、ステレオ空中写真から地形＋市街地をデータベース化する手法(官民共同研究)など、モデリング機能を拡充し実用性を高めた。福島市の区画整理事務所や幕張駅南口の再開発事務所で地元説明に利用して頂きながら、バグ報告や改善要望に迅速に対応するように努めた。メモリリークを解消し、形状の生成・削除を繰り返すモデリングの持続可能性を高め安定化させた。

ビューワとして固定的なデータに対して視点移動の表示を行うだけであれば、外部ファイルを読み込んでメモリ上にデータを構築するだけで事足りる。しかし、モデリング作業の中で削除したデータが完全に処理されていないと､「ごみ」としてメモリ等を浪費する（メモリリーク）。構築に対応する除却の機能が正しく使われていない場合、あるいはそもそもそのような機能が存在しないような場合に対処する必要があった。プログラマが十分と考えて固定長配列で処理している部分が大きなデータにおいて破綻するような場合についても、動的にメモリブロックを割り当てる方法に修正する必要があった。長期間にわたり、都市計画条件に従って市街地の建物を更新することは、基幹部分に対するストレス・テストとしての意味をもった。

（４）データ WEB 配信(Ver.2.07,2001)

WEB サーバーから配信する三次元データを、WEB ブラウザと景観シミュレータを連携させて表示できるようにした。ユーザーが作成・編集した三次元データを意見と共に投稿し、審査員の判定の上公開するサーバー機能を開発した。15 ヶ所のモデル現場で、ステレオ空中写真から作成した現況市街地データの上にまちづくり計画案を乗せ、WEB 公開した。Windows ME、2000 に対応する中で、多くの潜在バグを発見し解決した。

（５）分野毎に目的を特化した機能追加(1998～2004)

ダムや国営公園等での応用に際して、影、高速表示、地形編集等の部分的機能拡張が行なわれた。現場に対応した枝分かれバージョンが生じた。

機能追加のニーズが生じた場合、その時点における最も安定したソースコードに、追加する方法が最も効率が高い。1997～99 年度に、建築研究所において再開発等の現場担当者による使用をサポートするために、機能追加を抑えて、安定性を追求してデバッグを進めた Ver.2.0X 系列が現在の基幹部分となっている。この時期に、並行して土木研究所において、同じ Ver.2.00 のコア部分に機能を豊富に追加した Ver.3.2、Ver.4.0 の系列が、高規格道路やダムなどの土木系現場における調査業務の中で作成された。2001 年度以降も、国総研において、更に公園などの景観検討のための機能が追加された。結果的に、枝分かれバージョンが生じることとなった。

（６）多言語版(2006)

1997 年に、日韓科学技術協力協定の中に、「景観シミュレーション技術の地域開発への応用」が採択され、建設省建築研究所と韓国農業基盤公社農漁村研究院の間で共同研究が開始された。その中でソフトウェアの韓国語への翻訳と、韓国側での開発成果の還元が行われた。この段階での翻訳移植は、ソースコードを翻訳する形で行った。同様の方法によりインドネシア語への翻訳移植も行った。しかし、この方法では、デバッグや機能追加を行う度に、その結果を反映させるために、現地でのソースコード修正と再ビルドを行う必要があった。

2006 年度に、日韓共同研究の中で試みた国際化を発展させ、言語に依存するメニュー、メッセージ、ヘルプ等を全て外部テキスト化し、同じ実行形式で複数言語を使用できる機構を開発した。これは、インドネシアでの、まちづくりワークショップに投入された。

多言語版においては、デバッグ結果などを直ちに各国に反映できるように、実行形式を言語に依存しない１本に取りまとめ、言語に依存するメニュー、ダイアログ、エラーメッセージ、ヘルプ等を全て外部テキスト化し、起動時に必要な言語を選択的・動的にロードする仕組みとした。これにより、新たな言語における使用も、プログラマや現地語での開発環境なしに、翻訳家によるテキストファイルの作成のみにより実現可能となった。これにより、言語に起因するバージョン枝分かれの問題が解決した。

（７）完成を目指した集約作業(2007～8)

機能が異なる各枝分かれバージョンのソースコードを再整理し、共通の基幹部分を一本化するとともに、分野固有の目的機能を、必要となった時点でロードするプラグイン DLL として、ソースコードにおいて分離・独立させ、選択的に追加できる機構を追加した。その上で、これまでの枝分かれバージョンから、地形編集、トンネル、園路、道路法面を、プラグイン DLL の実装例として分離独立させた。基幹部分に関しては、多言語版とした。

これと共に、プラグイン DLL による新たな機能開発や、新たな言語への翻訳移植、あるいは本システム（図形処理アルゴリスム等）を部品として利用し、システム開発を行なおうとする開発者・プログラマのための解説書を執筆した(本章２～１５章及び付録 A,B)。ソースコードに関しても、整理・整形した上で、本書に CD-ROM として添付した。

図１は、以上の経過を整理したものである。

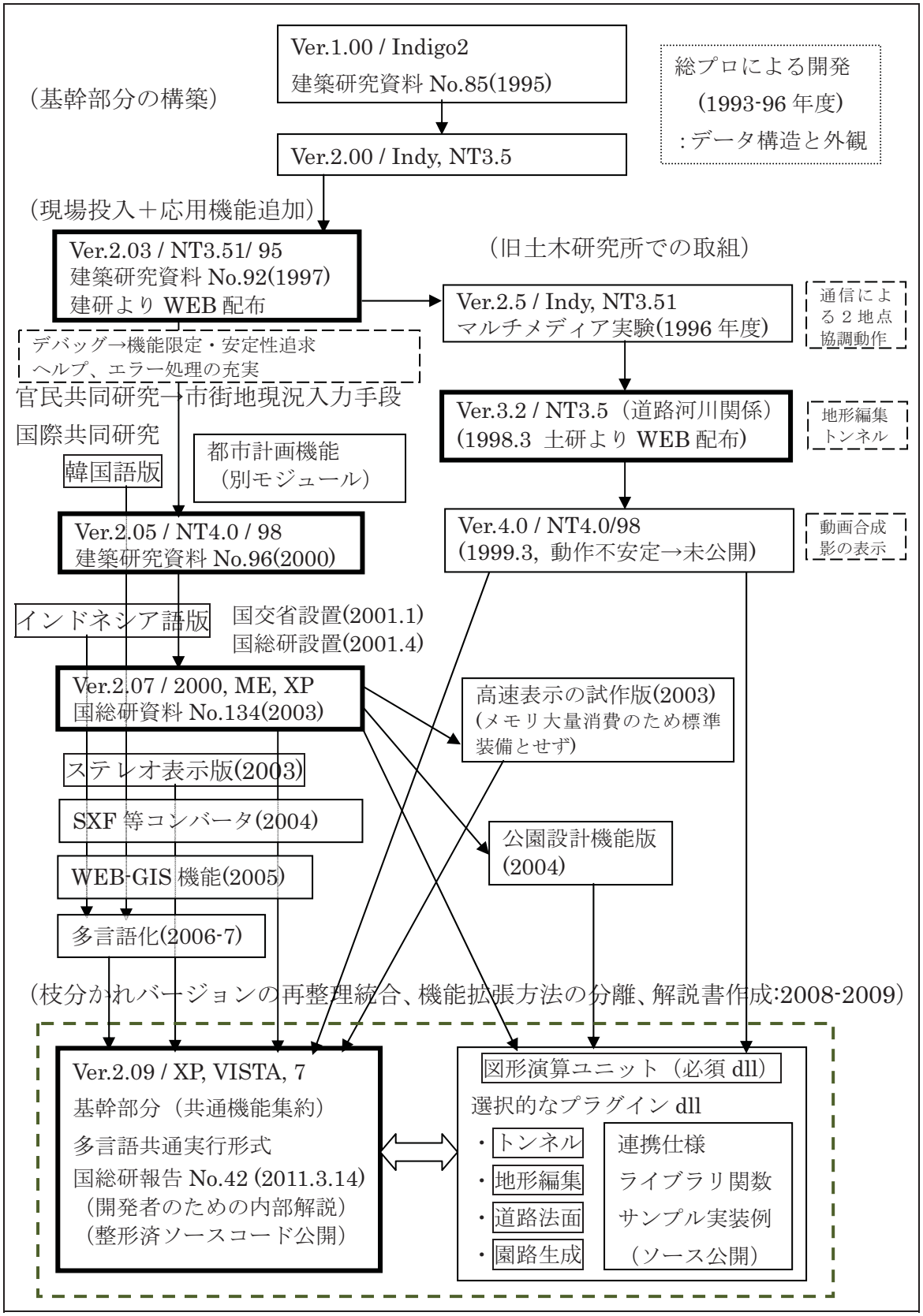


図１－１：景観シミュレータの開発経緯と統合化

１－２．開発における理念と実現過程

1993 年に開発に着手した当初の理念は、「マルチプラットフォーム」「オープン・ソース」「フリーウェア」「オブジェクト指向」である（文献 17 p.282-290）。

（1）マルチプラットフォーム

マルチプラットフォームの考え方は、当時実用的な速度で表示可能なマシンが、UNIX 系のグラフィック・ワークステーションであったこと、それに対して現場で PC の普及が進み、WindowsNT 系の OS にも三次元表示を行うための OpenGL ライブラリが実装可能と予告されていたことによる。実際に、Ver.2.03 以降は、実質的に PC ベースのアプリケーションとなった。しかし、マルチプラットフォームとするために、ソフトウェアの構成を、OS に依存しないデータ処理と、OS に依存するユーザー・インターフェースの部分を分離し、前者をライブラリ関数としたことは、後の Windows 系 OS の頻繁なバージョンアップに対する作業効率に貢献したと考えている。

（2）オープン・ソース

オープン・ソースとした消極的な理由は、限られた開発期間が終了した後に、デバッグやメンテナンスの責任を研究機関が継続的に負担することが困難であることが予想された点にある。非公開としたまま開発と予算措置が終了し、以後適切なサポート体制が組まれない場合、開発担当者の手元に開発成果が死蔵される危険性がある。このことについては、第 16 章で考察する。実行形式だけが公開されているようなシステムに関しては、以後のデバッグや機能追加は非常に困難である。

公開されるオープン・ソースとすることで、特に外注部分のソフトウェア（プログラマの署名が入る）の書法が引き締まったものとなった。但し、その後発展した Linux 等のように、一般のボランティア的な貢献による機能追加やデバッグ等は、これまで行っていない。しかしながら、三次元形状や画像ファイルの処理において、既に公開されていたオープン・ソースのプログラムは、開発担当者プログラマにより適宜引用され、必要に応じてデバッグされている。その中には、Ver.2.09 にむけた統合整理作業の中で、出所が必ずしも明らかにできなかった部分もあることは残念である。しかし、ソースコードの冒頭部分で、担当プログラマによるオリジナルのプログラムであるか否かを明記するようにした。

中国人プログラマに下請けされた一部に、ソースコードではなく、スタティック・ライブラリの形で納品されていた箇所があり、OS のバージョンアップに際して障害の原因となった例がある。結果的にソースコードを入手し解決したが、それまでは、ライブラリのバイナリファイルを修正する必要があった。

オープン・ソースとしたことにより、海外（韓国）との協力が可能となった。WEB 公開されたソースコードの取得・ダウンロードは外国からも可能である。とはいえ、国内予算で開発した知的財産を国際的に無償提供することはフェアではないと考え、協定を締結して、役割分担と相手方の開発成果の還元を協力の条件とした。実際に韓国で翻訳移植の作

業を共同で実施した所感では、使用する開発環境の違い（それらが含むバグ等）により生じる新たな障害等への対応が付随的に必要となり、単なる言語依存部分の置き換えでは作業は完了せず、ソースコード提供だけでは技術移転は容易ではないという印象を受けた。

Ver.2.07 以前のバージョンにおいて公開したソースコードには、デバッグのためのコンパイル・スイッチや注記、あるいはソースコードの一部を暫定的に削除するためのコメント・アウトなどが多数残されたままの、いわゆるビルドをそのままの形で WEB 公開する方法を採っていた。試行錯誤の記録が残されていることが、将来のデバッグの参考になる可能性もあるが、全般的には、ソースコードを読みにくくしており、このため、これを具体的に理解し改善しようとするプログラマのためには、必ずしも親切ではないと考え、本報告書と共に公開する Ver.2.09 に関しては、不要なコメントを削除した。その際に、手違いによるロジックの変化を防ぐために、リリース・モードでコンパイルした結果生成するオブジェクト・ファイルが、バイナリで変化していないことを確認しながらソースコードの整形作業（外注）を行った。

（3）フリーウェア

国立の研究所が開発したソフトウェアであり、開発に要する費用は国費であることから、開発成果を、メディア代以上の費用で有償販売するような体制は不適当と考えた。

建設省の出先で利用する社内ソフトウェアであれば、必ずしも自由公開する必要はない。しかし、都市計画や景観検討が、国の直轄事業のみならず地方公共団体や公団等においても行われることを考えると、フリーウェアとして扱うことが適当と考えた。また直轄事業の設計は多く民間コンサルタントにより行われているため、民間における使用を制限することは、実用上の支障となる可能性がある。このことから、ランタイム・ライセンスを必要とするようなライブラリを用いて開発手間を節約するような方法を避けた。

また、当時ソフトウェアを配信するために、パソコン通信や、インターネットによるダウンロードが利用可能となっていた。このようは配布方法に際しては、フリーウェアは適切な運用形態である。

このような方法においても、料金を徴収するためには別途技術的な手段を必要とするが、そのような仕組（EC）はまだ発展途上にあった。

ネットワーク上での動作環境を実現した 2001 年度時点(Ver.2.07)では、景観データベースに含まれる事例データや景観データベースのデータがかなりの分量に達していた。このため、その全てをダウンロードすることは、当時の回線速度(10Mbps 程度)では必ずしも実用的ではなかった。そこで、ビューワ（実行形式のみ）、コンパクト（景観構成要素データベースのみ）、フルセット（すべてのデータを含む）の 3 種類のセットアップを用意し、目的やネットワーク接続環境に応じてダウンロードを選択できるようにすると共に、ネットワークに接続した利用環境において、未取得のデータにアクセスしようとした場合には、その都度、国土技術政策総合研究所で運用するサーバーから個別に追加ダウンロードするような構成・機構とした。

その後、ネットワークを対象とした悪意のある攻撃などが増加し、それに伴ってOSの側でもセキュリティの確保が進んだため、上記のような相互信頼に基づく便利な利用環境の導入にはかなりの慎重さが求められるような環境になった。

（4）オブジェクト指向

オブジェクト指向は、開発開始当時の思潮であった。具体的にはC++言語において、クラスを定義し、その下にデータ構造とともに各種オペレーション（関数）もパッケージ化するという手法である。但し、当時まだC++の開発環境は新しく、コンパイラ等のバグの懸念も払拭されていなかったことから、オブジェクト指向的な考え方を採りつつも、景観シミュレータにおける基本的なデータ形式であるグループや面などのデータに関しては、慣習的なC言語で、データ構造（構造体定義）とライブラリ関数の作成を一体的に進めた。

具体的なモデリング操作の多くは、まず画面上で処理対象を選択してから処理内容を選択指定する、という作業手順となっており、オブジェクト指向的な考え方に基づくものとなっている。

データ構造においては、「グループ」を基本的な単位として、地物を構造的に記述することとした。具体的に表示される面やそれを定義する頂点座標の「意味あるまとまり」を、一つのグループとして取りまとめる。更に、グループとグループの間にリンクを定義する。これにより、たとえば、親グループとして「テーブル」は、ひとつの「板」子グループと４の「脚」子グループにより構成することができる。データの冗長性を避けコンパクトな記述を可能とするために、同形ならば「脚」グループは一つだけ定義し、これと親グループである「テーブル」の間に４のリンクを定義することもできる。道に沿って多数配列する街灯や、公園に植樹される樹木などは、一つの子グループと多数のリンクにより実現することができる。言いかえると画面に見えているオブジェクトは、「リンク」に対応する。

このグループとして、固定的なデータのみならず、パラメトリックな部品を用意した。形状が固定的なグループであっても、リンクに付属するマトリックスによりXYZそれぞれに独立した倍率をかけることができ、サイズを変更することができる。しかし、土木建築施設を構成する要素の中には、拡大縮小だけでは表現しきれない変形を示す要素がある。例えば、時計は時刻をパラメータとして長針・短針・秒針を変形させるが、その変形は単なる拡大縮小ではない。従って、時刻をパラメータとして、３本の針の配置をコントロールできるような部品が定義できれば、コンパクトに形状を記述することができる。ドアや障子などの可動部品はその類である。また、型鋼のように発注に際していくつかのパラメータを指定することにより最終的な形状が決まる要素も存在している。景観シミュレータにおいては、これらを、外部ファイルの一種として扱い（３－３）、実装においてはユーザーが作成して追加することができる外部関数として実現した（第5章）。

（5）データ形式の一貫性と進化

景観シミュレータの開発当初には、三次元データを記述する標準的な形式がまだ存在し

ていなかったことから、独自のデータ形式（内部メモリ上のデータ構造と、外部ファイル形式）を定め、これを用いてシステム開発を進めるとともに、汎用部品や適用事例の蓄積を進めてきた。このデータ形式に関しては、現在まで全く変更を加えていない。

一方、その後 GIS や CAD、仮想現実等における技術が急速に発展する中で、各種データ形式が定義され、一部は ISO によりオーソライズされたこともあるが、以後も様々な形式が提示されている。これらの生成流転に対処して、主要なデータ形式に関する入出力の機能（ファイル・コンバータ）を作成してきた。

景観シミュレータで定めたデータ形式は、地物固有の属性と、具体的な表示のために必要な太陽等の光源配置や、重要な視点などの、個々の情況に関する属性を別ファイルに分離した点に特徴がある。三次元データの WEB 配信を目的とした VRML 言語においては、これらが一つのファイル中に混在している。

地物の表面仕上げ属性に関しては、色彩や鏡面反射率、テクスチャ等を固定的に指定する方法に加えて、表面仕上げ等の表示処理が将来高度化することに備えて、より抽象化した「マテリアル」（木材、鋼材、アスファルト、コンクリート等）を地物に対して間接的に指定しておき、マテリアル毎の具体的な表示方法（テクスチャ、色彩等）を別途詳細指定する方法を用意した。具体的には、材料毎の発光体としての輝度（EMISSION）や、平滑面の鏡面反射率（SPECULAR）等の属性をマテリアル・ファイルの中で定義している。これにより、地物属性を記述する LSS-G ファイル自体の形式の拡張や変更を回避した。

将来、具体的なマテリアルに即して更に高度なグラフィックス表示等が利用可能となった場合、同じ LSS-G ファイルを用いて、それらを利用する可能性が開かれている。

表面から構成される地物の単位である「グループ」を重要な単位として定め、更に一つのグループを、子グループの配置により記述し、その相対的な位置・回転を記述できる。これにより、例えば柱梁などの部材→住宅等の単体→敷地→地区→地域といった段階的な構成を、局地的な座標系と上位の座標系へのリンクとして表現でき、CAD のレイヤー等よりも複雑なデータ構造の記述を可能としている（２－１１参照）。

グループには文字列として任意個数の属性を設定することができ、これまで地面、樹木、住宅などを識別するために使用している。未定義の属性が定義・追加されても、従来のシステムにおいては表示や操作に影響することなく、ファイルの保存・読み込みではその属性は保持・継承される。プラグイン DLL による新機能の開発などにおいて今後積極的に活用される可能性がある（３－３参照）。

形状の大枠を定めておき、形状の一部のみを数値指定する、パラメトリックな部品を外部関数として、標準部品に対して追加が可能な仕組みを用意した。これにより、LSS-G ファイルにおいては、定形的な構成要素をコンパクトに記述することができる。外部関数は拡張可能である。

以上のような当初の仕組みにより、当初の仕様を変更することなく、その後公開された様々なデータ形式に対しても柔軟に対応することができ、コンバータを追加することにより、データの授受を行ってきた。

Ver.2.09 に向けた整理・統合作業の中で、公開している外部ファイル仕様と、内部の処理ロジックの照合を行い、仕様として定められていながら完全には実装されてかった項目（実際には殆ど利用されていない）に関しても追加で実装を行った。とりわけ、LSS-S 形式における EFFECT の利用に関しては、いくつかの有用な機能を実装している（３－２参照)。

１－３．改良とバージョンの枝分かれ

（1）機能拡張と安定性追求の調和

大規模な機能拡張は、プログラミング業務の役務発注として実施されてきた。新たに追加された機能は、納品段階においては、多くの場合比較的小さなサンプルデータによるテストを経たのみであり、実際の設計対象物等のデータによる検証が不十分である。現場での使用に際して生じるバグは、既に納品を終えた開発チームとは空間的・時間的に隔たっている。最もデバッグが進んだ時期は、1997-8 年であるが、この間のデバッグ、エラー対策、ヘルプの改良等は殆ど研究所職員による直営で行われた。その頃、Windows 版に対応した Ver.2.00 をベースとして、実際の現場でプレゼンテーションを行うと共に Ver.2.05 に向けたデバッグ作業を進める中で、バグの発見を容易にするためにまず、コンパイル警告等を解消した。次に、システム終了時点で生じていた大量のメモリリークを解消するために、不足・欠落していた、使用済みメモリブロックの解放処理を追加した。これにより、バグの探索が容易化された。更に、福島市役所や都市基盤整備公団が実際の土地区画整理事業や市街地再開発事業における現場説明に使用開始したため、現場での実データを用いた操作により生じる障害への対応を直営で行った。この間、デバッグの対象としたビルドに関しては機能追加を行わず、デバッグを中心に作業を進めた。

エラーメッセージに関しても、初期のバージョン 2.03 までは、データの異常や不整合が検出されても可能な限り処理を続行する、という考え方を採っていたが、デバッグの対象としたビルドにおいては、なるべく早期に些細な異常であってもメッセージや警告を表示するようにエラーメッセージを大幅に増補し、現場からのクレームに対して、原因の特定がより容易になるように努めた。

現場での使用と平行して行った、研究所における非常勤職員による景観検討データや景観データベースのコンテンツの拡充に、外部の CAD ではなく開発中のシステム自体を使用したことも、バグの発見に大いに資することとなった。この体制は、Ver.2.07 にいたるまで続けられ、とりわけバグに対して厳しい OS であった Windows ME への移行に際して、更に多くの潜在バグの発見と修正を行うことができた。

同一時期に外注により開発による機能追加が行われたバージョンの枝分かれが生じ Ver.2.5,3.2,4.0 等に関しては、重大なバグに関しては開発担当者に適宜情報提供を行ったものの、デバッグの結果が必ずしも十分に反映されず、バージョンの枝分かれが生じる結果となった。

一般に、複雑なデータ処理が求められる三次元 CG 製品のデモにおいては、高速視点移動などが重視され、性能の追求に関心が払われる結果、内部の一貫性の確保やバグの除去

への時間投入が後手になる状況が生じやすい。民間の CAD 製品と並んで現場説明を行った際に、現場の作業環境で CAD 製品の方が頻繁にシステムダウンを起こすような状況を体験したこともある。とりわけ、発注者やユーザーにとっては、システムの信頼性や安定性の問題は、実務的に本格的に使用する中で初めて認識される価値であるため、外注における納品段階で簡単なサンプルデータを用いたテストとデモにより評価することは難しく、仮に外注を中心とする開発体制であっても、ある程度のインハウスによる継続的サポートは不可欠であった。

長期的に見ると、デバッグを進める時期は、機能追加が停止している期間である方が、全体的な開発効率が高い、と言うことができる。

（2）ハード、OS と開発環境の変化への適応と機能的な進化との歩調合わせの問題

OS と開発環境の発達は、ゼロからのアプリケーション開発を容易化する。従前であれば、開発プロジェクトの中で独自のプログラムを作成しなければならなかった機能の内、各種アプリケーションに共通に利用されるような種類の機能は、OS の機能の一部や、開発環境と共に提供されるサンプル・コードを利用することにより、より迅速・簡便に利用できるようになる。

しかし、一通り完成し、実務に投入されているアプリケーションがそのまま新しい OS で正しく動作するとは限らない。また、機能追加のために、ソースコードをバージョンアップされた開発環境に載せ替えた時に、開発済みの機能がそのまま自動的に正常に動作するとは限らない。これらの環境変化に伴い、全く同じ機能のレベルを維持するだけのために、コストと時間を投入しなければならないことは、メンテナンス上の負担となる。

一方、異なる OS への移行に伴い、潜在バグの顕在化も起こる。とりわけ、2001 年度のネットワーク化に向けた開発において、ユーザー側の OS と想定してテスト・デバッグを行った Windows ME は、既存の多くのソフトウェアで障害が生じることにより、不評・短命な OS であったが、景観シミュレータの場合、OS の欠陥に起因する障害ではなく、それまでの Windows98 上で正常に動作している潜在バグが顕在化した場合が殆どであった。ポインタ変数の不適切な使用や、配列の限界を超えたアクセス、オート変数へのポインタを使い続けることなどによるバグは、それ以前の OS においては、原因発生から暫く操作が行われた後に、顕在化する場合が多く、原因と結果の関係が極めにくい。これに対して、原因発生後直ちに障害となって現れる OS は、むしろ原因の特定を容易化した。このため、その次に長期的にユーザー側の環境となった Windows XP への移行に際しては、殆ど修正を必要としなかった。Windows VISTA においては、重要な修正は２ヶ所に留まったが、三次元グラフィックスの表示に直接関係する基本的な部分であり、従来の実行形式では全く使用不可能な状況であった（７－１参照）。

Windows 系のサーバー上の機能（WEB-GIS 等）に関しても、OS の 2000Server から 2003Server への移行に際して、「設定されない機能は許可」から「設定されない機能は不許可」に変更がなされ、不正侵入等の原因となる不要なサーバー機能が未設定のまま動作

しているような環境が改善された。これに伴い、開発したソフトウェアをセットアップする際の設定作業がより厳格化した（サーバー上で動作するソフトウェアに関しては、別稿に譲ることとし、本書では扱っていない)。

開発環境として C、C++言語を使用した。Ver.1.00 は UNIX 系の C コンパイラを用いたが、Ver.2.00 から、Windows 系の Visual C++を開発環境として併用し、以後こちらが開発の中心的なプラットフォームとなった。開発環境のバージョンは、4.0→6.0→8.0 と推移してきた。2008～9 年度に行った Ver.2.09 に向けた統合作業において開発環境とした 8.0 (VS2005)への移行に際しては、２バイト系の文字列、セキュリティ・ホールとなりうる原始的な C の入出力関数や文字列処理関数等の使用に対する警告が大幅に増補された。これらに伴い、ソースコード全般に対する修正作業が必要となった。

更に、Windows 画面への描画表示に関連する基本的な部分に関しても、OS の変更に伴う仕様変更があり、しばしばソースコードの修正と、最新の OS のみならず旧来の OS 環境上でのテスト・デバッグによる検証を必要とした。

表示速度を向上するためには、様々のノウハウや工夫がある。例えば、バイナリーデータ形式の追加や、OpenGL によるディスプレイ・リストの使用などがある。しかしながら、これらを採用せずとも、開発初期から現在に至るまでの間に、同じ実行形式とデータの組み合わせであっても表示速度は劇的に向上してきた。その最大の要因はコンピュータのハードウェアの速度向上、とりわけ OpenGL 描画処理をハードウェアにより実現する表示部分の機能である。初めて現場に投入した時点で採用した Intergraph 社の TDZ マシンは、WindowsNT3.5 を OS として使用するマシンであったが、ハードウェアの大きな部分を専用グラフィック表示回路が占めていた。その後、CPU 性能に加えて、ゲーム・ソフトの普及に対応した GeForce シリーズ等の、グラフィックス・カードの使用が一般化した。最近では、このような機能が標準でマザーボードに搭載されるようになった。

ディスプレイ・リストを使用する方法は、大容量メモリを必要とするため、2003 年頃に枝分かれバージョンでテストを行ったのみであったが、Ver.2.09 においては大容量メモリの普及状況を考慮して、選択的に使用できるようにした（７－６参照)。

初期から現在までの間に投入した速度向上方法の内、特に有効であったのは、

・ファイル・ロード（IP ライブラリ）におけるシンボル・テーブル検索処理の改善
・ファイル保存時の最適化（同一シンボル名の再利用等）
・地形や周辺市街地のあるデータ表示における三角形と四角形の連続処理

である。

（３）機能追加に際して問題となった事項

個々に行われてきた機能追加に関して、以下のような共通の問題点が指摘できる。

① オペレーションの持続可能性

単純なビューワとしての、言い換えると地物を記述するデータを読み込んで様々な視点から表示するだけの景観シミュレーションであれば、１回のファイル読み込みに伴うデー

タ構築ができれば十分である。ロードした三次元データを目視確認した後、システムを終了すれば、OS により不要となったリソースは自動的に解放される。しかし、モデリング作業を行う場合、起動から終了までの間に数十〜数百回の構築と除却が行われる。更に市街地生成等のシミュレーションを行うためには数万回以上の構築と除却に耐える安定性が求められる。このためには、内部データ構造に対して、構築機能に対応した除却機能や原状復原機能（各種関数）を「可逆的」に作成され、各オペレーションに際してそれらが漏れなく実装されている必要がある。

② 想定外のデータに対する安定性

例えば固定長配列の長さを、プログラマ（必ずしもシミュレーションを行う対象の専門家であるとは限らない）が、「この程度確保しておけば十分」と判断して適当に設定した部分が、データが高度化した後にオーバーフローを起こす。システムのリソース（メモリ容量など）が許す限り、無制限の長さの配列を確保できるようにコーディングするためには、一定の手間が必要となる。まず、コーディングが単純な固定長配列を用いてアルゴリスムの実現を確認し、次に、想定外の大きなデータに備えてメモリブロックの取得と解放処理に書き換える、という手順で作業を進めることが能率的である。メモリブロックを使用する場合、配列長の変更に伴うメモリブロックの再取得(realloc)の処理は一般に処理時間が長いため、配列の長さの変更にある階段を設け、配列の処理がある幅に達した時点で再取得するようにコーディングするが、その幅をどの程度に設定するのが最適かは経験的にしかわからない。

納期が限られた外注の中でこの第二段階をクリアする条件を確保することは難しい。外見で判定できないため、検収段階でソースコードの検査を行い、さらに現場投入後のメンテナンスの中で適宜調整する必要がある。

③ＯＳと開発環境の変化に対する安定性

未来の OS が予見できないため、絶対的な解決はない。同一シリーズのソースコードとそれに基づく実行形式を複数時期の OS 環境上で動作するようにコーディングする努力を、数次にわたる OS の更新にまたがって続ける以外にない。便宜的に OS の種類を認識して条件分岐するようなコーディングは避けるべきである。

機能追加を伴うバージョンアップを外注しようとする場合、ゼロから作り直すことがしばしば提案される。確かに、他者がコーディングしたプログラムを理解した上で、これに機能を追加することは、ゼロから独自に作成するのとは別の努力を必要とする。景観シミュレーション・システムの場合のように、多くのプログラマによるソースコード作品と作風が共存しているような場合には、最初の読み込みに時間を要するであろう。

しかし、複数の OS や開発環境を越えた安定性を獲得するためには、そのような実績を積み上げるという長い時間が必要である。ゼロからの再開発は、過去に様々な環境で様々なデータを処理してきた、という経験もまたゼロから繰り返す出発点に再び立つ、ということを意味する。

１－４．バージョンの再統合と今後の展望

（1）バージョンの再統合

2008～9 年度に、枝分かれバージョンのソースコードを再確認すると共に、以下のような作業を行った。

①枝分かれバージョンに含まれる共通性の高い基本機能に関しては、基幹部分に追加しデバッグすることにより、有効活用を図った。これには以下のものが含まれる：

・ステレオ表示機能
・影の表示機能
・高速表示機能
・環境設定機能

②枝分かれバージョンに含まれている分野別のニーズに対応した機能については、プラグイン DLL の形で独立させた。

・地形編集（従来、水際形状編集機能として開発されていたもの）
・園路生成
・トンネル

③Ver.2.07 に含まれていた機能の内、改良の余地がある機能に関して、個別にデバッグを進めることは時間を要するため、プラグイン DLL として独立させ、個別のデバッグを将来の課題とした。

・道路法面生成

④多言語機能に関しては、外部関数及びプラグイン DLL も協調動作するように機能拡大を行うと共に、その実装方法をサンプルと共に解説し、サンプルとして配布するこれらのソースコードに実装した。

⑤基幹部分における多言語機能の部分は、コンパイル・スイッチを残しその所在を明示すると共に、コンパイル・スイッチを有効化した（#ifdef MULTI_LANG～#endif）。

⑤将来ニーズの変化により基幹部分に活用可能と考えられる共通機能で、Ver.2.09 に使用しないものに関しては、削除せずにソースコード中に残し、コンパイル・スイッチを用いて無効化した。

・バックアップ・アンドゥ系の機能の内、ユーザー操作をコード化して記録する機能
（#ifdef RDH_NONE～#endif）

（2）機能拡張の方法の明確化と技術資料の整備

短期的に新たなニーズに応えるためには、ソースコードの基幹部分に機能拡張を行う方法が効率的であるが、この方法では無数の枝分かれバージョンが発生する。モデリングのために初期から提供した「外部関数」は、ユーザーが自由に形状生成機能を拡張できるが、現在メモリ上の構築されている地物を削除・編集することができなかった。

2008 年度に導入したプラグイン DLL による方法では、既存の地物に対する編集操作を可能とし、基幹部分の基本的な編集機能と同様にあらゆる地物やシステム状態に対する操

作が可能である。そのために、DLL 側から基幹部分のライブラリ関数を利用できる機構とした。

長期的には、開発に関与した人材が入れ替わった後にも、継続的に機能拡張が行える状態を実現することが必要である。このために、本書の中で技術資料を提供した。

（3）記録保存と公開

過去のトラブル対策が個別記録に残されていると、後日、別のトラブルに対処する場合の参考資料となる。景観シミュレータの場合には、1997 年頃から、バグ・レポート様式を用意し 1 枚紙のレポートに問題と対処の記録を残した。2002 年以降は、情報公開のためのサーバー上にバグ・レポートのコーナー(sim2.nilim.go.jp/AprilFool)を設け、電子的な投稿・記録を行っている。機能改善に関する提案と対処結果についても、同じように処理している。

（4）今後の展望

地理空間情報活用推進基本法(ＮＳＤＩ法)が目指すように、情報化された国土が形成されつつある。１～４次元電子データとしてパーツが蓄積されるとともに、全体マップへの統合が進みつつある。

国土交通省においても、既に運用段階にある電子納品（CALS / EC）において、三次元データが検討され始めた（H20-）。従来は、三次元データ処理は主に機械化施工をターゲットに研究・開発されてきたが、設計から竣工後の維持管理まで利用するとなれば、業務形態を大きく変えることとなる（文献 36)。

景観法はじめ、分野を超えた有形資産を地域の中で全体的に考える機運が生じてきた。低炭素社会等の概念は、可視的な地物を、美的な意味での景観よりも広い概念であるランドスケープとして、地物の背後にある不可視属性の認識を要求している。

①開発の主体

現在とりまとめている構成においては、プラグイン部分を開発することにより、大幅な機能追加を行うことが可能となっている。その主体は任意である。また、運用として、プラグインに関してはオープン・ソースやフリーウェアではない開発方法も可能と考えられる。

②方法

プラグイン部分には、各種機能が追加可能である。現在までの実装例は地形編集等に関係したモデリング目的のものであるが、アニメーション等の表示系の機能も可能である。また、必要であれば、基幹部分よりも巨大なモジュールを開発することも可能である。

一般的に、ソフトウェア開発においては、ゼロベースの開発よりも、ビルドが通り、動作が確認できる既存ソースコードを出発点として機能を作りこんで行く方法が能率的である。プラグイン DLL のサンプルを出発点として開発を進めていくことは、枝分かれの問題を生じることなくローコスト・短期間で開発を進める方法を提供している。

③用途目的

景観をキーワードとする場合、見ることができる有形地物全体の、行政分野を超えた調和を図ることが当面の課題である。たとえ、美しい構造物が建設されても、その周辺にある居住地や里山・田畑等の風景が荒廃すれば、良好な景観は維持できない。このことは、単に風景のレベルにとどまらず、水の流れ、土砂の流出などを通じて、地域の安全性にも影響を及ぼす。更には森林の維持管理の適否が、地球環境にも影響を及ぼしている。

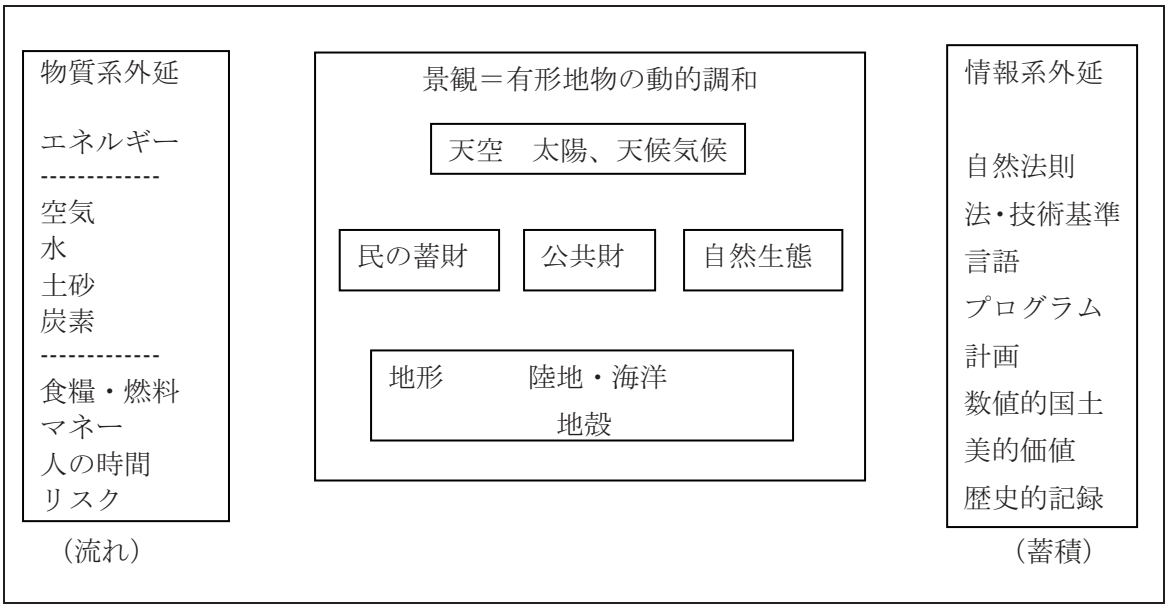


図１－２：景観シミュレーションの対象とその外延

このような事象を広く扱うためには、可視的な地物を扱うのみならず、その背後にある不可視な属性を対象とした評価や数値シミュレーションが必要となる。幸い、国土に関する数値情報やインフラ、市街地等に関する利用可能なデータは標準化され利用可能な形で蓄積され始めており、利活用の可能性が開かれつつある。その一つの方向は、鮮度の高いリアルタイム情報に基づく行政サービスやビジネスモデルであり、もう一つの方向は、歴史的視野に立った、地域の長期的な将来像に関するコンセンサス形成である。

このような目的のためのソフトウェア開発には、それが実現可能なものである限り、今回統合作業を行った結果として提供する Ver.2.09 の基幹部分を、変更することなく、プラグインを開発することにより応用できると考えている。それは、基幹部分よりはるかに大きな規模のものであっても良い。

このような目的のためのソフトウェア開発には、それが実現可能なものである限り、今回統合作業を行った結果として提供する Ver.2.09 の基幹部分を、変更することなく、プラグインを開発することにより応用できると考えている。それは、基幹部分よりはるかに大きな規模のものであっても良い。

２．景観シミュレーションの幾何学的基礎とデータ構造

２－１．三次元モデル

景観を任意の視点から検討するためには、検討対象である地物（地形や構造物など）が三次元空間の図形によって表現されている必要がある。景観として見るためには、地物の各要素の内部に関する情報は必ずしも必要ではなく、表面の形状と光学的性質（色彩、反射率、輝度、テクスチャ（色柄）など）が記述されていればよい。モニタ画面やプリンターに出力する画像（透視図など）は二次元画像である。この画像を得るためには、光源と物体と視点の位置関係から射影変換を行い、スクリーン上の画像を計算する。この計算処理はライブラリ化されており、景観シミュレータにおいては、OpenGL のライブラリを使用している。このライブラリは、Windows 系の OS には、標準で提供されている(OpenGL32.dll, GLU32.dll)。

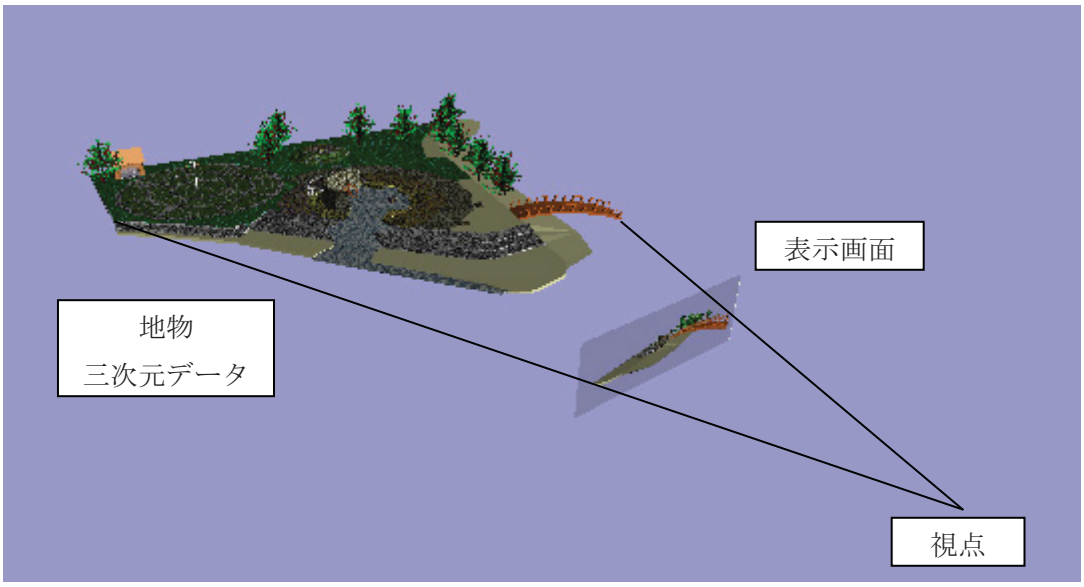


図２－１：射影変換（地物三次元データ・視点・表示画面）

景観シミュレーションを実用的に実施するためには、景観を構成する地物を表面の集合体として表現するデータ構造、それを外部ファイルとして入出力するファイル形式、同じ地物に関して表示の条件となる視点設定や光源設定、データの構築と削除などに関する各種処理機能が OpenGL による表示の一つ上の階層を構成する基盤的なシステムの要件として必要となる。これらの基本的機能の多くは、OS に依存しないライブラリとして用意した。国土交通省版・景観シミュレーション・システムの特徴の一つは、地物に固有の属性（形状、表面の光学的特性など）と、表示段階で必要となる地物以外の条件（視点設定、光源、時間）を、データ構造、および外部ファイル形式において明確に分離している点にある。

三次元の立体である地物の単位（地面や構造物など）を、表面の形状として表現する方法は、サーフェス・モデルと呼ばれる。

曲面を表現する方法としては、適切な多角形に分割して表現する方法を用いている。数

学的には、曲面を有する幾何学図形（球や円柱など）を関数とパラメータで表現する方法が可能であり、景観シミュレータにおいても、主にデータを軽くするために用いているが、表示のために OpenGL ライブラリを使用するため、出力段階では、物体の表面を多角形（ポリゴン）に分割している。そのような場合における分割数は、個別のデータには記述せず、環境設定で設定し、形状生成段階で自動的に処理している。

立体を構成する多角形を、多数の小さな三角形に分割するような方法と、ひとつの平面図形については、可能な限りひとつの多角形で表現しようとする方法がありうる。構造物等を扱うことの多い本システムにおいては、基本的には後者の方法を採用しているが、その中で前者の方法に基づいて作成されたデータを扱うことも可能である。

三次元空間内の多角形の幾何学的形状と光学的性質を記述するデータ構造（三次元ポリゴン）について、以下に説明する。

２－２．三次元ポリゴン

三次元ポリゴンとは、実空間、より正確には実空間をモデリングするためのユークリッド幾何学における三次元空間（以下、簡単に「実空間」と記す）に配置された、少なくともｎ個（≧３）の頂点を有する多角形を表現するために、コンピュータ上で座標点の配列として表現されたデータである。幾何学的には、頂点の数に等しい数の有限の長さの辺を有している。

ｎ頂点の三次元ポリゴンは、コンピュータ上のデータとしては、頂点を XYZ の座標値として表現したものを要素とする、長さｎの配列として表現される。景観シミュレータにおいては、以下の d3Face 構造体によって表現している（d3dml.h）。

リスト２－１：d3Face 構造体による面の定義

```
struct _d3Face {
/*private:*/
        d3Vertex *vl;           頂点リスト（配列）
        d3Face *next;           次の面へのポインタ
/*public:*/
        unsigned long va_f;     面の属性フラグ（法線、カラー、テクスチャ）
        unsigned long va_v;     頂点の属性フラグ（法線、カラー、テクスチャ）
        int vnum;               頂点数
        int material;           マテリアルID
        int texture;             テクスチャID
        float n[3];             法線ベクトル
        float c[4];             カラー
        int display;
        int shape;              形状種別（面、凹ポリゴン、折れ線、異常データ）
};
```

一つの d3Face 構造体（メモリ・ブロック、長さ：64 バイト）毎に、長さが vnum の頂点列(d3Vertex*)が定義される。個々の頂点は、三次元座標値を基本的な情報として持っているが、この他に、必要であれば、頂点毎の色彩や、テクスチャ座標（図柄の位置合わせ

に関する情報)、法線の情報をもつ。座標以外にどのような情報が登録されているかに関して、ポリゴンの va_v というフラグ、および頂点毎の va というフラグに記述している。法線や色彩に関する値が設定されていても、フラグが OFF となっていれば、表示には使用しない。

リスト２－２：d3Vertex 構造体による頂点の定義

```
struct _d3Vertex {
/*private:*/
/*public:*/
        unsigned long va;           頂点座標以外の属性の有無を示すフラグ
        double v[3];                頂点座標
        float n[3];                 頂点の法線ベクトル
        float t[2];                 頂点のテクスチャ座標
        float c[4];                 頂点のカラー値
};

注：va に設定する頂点毎のフラグ
#define D3_VA_NORMAL      0x0001      法線
#define D3_VA_COLOR       0x0002      色彩
#define D3_VA_TEXTURE     0x0004      テクスチャ
#define D3_VA_VIRTUAL     0x0008      仮想線の起点
```

内部のデータ、及び外部ファイルとして保存されるデータにおける座標値を記述するための座標軸の設定は、景観シミュレーションが対象とする空間尺度においては、慣習的に東を X 軸、北を Y 軸、上を Z 軸としている（CAD システムの慣習。コンピュータグラフィックス系のシステムでは、カメラが水平向きの場合に、レンズの軸を Z 軸とし、上を Y 軸とする場合もある)。

実空間に存在する多角形であれば、表側と裏側を定義することができる。景観シミュレータにおいては、上記の頂点列が反時計回りに見える側が表側である。

しかし、コンピュータ上では、現実的な図形としては許されないようなデータ（三次元ポリゴン）も作成が可能である。例えば、４の頂点を有するが、そのすべてが同じ座標値となっているようなデータである。従って、三次元ポリゴン・データとして多角形の図形処理を行うためにはいくつかの制約条件や前提条件が必要であり、形状生成や形状演算の処理結果については、適切な異常チェックを行わなければならない。以下、いくつかの例について示す。

（１） 頂点の数が２以下の面

面として描画が不可能である。点または線分に変換する必要がある（景観シミュレータでは、線のデータは表示可能であるが、点のデータは扱っていない）

（２） 多重頂点または、長さゼロの辺

同じ座標値を有する点が２以上連続する場合、見かけ上頂点の数が少ない多角形として表示される。演算処理によっては、不具合を起こす場合がありうることから、冗長な頂点は削除する必要がある。

（３） 連続する一直線上の辺

連続する辺が一直線上に連なる場合、中間の頂点は図形的に意味がない。但し、一直線上を折り返すような幅のない「ヒゲ」状の図形であって、面の内側に向かうものは、図形演算過程で生じる切込や切れ目を表現する準正常な面として許容している（２－６）。

（４） 辺が自己交差する図形

ひとつの図形を構成する辺どうしが交わり、８の字のようなねじれた図形となっている場合、現実にはありえない図形となる。

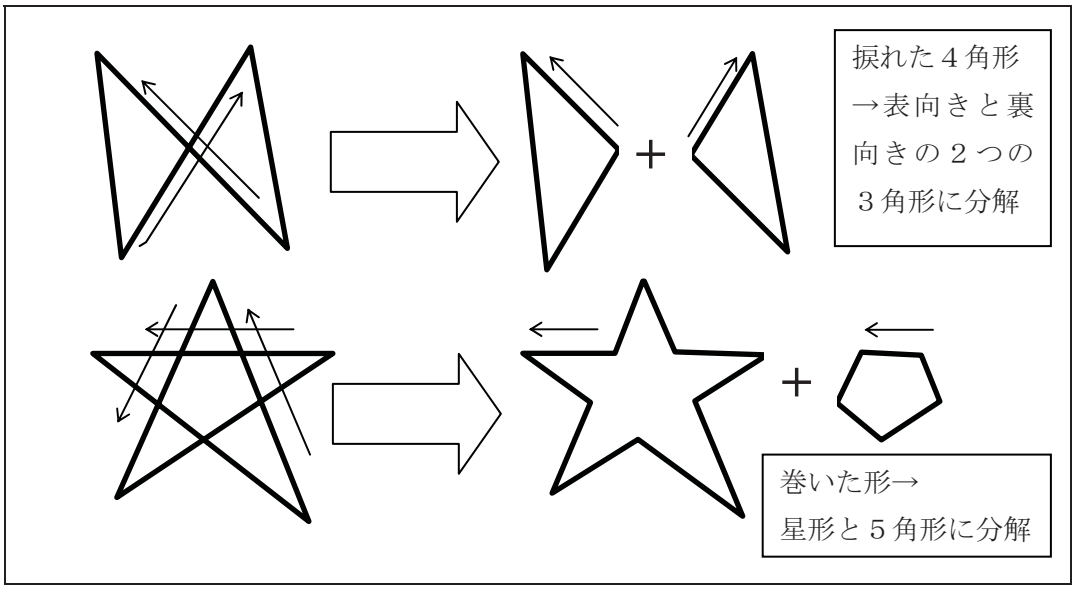


図２－２　辺が自己交差する図形と、正常な図形への分割

図形演算の結果このような図形が発生した場合には、交差する点に新たな頂点を追加した上で、表向きの部分、裏向きの部分、多重の部分を別の図形に分解処理している。

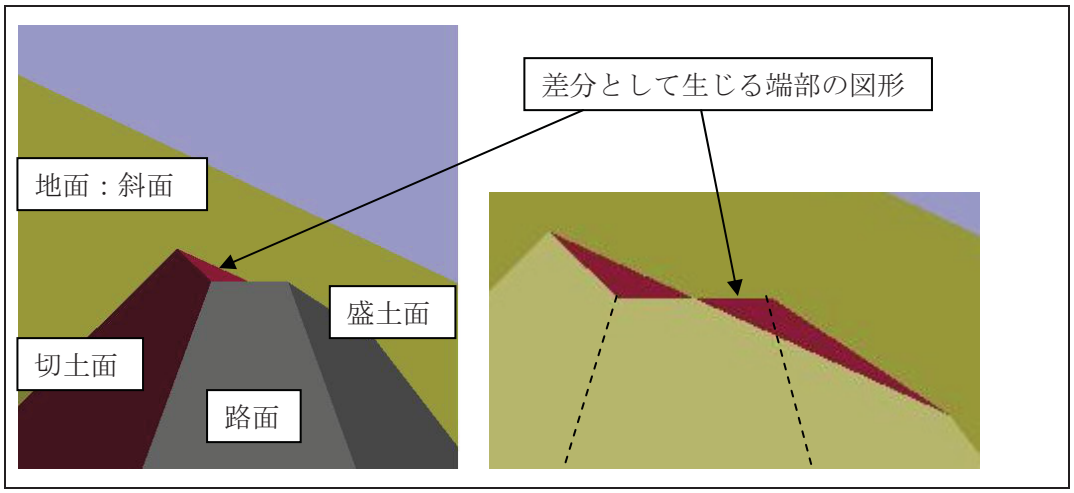


図２－３　地形と構造物の図形演算において端部に自己交差する図形が生じる例

例えば、地形の上に、切土・盛土による法面を有する道路を造成する場合、端部に、差分として捩じれた図形が生じる場合がある（図２－３）。これを表・裏で複数の面に分割し、表側を地形の上に露出した路面の端面として、また裏側を地形の下に埋没した路面の端面

として付加することにより、完結した正常な立体とすることができる。

（５） ひとつの平面上にない多角形

四以上の頂点を有する図形の場合、全頂点が同一平面上にはない図形がデータとしては生成しうる。極言すれば、浮動小数点で各頂点の三次元座標値を表す限り、多かれ少なかれ平面からのずれは生じる。ずれが少なければ、OpenGL による表示は、見た目には正常である。しかし大きいずれがある場合には、視点移動に伴い奇妙に変形する異常な表示となる。小さな図形（たとえば三角形）に分割することにより、異常な表示を解消することはできるが、分割方法は一意的ではなく、その仕方により、異なる図形が生じる。

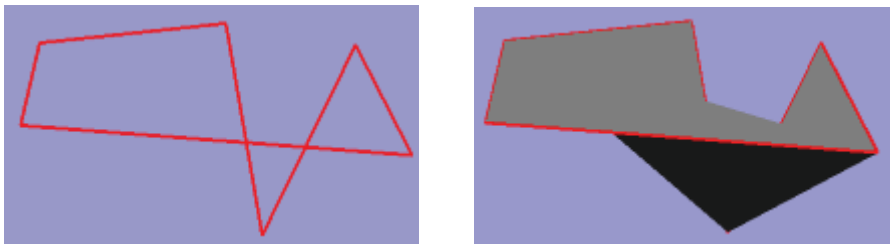


図２－４：頂点が同一平面上にないポリゴンと、OpenGL による表示状態（右）

数値地図のように、自由曲面としての地形をメッシュ型のデータで表現した立体を、サーフェス・モデルに変換する場合に、三角形ではなく四角形のポリゴンに変換すると、同一平面上ではないポリゴンが発生する。遠景の表示などにおいては支障がないが、図形演算処理においては支障を生じる場合がある。

（６） 凹ポリゴン

OpenGL に多角形データが出力された場合、ライブラリの内部処理において、最初の頂点を基準として、三角形に分割されて表示側に出力処理される。このため、多角形の頂角が 180 度よりも大きい頂点を有する凹多角形が正常に表示されない場合がある（図２－５左）。

景観シミュレータにおいては、多角形に凹ポリゴンであることを示すフラグを明示的に与えることにより、正常に表示を行うことができる（処理時間は長くなる）。このことを実現するために、フラグを有するポリゴンの表示に際しては、ポリゴンを OpenGL に出力する前に、プログラムの側で三角形に分割し、出力している。

凹多角形は、表裏に関しては明確であり、実空間に存在しうる図形である。

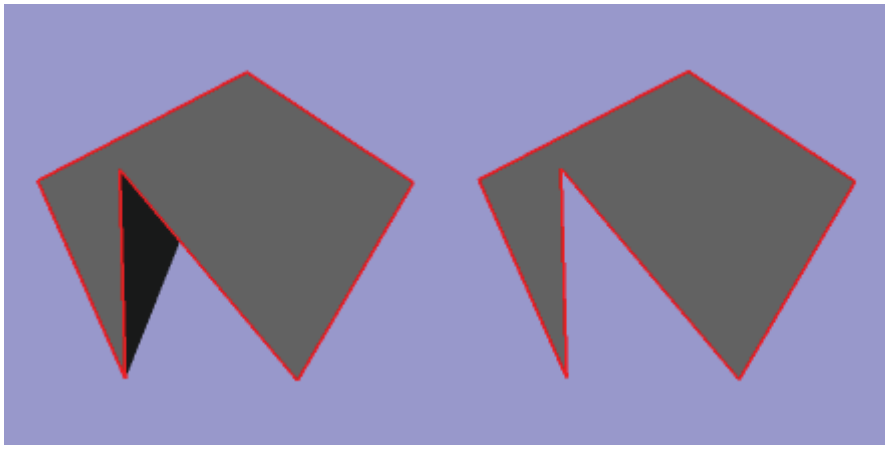


図２－５：凹ポリゴンの表示状態（左）と、分割出力による表示結果（右）

（７） 穴あき図形

図形の中に穴がある場合に、これを記述するデータ構造には、処理システムにより様々なものがある。窓がある建物の壁面など、穴あき図形は実空間においても頻繁に出現する。景観シミュレータにおいては、穴あき図形を表現するために仮想線法を用いた。この方法では、図形の外周を成すポリゴンと、穴のポリゴンを結ぶ仮想の２辺（橋）で結び、恰も穴が無いポリゴン（但し、その内の２辺が、逆向きに重複している）のように表現する。この橋の部分は、図形が生成された時点で頂点座標の比較を行い、仮想線であるフラグを辺の開始点側の頂点(d3Vertex の va メンバ)に付している。ワイヤーフレーム表示において、このフラグを有する辺を表示しないように処理している。但し、仮想線が存在することを確認するために、ポリゴンを含むグループが選択された際の強調表示（編を赤線で表示）に際しては、仮想線も表示している。

これにより、穴あき図形も、幾何学的演算において通常の凹ポリゴンと同様の処理を行うことができる。穴あき図形における仮想線は、他の辺や仮想線と交差してはならない。図形演算により、新たな辺や仮想線が生じ、これが既存の仮想線と交差した場合には、仮想線を適当な位置に付け替える処理を行う。

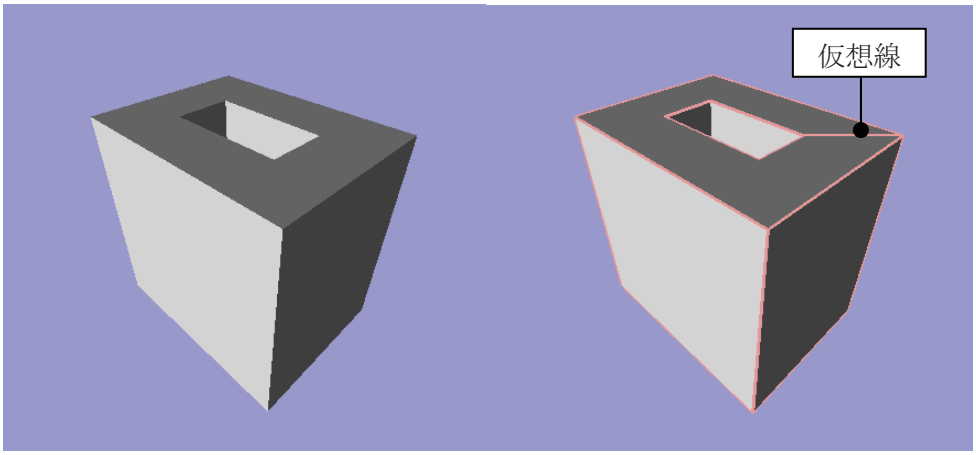


図２－６：穴あき図形と、仮想線

（８）その他

幾何学的には問題なくとも、座標値が不適切な図形が、実務上の支障となる場合がしばしば発生する。国家座標系を用いたモデリングでは、景観検討対象地域を構成する地物の座標地は通常大きな値（原点から数十～数百 km）を有する。作業中のデータに、原点付近の座標値を有する小さな物体が追加されると、それを発見し除去するためには手間を要する場合がある。また幾何学的に正常で、有限の値ではあるが、非常に小さなサイズの（不要な）ポリゴンが発生すると、ノイズのように障害となる場合がありうる。

２－３．立体の完結性（閉多面体）

複数の面により構成された立体が、「閉多面体である」とは、立体の表面が面により隙間なく埋め尽くされていて、かつ隣接する面が同じ向きであることが必要かつ十分な条件である。このことは、全ての辺が、二つの面により共有されていて、その辺を共有する二つの面の向きが等しい事、換言すると、その辺を共有する二つの面の頂点リストにおいて、その辺が逆向きに辿られていることをもって検査できる。いわゆる「メビウスの帯」のような図形、あるいは「クラインの壺」のような図形においては、辺を共有する面の向きがどこかで逆になるために、完結した立体とはならない。しかし、トーラス（ドーナツ形）のような図形は、閉じた図形となる。

閉多面体であれば、任意の点を頂点とし、各面を底面とする立体の体積（但し、高さが負ならば負値とする）の合計として、立体の体積を計算することができる。完結した立体の場合には、この頂点がどこにあっても、計算される体積は一定である。立体が閉じていなければ、頂点位置により不定となる。

また、立体であって、内部に「泡」のような空洞ないし中空部があり、その泡の面が、内側が表となるような閉じた面で構成されている場合、これもまた閉じた図形となる。

内部に空洞（泡）のある立体は、構成するすべての面を、ひとつのグループに帰属させることにより、表現することが可能である。

面＋頂点－辺＝２×（１＋ｎ）　ｎは空洞の数

という関係が成立する。

内部の空洞ではなく、複数の互いに離れて独立した閉多面体を構成する面群が、データ上ひとつのまとまりになることがありうる。このような場合においても、面＋頂点－辺が２とは一致しない。

一つの泡だけを取り出したような図形は、閉多面体ではあるが、体積が負値となり、一般的には不自然である。しかし、例えばトンネルや地下道、あるいは建物のインテリアなど、中空の空間部分だけを取り出して景観検討するような場合には、実用的な意味がある。

閉多面体を構成する面の面積ベクトルの合計はゼロベクトルとなる。しかし、逆は必ずしも真ではない。

ある任意の視点から閉多面体を見たときに、その立体を構成する各面の立体角の合計は、

その視点位置が立体の外部にある場合ゼロとなり、内部にある場合は４πとなる（視点位置の移動に伴い離散的に変化)。完結しない多面体の場合には、視点移動によりこの値が連続的に変化する。

２－４．立体の自己干渉

閉多面体であっても、それを構成する面が相互に干渉する場合が、データとして生成しうる。その場合、交差した範囲において、多重の「内側」となる領域が成立する。現実の空間と物質の世界においては、このような状態は存在しえないと考えられるので、論理チェックすることが望ましい（以後の図形どうしの演算が保証されない)。

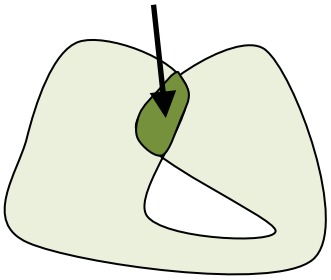


図２－７：立体の自己干渉

２－５．立体図形間の演算

立体 A と立体 B が干渉する場合に、例えば立体 B の内部に含まれることになる立体 A の部分を切除するような演算（CAD 等においては慣習的に図形の BOOL 演算と呼ばれる）の必要性が高い。例えば、山岳地に道路を通す場合、盛土・切土あるいはトンネルなどにより地形を加工する必要がある。その場合、技術基準や設計指針に基づいた法面の形状は、道路の形状から自動的に計算することができるが、計算された法面の内、地形と干渉する領域を計算し、その部分だけを法面として付加する必要がある。同様に、地形の側でも、法面として削られる部分、及び盛土により覆い隠される部分については、削除する必要がある（後者は不可視)。

これを実現するためには、いくつかの方法がある。初期の景観シミュレータにおいては、OpenGL が有するデプス・バッファを用いて実現していた。この方法は、地形と構造物・法面を真上から平行投影した画像を生成し（必ずしも表示する必要はない)、最終的に表示として残されたものが各地点において地形か、構造物・法面のいずれであるかによって、各地点における地形と構造物・法面の上下関係を判定するものである。この方法では、使用する OpenGL 画面の縦横寸法と対象エリアの面積で決まる格子間隔のメッシュ・データで上下判定を行うため、厳密な面と面の交線ではなく、概略の形状が得られるのみである。このため、複雑な地形・道路断面により発生する概略の法面形状を短い時間で検討するためには適しているが、近くから眺める価値のある正確な形状演算ではない。

より厳密な形状演算を行うためには、干渉しあう立体と立体の間で、関係するすべての面について演算処理を行う必要がある。現在の版の景観シミュレータにおいては、仮想線を含む面と面の演算に還元する方法を採用している。これにより、立体どうし図形演算を、切断される立体の全ての面と、切断する立体の全ての面の、全ての組み合わせに対して、面どうしの図形演算を適用する方法に還元することができる（立体１がｍ枚、立体２がｎ

枚の面を有する場合、ｍ×ｎ回の面と面の演算となる。但し多くの場合は空間的に離れていて相互に干渉しないため、存在範囲等の簡単な事前チェックのみで処理は終了する)。

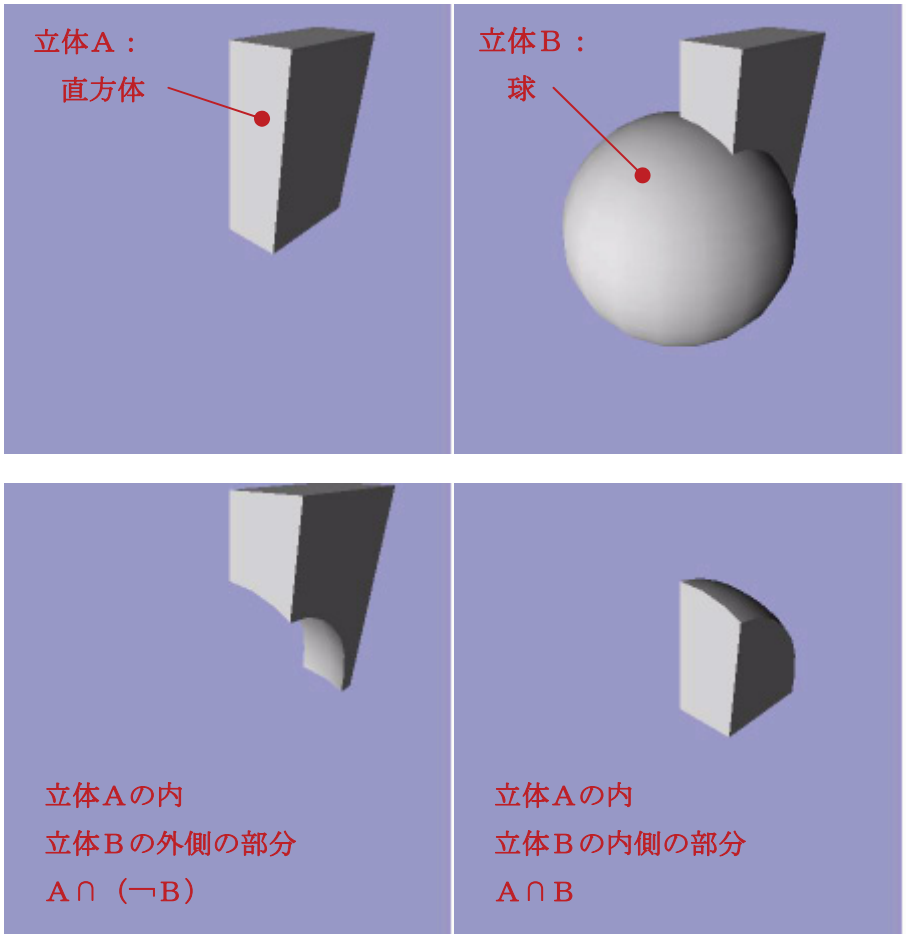


図２－８：箱と球の図形演算

２－６．面の準正常性

立体どうしの形状演算を、面と面の演算に還元するために、面の「準正常性」の概念を導入した。通常の正常な多角形であれば、それを構成する二つの辺が重なり合うことはない。これに対して、「準正常」な面においては、「幅のない切込み」を許容し、これを「反対向きに重なる二つの辺」により表現する。これには、頂点から内部に伸びるもの、辺の中間点から内部に伸びるもの、及び図形の内部に独立した切り込みの３種類がある。切断多角形が被切断多角形の全幅に亘り完全に交差し、被切断多角形が二つに分断される場合には、切り込みの形は生成せず、二つ（以上）の多角形が生成する。しかし、切断多角形が被切断多角形と部分的に交差する場合には、切込みの形状が生成する。適用する立体を構成する面を次々と被切断多角形に対して適用することにより、最終的には、切込みが次第に成長し、最終的には、切断される。このプロセスの中間段階を記述するための、拡張した面の概念である。現実世界においても、紙の多角形にハサミやカッターで切込みを入

れたような状態は存在している。

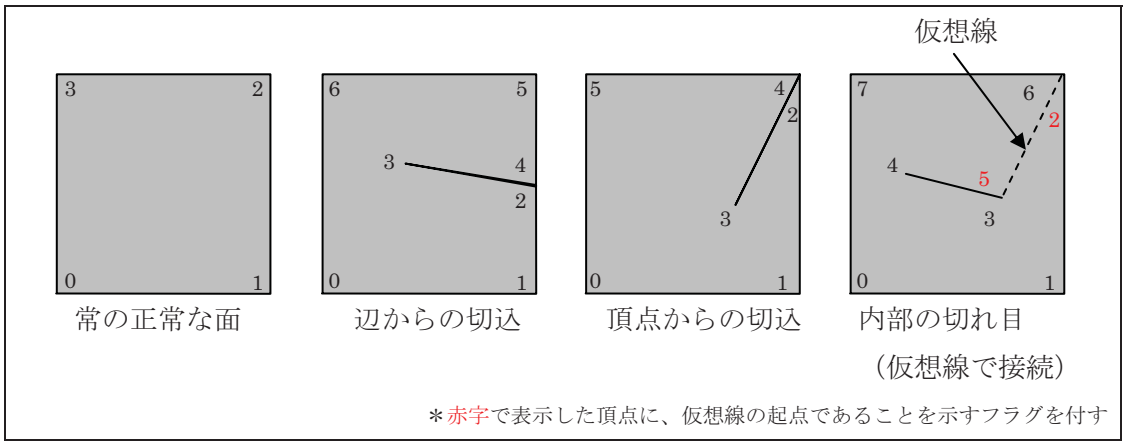


図２－９：準正常な面の例

その際、面と面の演算の結果は、媒介的なデータ無しに、処理結果としての面の形状として表現しており、どの中間段階においても、表示することができる。従って、演算処理がエラー終了した場合に、直前の状態を表示確認することが容易となった。

２－７．面の内外判定

適用する立体の各面を、被切断多角形に次々と適用することにより、切込や切目は次第に（折線状に）延長してゆき、最後に切断多角形により被切断多角形が切断され、切れ込みが解消した時点で、被切断多角形から分離独立した各部分が、切断立体の内側にあるか外側にあるかが確定する。従って、個別の切断に際して、分類を行っておき、最終的に、被切断立体の全ての面の処理が終了した次点で、分類を行うことにより、切断された立体の形状を得ることができる。

切断面との接触がない面に関しては、切断立体との内外関係による判定を行う。切断に関係しない（相互接触しない）面の内外判定のためには、切断立体が閉じた図形になっている必要がある。切断立体の側の面群が閉じていない不完全な場合であっても、交差部分が閉じた折れ線として完結していれば、切断される側の立体を構成する面群における、切断が生じた面との接続関係から辿って内外を判定することは可能であるが、まだ実装していない。

２－８．立体間の相互演算処理

被切断立体の各面から、切断立体の内部に包含される部分を取り除く処理の結果生成する図形は、閉じていない。そこで、この開口部を、切断図形と同じ表面形状で型押ししたように塞ぐことにより、閉じた立体図形を生成することができる。この塞ぐ部分を得るために、切断図形と被切断図形を入れ替えて演算処理を行い、切断の結果内部に取り込まれる部分だけを取り出し、これを先ほどの開口部に取り付ける。「切断図形を押し付けて凹ませた」ように見立てることができる（図２－８）。

２－９．テクスチャ座標

画像データをテクスチャとして面に適用することにより、恰も壁紙を貼ったような効果を得ることができる。テクスチャのピクセルは、RGB とアルファ値を持つことができ、アルファ値が零の点は透明となる。これにより、樹木などの複雑な外形を有するオブジェクトの画像を単純な長方形に適用するだけで、擬似的に樹木を表現することができる。但し、このような透過属性をもつ画像を保存できるデータ形式は限られている。

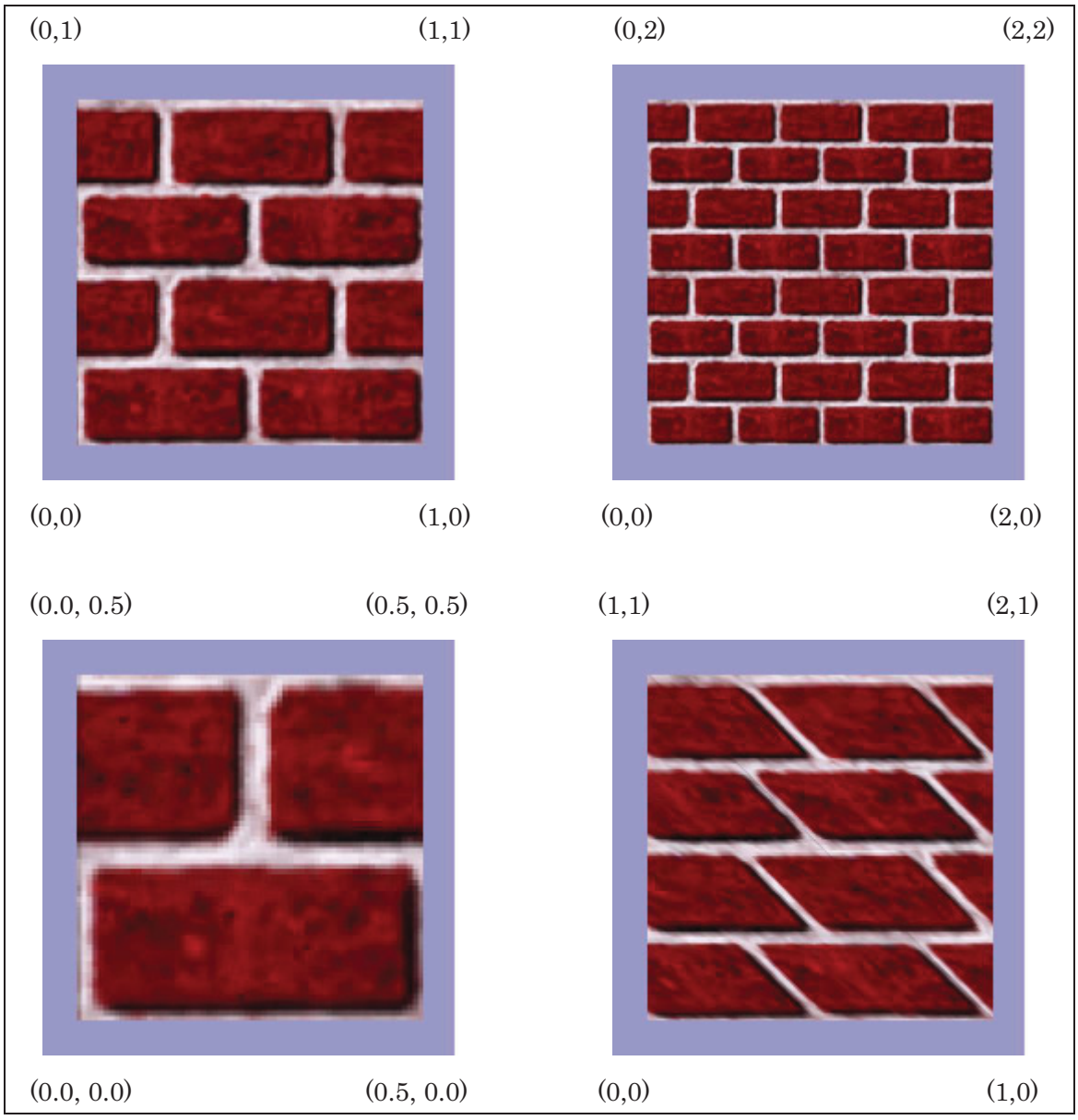


図２－１０：テクスチャ座標設定

図形の側の各頂点にテクスチャ座標（二次元）を定義することにより、貼る位置や縮尺を変えることができる。長方形の画像の中での縦・横の位置は、画像サイズにかかわりなく、それぞれ、0.0～1.0 の範囲の数値で示される。従って、テクスチャ座標として、例え

ば横が 0.3 と 0.6 の値を二つの頂点に適用すると、画像の中間部分だけが使用される。

0 より小さい値、1 よりも大きな値を適用した場合には、格子状にテクスチャを敷き並べた平面から、その座標の範囲を取り出したように、テクスチャが繰り返し適用される。たとえば、0.0 と 10.0 を適用すると、その間にパターンが 10 回繰り返される。従って、タイルやレンガのような、パターンが繰り返される素材をテクスチャとして適用する場合には、最小繰り返し単位となっている部分を画像データとして用意し、繰り返し回数を表現するテクスチャ座標を頂点に適用することで、大きな面を小さなテクスチャ画像（即ち、小さなデータ）で表現することができる。

曲面が複数の隣接する多角形で表現されている場合に、共有される頂点に同じテクスチャ座標を設定することにより、一つながりの曲面としてテクスチャを適用することができる。テクスチャ編集画面においては、一つの立体を構成する多くの面の各頂点に対して、このような座標計算を一括で自動的に行い、一斉にテクスチャを適用するような機能を用意している（４－４(22)(26)(27)）。

２－１０．法線ベクトル

法線ベクトルは、面および頂点に対して定義することができる。通常の多面体の場合（直方体や角錐）、各面の法線は、面に垂直のベクトルである。法線ベクトルが面や頂点に定義されていない場合には、自動的に面に垂直の法線ベクトルがあるものとして表示を行う。

面に明示的に法線が適用された場合、表示に際しての面の明るさは、その法線に垂直な面として計算されるため、例えばタイルを平面的に敷き並べたような物体の各面に対して、揺らぎのある法線を定義することにより擬似的に立体感（凹凸感）を表示することができる。

球や円柱など、曲面をもつ図形を、多数のポリゴンで近似的に表現する場合、共有される頂点に同一の法線ベクトル（例えば球の場合、中心からその頂点に向かうベクトル）を適用することにより、面の明るさが辺や頂点で連続することになり、滑らかな曲面として表示することができる。その場合、多面体を構成する各面は、平面ではあるが、頂点毎に異なる法線をもつことになり、明るさは面の内部で連続的に変化する。

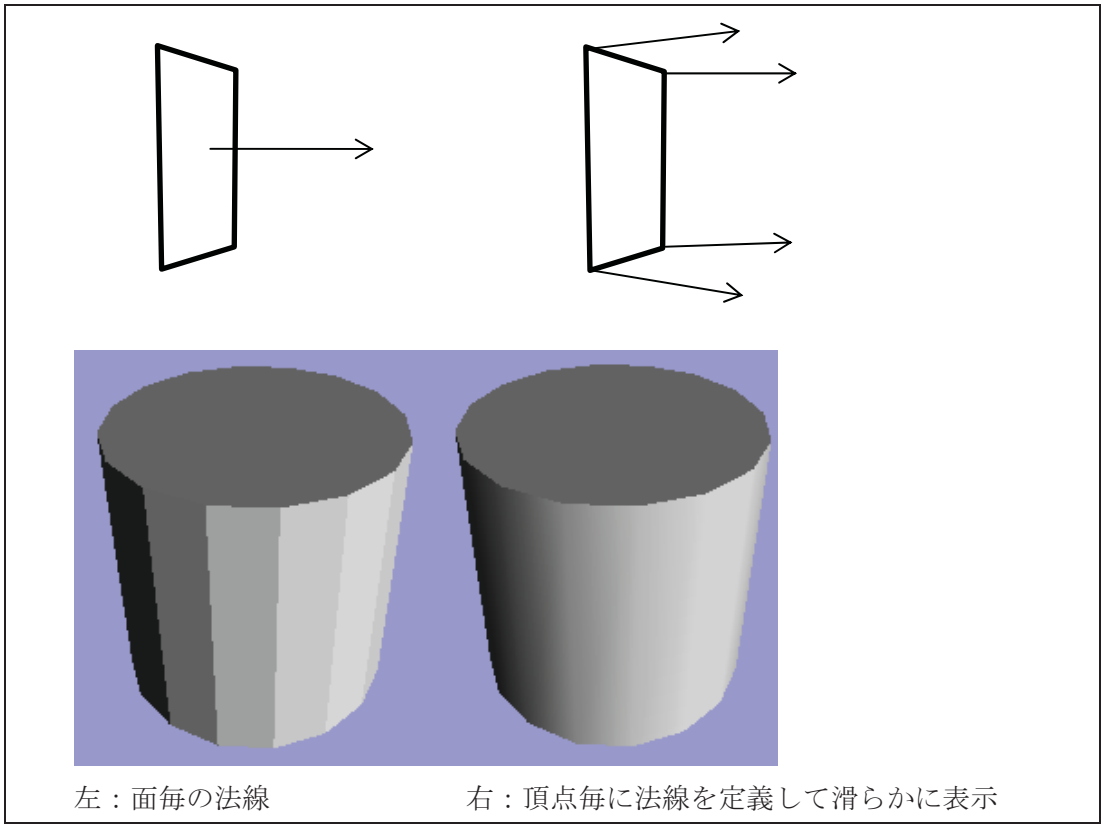


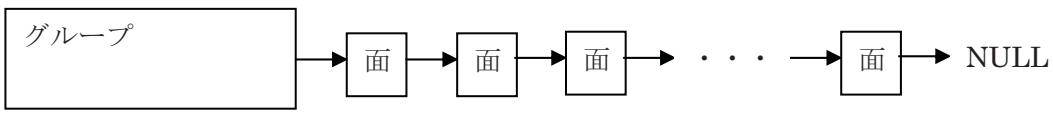
図２－１１：面の法線と頂点の法線

図形演算により発生する新たな頂点に関しては、既存の頂点のテクスチャ座標や法線ベクトルから補間計算し補うことにより、面の質感に係る情報を継承することができる。

２－１１．グループとリンク

景観シミュレータの内部において、「グループ」が重要な単位となる。面（d3Face）は、あるグループに帰属することにより、初めて存在することができ、表示可能となる。さらに、グループは、別のグループとリンク（親子関係）を結ぶことができる。これにより、CAD や GIS におけるレイヤーよりも複雑なデータ構造を表現することができる。

一つのグループは、複数の面をもつことができ、その数に制限はない。典型的には、一つの直方体等、閉じた多面体を一つのグループとして定義し、これに多面体を構成する複数の面を関連づける。しかし、閉じていない図形や、面を持たない（表示されない）グループも、データとしては存在することができる。特に、自らは面をもたないが、複数の部品を束ねる親グループとして機能するようなグループは頻繁に用いられる。



グループのポインタ f が最初の面を指す。面のポインタ next が次の面を指す。

図２－１２：面とグループの関係

グループは、リンクによって階層的に関連づけることができる。例えば、板と４本の柱から構成されるテーブルをモデリングするためにはいくつかの方法がある（図２－１３）。

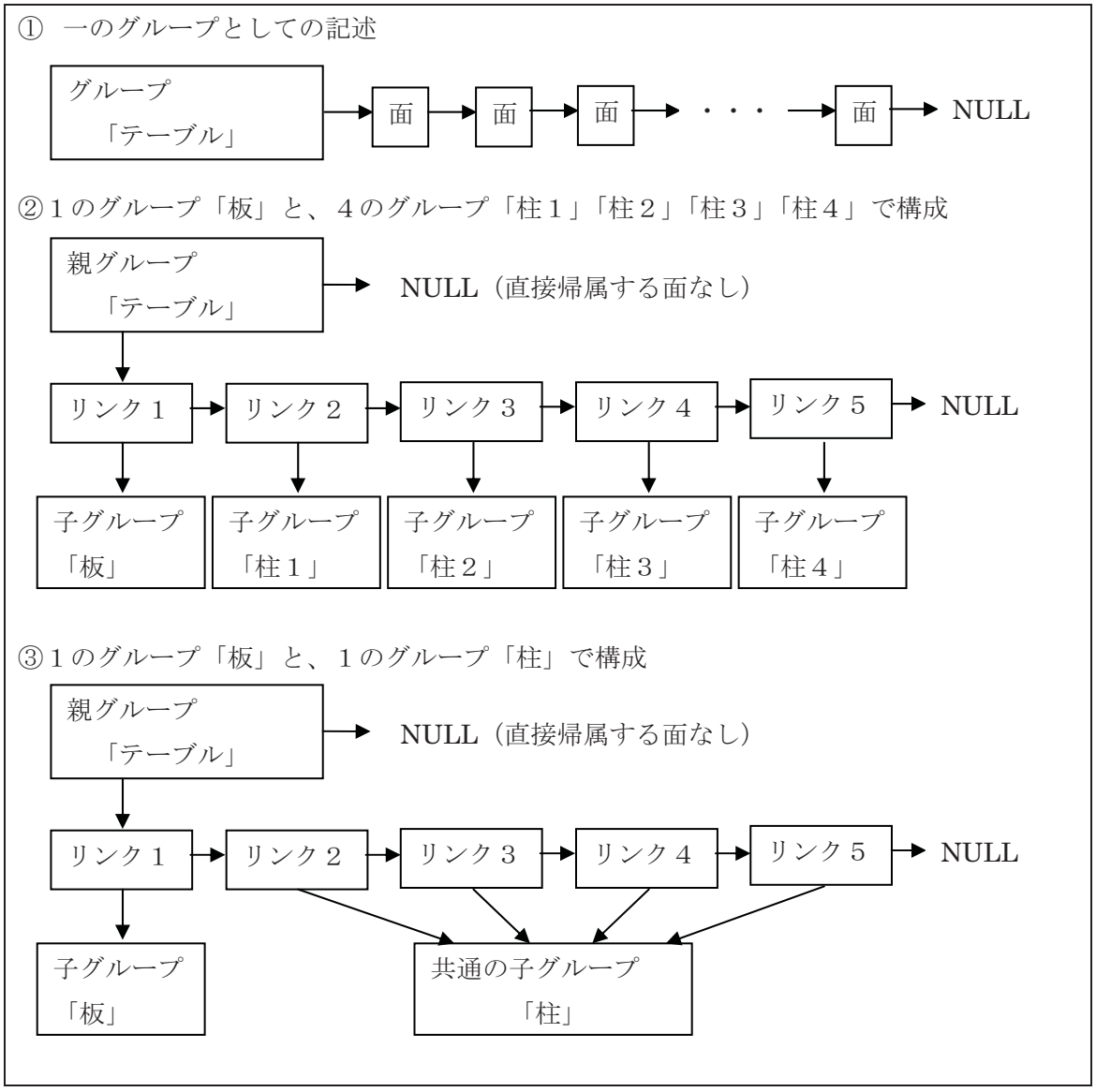


図２－１３：グループとリンクによる構成方法

単一のグループとして「テーブル」を作成し、全ての面をこれに直接帰属させる方法①、５の部品（板と４本の柱）を子グループとして定義し、これを親グループ「テーブル」にリンクする方法②、二種類の部品（「板」と共通の「柱」）を定義しておき、親グループ「テーブル」から板への１のリンクと、柱への４のリンクを定義する方法③が可能である。方法③の場合、それぞれの柱の位置を、リンク・マトリクスの違いによって指定する。共通の柱の側から見ると親グループは４存在するが、全て同じものである。

親グループに「板」を構成する面を直接定義してしまい、子グループの「板」を省略する方法も可能である。最後の方法が、最もメモリ資源を節約できる。

グループは、d3Group 構造体（メモリ・ブロック、サイズ＝144 バイト）により定義している。

リスト２－３：d3Group 構造体によるグループの定義

```
struct _d3Group {
/*private:*/
        int grpid;              ID番号
        int dbno;
        int num_u;              /* 親グループの数 */
        int num_d;              /* 子グループの数 */
        int material;           マテリアル
        int texture;            テクスチャ
        double xmin, xmax, ymin, ymax, zmin, zmax;/* 存在範囲 */
        d3Link *link;           子グループへのリンク
        d3LinkUp *lup;          親グループへのリンク
        d3Face *f;              面
        void *display;          ディスプレイ・リスト
        void *user[D3_USERDATA_MAX];  属性データ
/*public:*/
        char *name;              名称
        char *kind;              ファイル情報
        int type;                種別
};
```

グループに直接的に所属する面は、f メンバに、面を定義する d3Face 構造体（メモリ・ブロック、サイズ＝64 バイト） へのポインタを格納することにより定義される。複数の面は、d3Face 構造体の next メンバを用いてポインタでつなぐことにより定義される。グループに直接的に所属する面がない場合には、f メンバには NULL が格納される。

グループに帰属する子グループが存在する場合には、link メンバに、d3Link 構造体（メモリ・ブロック、サイズ＝144 バイト）へのポインタを格納する。複数の子グループが存在する場合には、d3Link 構造体の next メンバで複数のリンクを定義する。また、子グループの lup メンバに、d3LinkUp 構造体（メモリ・ブロック、サイズ＝8 バイト）へのポインタを格納することにより、親グループの所在を示す。

表示を行う際には、親グループから子グループに再帰的に辿るだけですべての面にアクセスすることができる。しかし、モデリング操作により、グループやリンクを削除する場合には、親グループにアクセスするために、d3LinkUp を用いる。一つのグループが、複数の親グループからリンクされている（例えば、共通のありふれた部品として参照されている）といったような場合が存在するため、d3LinkUp 構造体も、next ポインタでつなぐ構成としている。これにより、一つのグループから、子グループと親グループにアクセスすることができる。上記のテーブルの例に関しては、テーブルの子グループは複数（１の板と４の柱）存在する。一方、柱から見ると、親グループが４存在することとなる。また、このテーブルの柱が、子グループとして別の親グループにもリンクする（部品として参照される）ことができる。

一つのグループに関する、子グループの数、及び親グループの数は、それぞれ、num_u、num_d メンバに整数値として記述される。

リスト２－４：d3Link 構造体によるリンクの定義

```
struct _d3Link {
/*private:*/
        d3Matrix mat;               親グループとの位置関係を表すマトリックス(4x4)
        d3Group *parent, *child;    親グループ、子グループへのポインタ
        d3Link *next;               次のリンクへのポインタ（末尾の場合NULL）
/*public:*/
};

struct _d3LinkUp {
/*private:*/
        d3Link *up;                 親グループからのリンクへのポインタ
        d3LinkUp *next;             次のd3LinkUp構造体へのポインタ(末尾の場合NULL)
/*public:*/
};
```

リンクには、親グループと子グループの位置関係を示すためのマトリックスが定義される。これにより、上記のテーブルの例では、同じ形状の足を異なる４か所の位置に配置することができる。例えば、山（地形）に 10,000 本の桜の木（ファイル部品）を配置した場合、グループは山と桜の木の２個しか生成されず、その間に 10,000 個のリンクが生成され、それぞれが異なるマトリックスをもつこととなる。

モデリング作業においては、例えば配置作業においてグループとリンクの処理が行われる（４－４(11))参照)。ユーザーが部品等を定義したファイルを選択し、位置を指定して、現在の地物の中に新たな要素として付け加える場合に、単体配置を繰り返す方法、リニア配置（指定したラインに沿って指定した間隔で街灯などを配置)、エリア配置（指定した領域に指定した密度で樹木などを配置）の方法がある。単体配置の場合、配置する要素を記述するグループを具体的に構築するのは最初の１個に関してのみであり、２個目からは新たなリンクが追加される。またリニア配置・エリア配置においては、１個のオブジェクトに対して必要個数のリンクを作成する。例えば、山に 10,000 本の桜の木を配置する場合に、桜の木を記述するグループは一つでよい。逆に、このようにして配置された桜の木の１本を削除する操作は、リンクを一つ削除する内部処理により実現される。最後の１本の削除の際に、同時に実体としての桜の木のグループを削除する処理を行う。

言うまでもなく 10,000 本の桜の木を個別的にデータとして作成することもできる。しかしその場合、メモリ上で、あるいは外部ファイルに保存した場合のデータサイズははるかに大きくなる。

グループには、マテリアル、及びテクスチャを定義することができる。グループに所属する面にマテリアルまたはテクスチャが定義されていない場合には、グループに定義されているものを適用する。

また、子グループにマテリアルまたはテクスチャが定義されていない場合には、親グル

ープに定義されているものを適用する。カラーは、グループに直接定義することはできず、面または頂点に定義する。

グループには、name、kind、type の３の属性（必須）が定義される。

name（名称）は、グループのユニークな名称を示す文字列である。モデリングにおいて、ユーザーが指定しなかった場合には、システムの側で、”G314” 等のユニークな名称を自動的に生成して付す。

type （整数型）は、グループの種類を示し、０（通常）、１（ファイル型）、３（編集されたファイル型）がある。１（ファイル型）は、パラメトリックな部品、または固定的な部品を示し、データを外部ファイルに保存する際に、面を構成する頂点座標等の保存を行わず、ファイル名称または部品名とパラメータ値のみを保存する（詳細は３－３、及び５－１以下を参照)。３（編集されたファイル型）は、１（ファイル型）のタイプのグループを構成する面に対してユーザーが切欠きなどの形状変更や、面のカラー編集を行った場合に設定される型で、もはや部品名とパラメータや、ファイル名称などによりコンパクトに定義できない情報を含んでいるため、ファイル保存に際しては、０（通常）のタイプのグループと同様に、配下の面の記述も出力する必要があることを示す。

kind （文字列型）には、通常”unknown”（文字列）が定義される。上記の type がファイル型であった場合には、ファイル名称（文字列）が登録される。ファイル型のグループは、部品を別ファイルとして参照する場合に用いられる。前記のテーブルの例③において、板と、共通の柱（４回使用される）を一つの外部ファイルの中で記述するのではなく、板と柱をそれぞれ別のファイルで部品として定義しておき、テーブルの定義ファイルにおける板と柱の記述は、外部ファイルを参照する形式をとることができる。このことは、繰り返し利用する価値の高い部品をデータベース化するために必要な機能である。

更に、グループには「地面」など、様々の属性を定義することができ、これは user メンバに登録される（詳細は、３－３参照)。

２－１２．リンク・マトリクス

リンク・マトリクスは４×４の行列として定義されており、d3Link 構造体の中に組み込まれている。データとしては長さ 16 の倍精度実数として定義される。

リスト２－５：基本的なマトリクス計算処理

```
定義
typedef double d3Matrix[16];/* OpenGL specification */
基本的な座標変換計算
void d3TransformPointd(double *p1, double *p2, d3Matrix m)
{
        double p[3], w;

        p[0] = p2[0];
        p[1] = p2[1];
        p[2] = p2[2];
```

```
        p1[0] = m[0]*p[0] + m[4]*p[1] + m[8]*p[2] + m[12];
        p1[1] = m[1]*p[0] + m[5]*p[1] + m[9]*p[2] + m[13];
        p1[2] = m[2]*p[0] + m[6]*p[1] + m[10]*p[2] + m[14];
        w     = m[3]*p[0] + m[7]*p[1] + m[11]*p[2] + m[15];
        if (w != 1) {
                p1[0] /= w;
                p1[1] /= w;
                p1[2] /= w;
        }
}
```

三次元空間内座標変換に用いられる。OpenGL の中で基本的な演算処理機能が提供されている。実空間における移動、回転を表現できるほか、座標軸毎の拡大縮小、鏡像変換やアフィン変換など、実空間における地物ではありえない変形も記述できる点は、表示の調整等において便利であるが、部品製作などのモデリングにおいて、このような機能を多用すると、データの具体的な意味（寸法や体積など）が照会された場合に処理が複雑になる点は注意を要する。

座標変換を伴わないリンクの新規設定処理においてはデフォルトで単位マトリクス（対角成分のみ１で残りが０、これを掛けてもベクトルは変化しない）が設定される。リンクにおいては、子グループの面の座標値を列ベクトルとして、前からマトリクスを掛ける演算を行う。データとしては、リンク・マトリクスは二次元ではなく長さ１６の一次元の倍精度浮動小数点の配列として表現する。これを M とし、子グループのローカル座標における座標値をベクトル P0（計算機上は長さ４の配列）、親グループのローカル座標における座標値を P1 とすると、座標変換は以下のように計算される：

$$\begin{vmatrix} P1[0] \\ P1[1] \\ P1[2] \\ 1 \end{vmatrix} = \begin{vmatrix} M[0] & M[4] & M[8] & M[12] \\ M[1] & M[5] & M[9] & M[13] \\ M[2] & M[6] & M[10] & M[14] \\ M[3] & M[7] & M[11] & M[15] \end{vmatrix} \times \begin{vmatrix} P0[0] \\ P0[1] \\ P0[2] \\ 1 \end{vmatrix}$$

このように、ベクトルに XYZ の３値に定数項を加えた４元とすることにより、１回のマトリクス演算で、回転だけではなく、平行移動も含めた計算処理を行っている。

親グループが、更に別グループの子グループとしてリンクされた場合には、上位のリンク・マトリクスが前から掛けられる。

実際のプログラムでは、結果の列ベクトルの４番目の数値が１でなかった場合、１～３番目をその値で除して調整している。

２－１３．マテリアル

景観シミュレータにおいては、面の光学的特性（色彩、質感など）を記述する方法として、直接カラー、テクスチャを数値やファイル名として指定する方法と、マテリアルの名称のみ指定し、その具体的な内容（カラーやテクスチャ等）別途マテリアル・ファイルの

中で定義することで間接的に指定する方法を二通り用意している。

実用的には、色彩がほぼ決まっている定番の材料に関して、マテリアルを用意しておくことにより、個々のオブジェクトに対して手間のかかる色編集を行わずに、マテリアルを適用することで作業が効率化する。頻出する素材（アスファルト、鉄、煉瓦等）や、日本塗料工業会（日塗工）の色見本帳を収録したマテリアル・ファイルを標準で用意している。

これに加えて、あるプロジェクトに関して、同じ材質を共通して適用する部位に関して共通のマテリアルを設定しておき、マテリアル自体を編集することにより、リアルタイムでの景観検討の手間を省くことができる。例えば、マンションのバルコニーの塗装を検討する場合、数多くのバルコニーの色を同時に編集するためには、例えば「バルコニー塗装」という共通のマテリアルを適応してモデリングを行った後に、このマテリアルの色彩を編集する方法が可能である。

マテリアルにはまた、経年変化を検討するために、経過時間（日数）で区切って異なるカラー値や、異なるテクスチャを与えることができる。これにより、木材の熟成や、塗装の経年劣化などを表現することができる。

更に、現在の景観シミュレータが出力部分でグラフィック表示に使用している OpenGL ライブラリでサポートされている表面光学特性の内、直接編集機能や外部ファイルでの記述を用意していない特殊な属性（輝度や鏡面反射率など）に関しても、マテリアルの中で定義し、表示に反映させることができる。

マテリアルは、グループや面に対して適用することができる。しかし、カラーのように頂点単位で細かく設定することはできない。また、マテリアルには法線ベクトルは含まれない。

表２－１：部位と設定可能な属性の対照表

	グループ	面	頂点
カラー	×	○	○
法線ベクトル	×	○	○
テクスチャ	○	○	テクスチャ座標
マテリアル	○	○	テクスチャ座標

あるグループに対して指定されたマテリアルは、そのグループに属する面、およびそのグループにリンクした子グループに対してデフォルト値として機能する。即ち、個々の面や子グループに関してマテリアルが指定されていれば、そのマテリアルが親グループに適用されたマテリアルに優先して表示に反映される。面にマテリアルが指定されていなければ、グループに適用されたマテリアルが表示に使用される。

景観シミュレータにおいては、最上位のルート・グループに関しては、デフォルトのマテリアルを固定的に設定しており、これはユーザーが編集できないようにしてある。これにより、全てのグループおよびこれに帰属する面のデフォルト値は、ルート・グループのデフォルトのマテリアルとなる。このデフォルト・マテリアルは、明るい灰色に対応する

カラー値{0.8, 0.8, 0.8}を持ち、その他の項目（テクスチャや輝度、反射率）を持たない。

面に関して、カラー定義を含んだマテリアルと、カラー値が共に設定されている場合には、後者の方が優先される（面の標示を行っている drawMuka(d3Face *f) の中で、setMaterialTexture(f->mat, f->tex) を実行した後に、setColor(float *c) を実行するため、前者が設定されても後者で上書きされる）。

マテリアルと個別のテクスチャ・ファイル名の両方が設定されていて、マテリアルの中にもテクスチャ・ファイル名の定義が含まれている場合には、表示において前者を使用する。

面および頂点に定義されたカラー値は、表示段階で setColor 関数の中で、diffuse[4]（光源に対応するカラー）と ambient[4]（環境光に対応するカラー） の数値に変換され、OpenGL ライブラリに送出される。この時、diffuse にはカラー値×1.0 が、また ambient にはカラー値×0.2 が渡される。

カラー値は長さ 4 の単精度実数配列として表現されている。その値は赤、緑、青の三原色とアルファ値となっている。数値の範囲は、0.0～1.0 である。アルファ値は不透明度を表し、低くなる程透明となる。透明な物体においては、描画段階で、既に描かれている背後の物体に物体色を上書きするのではなく、アルファ値に応じた比例按分により混合を行うことで実現されている。

OpenGL への最終的なデータ送出は、

```
glMaterialfv(GL_FRONT_AND_BACK, GL_AMBIENT, amb);
glMaterialfv(GL_FRONT_AND_BACK, GL_DIFFUSE, dif);
```

という関数により実行している。OpenGL の基本的な機構が状態マシンであることから、基本的には、この関数によりカラー値が設定された後、異なる値のカラー値が設定されるまでの間、送出される頂点には、設定されたカラーが適用される。しかし、Windows VISTA の OpenGL では若干仕様が変更されたように見える。即ち、glBegin-glEnd の内側で設定されたカラーは、その直後に送出された頂点にのみ適用され、以後の頂点には適用されない。このため、高速表示モード（７－６参照）において、GL_TRIANGLES、GL_QUADS 等の機能を用いて glBegin、glEnd を省略する場合、例えば三角形が連続する途中にカラーの変更が行われるような場合に、表示が従来とは異なる結果となる。このため、プログラムの側で面のカラー値が変化したかどうかについて検査を行い、必要であれば glEnd →カラー設定→glBegin を送出するように修正を行うことで対応した。

２－１４．線のデータ

景観シミュレータにおいては、線のデータ形式を用意している。景観を構成する地物の直接的な表現においては、線が具体的な構成要素となる場合は余りないが、作業の途上での補助線的な使用や、CAD 図面の変換結果を作業下図に用いる場合の表示などに便利である。また、道路や堤防などの長尺物の断面や中心線軌跡を線のデータとして予め用意して

おき、これを用いて掃引体等の形状生成機能（外部関数）により立体を生成するような用法がある。更に、平面として表示困難な図形（８の字形など）が生成した際にエラー処理する中で、線のデータに変換して表示し、原因の理解を助けるような用法がある。

線は、頂点の配列として表現されるため、データ構造としては面とほぼ等しい。従ってd3Face 構造体により定義し、属性として D3_SHP_LINE を付与している。表示にあたっては、最初の点から始まり最後の点で終わる折れ線として処理する。従って、面の場合とは異なり、頂点の順序を巡回的に移動させると、表示される実体は変化する。ループとして閉じた折れ線を表現するためには、最初の点と同じ座標値をもつ最後の点を定義しなければならない。

線に対しても、カラー、法線、テクスチャ、およびマテリアルを定義することができる。この内、線の表示に当たって直接的な意味をもつのは、カラー、およびマテリアルの中で表現されたカラーのみである。線分の標示におけるカラーは、光源とは無関係に、単純に定義された色で描画を行う。従って、暗闇の環境条件であっても同じように表示される。

法線、テクスチャおよびそれらの定義を含むマテリアルが線に対して定義された場合、これを断面形として用いて道路や堤防などの長尺物を生成した際に、生成される立体を構成する面にそれらのデータが反映される。

なお、面の表示との一貫性を保つために、カラーが定義された線であっても、ワイヤーフレーム表示（外形線のみの表示）においては、立体の稜の表示に使用するのと同じデフォルト色の線として表示する。

３．外部ファイル形式

３－１．概要

（1）ＬＳＳ－Ｓ形式とＬＳＳ－Ｇ形式

景観シミュレーション・システムでファイル入出力や通信に用いるデータは、独自のデータフォーマットであるＬＳＳフォーマット[補注)]となっており、適用事例や、共通性の高い景観構成要素を登録したデータベース等の蓄積保存には、この形式を用いている。

初版から、データ形式に関しては全く変更しておらず、1996 年に配布開始した初版のシミュレータで作成したデータを最新版で利用することができ、また最新版で作成したデータを初版で開くことができる。各分野で実務的に必要とされる各種のデータ形式とのデータ交換のためには、ファイル・コンバータを増補することにより対応してきた（第８章)。

三次元的な地物を表現するためのフォーマットには、ＬＳＳ－Ｓ（シーン）とＬＳＳ－Ｇ（ジオメトリ）がある。地物に固有の属性と、表示に必要な環境条件を別ファイルに分離している点に特徴がある。

どちらもテキスト･ファイルであり､エディタ等で開いて内部を解読することができる。開発に着手した 1993 年頃､データ形式の検討段階で､OpenGL をベースとした Open Inventor を参考としたため、その後開発された VRML とは親和性が高い。

記法は「;」（セミコロン）を記述の区切りとするＣ言語の記法となっている。一つのコマンドを複数行にまたがって記述することができる。ラベルの大文字小文字は区別する。ラベルを構成する文字はトークン、禁則文字以外すべて許される。

ラベルの途中で改行があっても、接続して一つの文字列として処理する。コメント行、改行、スペース、タブコードはロードの際に取り除かれ、保存する際には再現しない。Ver.2.09 においては、ラベルには ASCII コードから派生した、例えばシフトＪＩＳによる日本語などの２バイト系言語の使用を排除していないが、異なる言語のＯＳ上への可搬性は保証されない。UNICODE による２バイト系文字列を用いたファイルはサポートしていない。

リスト３－１：コマンドの基本形

```
(1)コメント行
#文字列
(2)ラベル付のコマンド
ラベル=コマンド(引数 1, 引数 2, 引数 3, .... );
(3)ラベルの無いコマンド
コマンド(引数 1, 引数 2, 引数 3, .... );
```

ＬＳＳ－Ｓ形式は、シーンファイルを構築するデータフォーマットで、ＬＳＳ－Ｇ形式で定義された地物を、二次元画像として一意的に表示するために必要となる各種環境条件（光源、時刻、視点位置等）の定義をしている。ＬＳＳ－Ｓ形式を構成する各コマンドに

ついては、３－２で解説する。

ＬＳＳ－Ｇは、環境に依存しない地物固有の性質（幾何学的形状、表面仕上などの各種属性）を記述するジオメトリファイルを構築するデータフォーマットである。また環境条件の一部である時間の関数として表現できる、地物の規則的な経年変化（時間の関数として表現できる形状や表面仕上げ）も、ＬＳＳ－Ｇの中で定義する。ＬＳＳ－Ｇ形式を構成する各コマンドについては、３－３で解説する。

ＬＳＳ－Ｓ形式とＬＳＳ－Ｇ形式に共通するコマンドについては、３－４で解説する。

通常の景観検討の作業は、まず地形や設計対象物などを定義するＬＳＳ－Ｇ形式のファイルを構築し、次に複数のＬＳＳ－Ｇファイル（計画案選択肢）を参照すると共に、視点場や光源条件などを設定するＬＳＳ－Ｓファイルを作成する手順で進められる。ＬＳＳ－Ｇファイルの構築に際しても、部品を定義するＬＳＳ－Ｇファイル群をまず作成し、次にこれらを配置し組み合わせることにより、より大きな、複雑な地物構成を表現する上位のＬＳＳ－Ｇファイルを作成する、という手順で作業が進められる（３－５）。

ＬＳＳ－Ｓファイル及びＬＳＳ－Ｇファイルは、通常は、本システムの各種機能を用いて画面操作を行い、結果をファイル保存することにより作成される。また、テキスト・エディタで作成・編集することも、あるいは、ファイル・コンバータや形状生成を伴うシミュレーションを行うアプリケーションの出力として作成することもできる。

ＬＳＳ－Ｓファイル及びＬＳＳ－Ｇファイルの入出力に関する機能は、スタティック・リンク・ライブラリの一つである IP.lib により提供している（４－２）。

なお、編集ダイアログ等においてシステムが自動的に生成するラベルには、パターンが存在する（例えば、時間のラベルは「T00X」といったスタイル）。しかし、ユーザーが手入力するラベル、あるいはコンバータ等が生成するラベルが、この慣習に従う必要は全くない。以下のような人を惑わす記述であっても、ラベル名に関わりなく、用いられたコマンドの規約に従って、整合性がある限り正しく処理される。

リスト３－２：紛らわしいが許されるコマンドの例

```
MODEL1 = COLOR(0,0,3);
TEXTURE1 = VERTEX(COLOR1,COLOR2,COLOR3);
MATERIAL1 = FACE(TEXTURE1,TEXTURE2,TEXTURE3);
COLOR = FILE(TRANSPARENT.GEO);
MESSAGE = GROUP(URGENT);
```

（２）各種画像ファイル

景観シミュレーションを行うために、現況写真などの画像を背景・前景として使用する場合、及び樹木や構造物の仕上げ面を画像で表現するためのテクスチャに各種ファイルを使用する。本システムの初期のバージョンにおいては、ＳＧＩ形式を標準としていたが、その後デジタルカメラや、インターネットによる画像交換の普及に伴い、画像ファイル形

式が普及したため、これらも利用できるようにした。これらについては、３－８で解説する。ファイルの入出力処理には、既存のオープン・ソースを使用したもの(SGI, JPEG)、Windowsに関連したAPI関数を使用したもの(BMP)、仕様書などから独自に作成したもの(その他）がある。

（3）マテリアル・ファイル

ＬＳＳ－Ｇファイルを構成する各種地物（地形や、住宅・社会資本等）の材質を記述するためには、ＬＳＳ－Ｇファイルの中で、各種オブジェクトの色彩やテクスチャを直接記述することもできるが、頻繁に使用される各種材料に関して、あらかじめ共通のマテリアル・ファイルに、各種材料（鉄、コンクリート、木材、塗料、水面、芝生等）毎に、表示に必要な光学的属性を定義しておき、これを間接的に参照することにより、ＬＳＳ－Ｇファイルの中での材質に関する記述をシンプルにすることができる。

各種マテリアルには、色彩やテクスチャ以外に、鏡面反射率、輝度、テクスチャと、その経年変化を記述することができる。将来、グラフィクスの表現技術が高度化した場合であっても、マテリアル・ファイルによる定義を詳細化することにより、ＬＳＳ－Ｇファイルは変更修正することなしに対応することができることを配慮している。

マテリアル・ファイルの形式については、３－６で解説する。マテリアル・ファイルの読み込みは、DBILライブラリ（４－２(3)参照）のdbLoadMaterial関数により行う。マテリアル・ファイルを構築するための編集ダイアログ等は、Ver.2.09の段階においてもまだ存在しない。テキスト・エディタにより入力・編集している。外部関数として実装されたファイル・コンバータ VRML2LL.exe の中に、変換元のファイルにおいて適用された鏡面反射率や発光体等の属性をマテリアル・ファイル形式で出力する機能が組み込まれている。

（4）データベース関連ファイル

景観シミュレーション・システムの操作に際して使用する、①事例、②構成要素、及び③材料の３種類データベースの登録データを記述するためのファイル(com.txt)と、分類体系を記述するためのファイル(XXX.cls)を使用している。これらについては、１４－２、１４－３で解説する。

Com.txtについては、DBILライブラリ（４－２(3)参照）に入出力の機能を用意している(dbMakeData関数、dbSave関数)。また、３種類のデータベースに対応する検索機能(yuu.exe、kou.exe、zai.exe)のほかに、入力のために、専用エディタ editor.exe を用意している。

（5）環境設定ファイル

景観シミュレーション・システムの操作環境を定義するために、環境設定ファイル(標準名kdbms.set)を用意している。これについては第12章で解説している。

この他に、ステレオ表示の条件を記述する小さなファイル(eye_param.set)がある（７－

４参照)。

（６）エラーメッセージ定義ファイル

エラーメッセージは、初期から外部ファイル err_msg.txt により定義してきた。Ver.2.09 においては、これを継承し、言語別にファイルを用意することとしている。詳細は、３－９で解説する。

（７）ヘルプ・ファイル

ヘルプ・ファイルは、テキスト・ファイルであり、特に形式を定めていない。表示された状態でユーザーが編集・追記し、上書き保存することができる。拡張子は、XXX.txt である。ファイル名称 XXX は通常、基幹部分およびプラグインＤＬＬの各ダイアログの名称を用いている。Ver.2.09 においては、多言語の管理のために、言語別のディレクトリに置くと共に、ファイル名称を XXX.yy.txt としている(yy は言語コード、例えば、haichi.ko.txt は韓国語による配置操作のヘルプ)。表示処理について１１－６(4)で解説する。

（８）選択項目定義ファイル

システムの拡充に伴い、プログラムを修正することなく選択肢を拡充できるように、いくつかのコントロール・ファイルを用意している。例えば、頻繁に使用するテクスチャを登録する autotex.set、ユーザーが追加することができるパラメトリックな形状を生成する外部関数を登録する ext.tab、道路断面を登録する roadsec.set や河川断面を登録するファイル riversec.set 等である。これらについては、３－１０において解説する。

（９）多言語対応のための訳語格納ファイル

基幹部分(sim.exe)の実行に際して、使用する言語の変更が行われた場合に、メニュー項目や、各種ダイアログ中のコントロール（操作ボタン等）の表示に用いる文字列を各国語の訳語に変更するために、訳語格納ファイルを xml 形式で用意する必要がある。

外部関数およびプラグインＤＬＬに関しても、多言語に対応する場合には、それぞれについて、同様の xml 形式の訳語格納ファイルを用意する。

プログラムが改良・修正され、必要となる表示文字列が増加した場合には、基幹部分の多言語対応機能により、リソースファイルと xml のタイムスタンプを比較し、自動的にバージョン管理を行っている。これらについては、第 11 章で解説する。

（１０）その他の一時的ファイル

また、関連する実行形式の間でデータを交換するために一時的に作成されるファイルがいくつか存在する。これらについては、３－１１において解説する。

（補注：LSS は Landscape Simulation の略である。開発初期の構想の中で、景観データベースを構築するにあたり、

優良景観事例にＬＳＳ－Ｓ、景観構成要素データベースにＬＳＳ－Ｇ、景観材料データベースにマテリアルを登録する、という概念が存在していたが、開発はそのようには進展しなかった。現在は、いずれのデータベースにおいても、ＬＳＳ－Ｇファイルが登録されている。ソースコード中の変数名等に一部そのような概念の名残がある。）

Ver.2.09 のインタープリタにおいては、コマンドに関して、以下のように処理している。

(1) 名称の前後のスペース、タブコード(0x07)、改行文字(0x0d)は無視する。

(2) 名称に２バイト系の文字が含まれていても、処理を行う。

(3) コマンドのないトークン「;」：

エラー（コマンドがない）

(4) 最後のトークン「;」の後の記述（トークンの無いファイル終了）：無視

(5) トークン「;」の間にスペース、タブ、改行しかない場合：無視

(6) 名称に含まれる日本語のホワイトスペース：文字列の一部として処理

(7) 名称に含まれるスペース：詰めて処理(「a b c d e」→「abcde」

(8) 名称が”引用符”で囲まれていた場合：文字列に含まれるスペースを詰めない

(9) 名称に”引用符”で囲まれた部分が複数あった場合：連結。”1 “ “ 2 “ “3” →「 1 2 3」

(10) 始まりの欠落した“引用符”:無視。「 abcd”」 →「abcd」

(11) 文字列の途中にある”引用符” :「ab”cd”ef」→「abcdef」

(12) 名称が数字であった場合：名称として処理。 [例]0 = GROUP(); は受け付ける。

(13) 数値の中にスペースがあった場合 3. 14 →3.14

(14) 数値が引用符「”」で囲まれていた場合：数値として解釈

(15) 数値が引用符「”」で囲まれていて、その中にスペースが含まれる場合：スペース以前までを評価

(16) 数値が引用符「”」で囲まれていて、その冒頭にスペースがあった場合：スペースを無視(“ 3.0” →3.0)

(17) シングル引用符「’」:文字列の一部として解釈

(18) 小数点以上の省略 .3 → 0.3 として解釈

(19) 小数点以下の省略 3. →3.0 として解釈

(20) 小数点のみ . →0.0 として解釈

(21) 小数点が複数あった場合：「1.2.3.4.5」→1.2

(22) 指数表現：可 1e3 →1000.0

(23) 引数の数値が省略された場合（,,,）→(0.0, 0.0, 0.0, 0.0)として解釈

(24) コマンドとして定義されていない文字列があった場合：エラー（コマンドが未定義）

(25) LSS-S ファイルの中に、LSS-G 系列のコマンドが使用された場合：

解釈するが表示には反映しない。エラーがあればメッセージを作成。

(26) LSS-G ファイルの中に、LSS-S 系列のコマンドが使用された場合：

解釈するが表示には反映しない。エラーがあればメッセージを作成。

(27) コマンド文字列の中に空白があった場合：詰めて解釈 「G 1 = FI LE(SA MPLE);」→「G1=FILE(SAMPLE);」

(28) コマンド文字列の中に改行があった場合：詰めて解釈

(29) 引数部() が欠落した場合：エラー [例] G1 = FILE;

(30) 引数の括弧閉じ「)」が欠落した場合：エラー

(31) 引数が不足する場合：エラー　但し、コマンドにより、省略を許容する場合がある。

(32) 引数が超過する場合：インタープリタでは全て拾い処理に渡す。処理は各コマンドによる。

(33) 定義されていない引数（文字列）が引用された場合：エラー。前方参照は行わない。

３－２．ＬＳＳ－Ｓコマンド

ＬＳＳ－Ｓ（Landscape Simulation-Scene）ファイルは、画面の背景画像、前景画像、表示する三次元モデル（３－３で解説するＬＳＳ－Ｇ形式のファイル)、経過年数（建築後年数)、視点、太陽位置などの光源、及びその他の表示に係る効果を記述するファイルである。テキスト・ファイルであり、インタプリタ（ＩＰ）により解釈され、シーンに関するデータを扱うＳＭＬライブラリ（４－２参照）を介してメモリ上に格納される。

ＬＳＳ－Ｓの拡張子は".scn"である。ＬＳＳ－Ｓファイルは、ＬＳＳ－Ｓコマンドにより記述されている。

開発初期においては、ＬＳＳ－ＳおよびＬＳＳ－Ｇのコマンドをオペレータ（プログラマ）が１行ずつコマンドとして入力し、テスト・デバッグを行っていたようであるが、システムとして完成したシステムの中には、１行ずつ入力するコマンドラインのコンソールなどのインターフェースは残されていない。従って、通常は外部ファイル（ＬＳＳ－Ｓ、ＬＳＳ－Ｇ)の中に記述されたコマンドとして､インタープリタによりバッチ処理される。

ただし、ファイル経由で１行ずつハンドシェークしながら入力・実行する機能は残されており､「成熟都市シミュレータ」はこの機能を利用して、市街地の変化（建物の追加・削除）をシミュレーションし、その結果を、リアルタイムで景観シミュレータに出力・表示するインターフェースを用いている。逐次コマンドが処理され変化する表示は、アニメーションとなる。

リスト３－３：ＬＳＳ－Ｓファイルに用いられるコマンドの一覧

```
TIME
CAMERA
LIGHT
LIGHTGROUP
EFFECT
EFFECTGROUP
MODEL
IMAGE
SCENE
OUTPUT_SCENE
```

（１） 時間 TIME

時間データを作成する。時間は、日数を単位として、浮動小数点値で示す。景観シミュレータは、現在の時間データを保持しており、表示に際しては、材料の経年劣化や熟成など、時間に依存して変化する性質を有する地物に関して、このシステム時間を適用した表示を行う。ＬＳＳ－Ｓ形式の記述の中では、後述するSCENEコマンドの中で、時間を数値で直接指定するのではなく、TIMEコマンドで定義されたラベルを使用する。

[記法]　時間名称＝TIME(日数);

　日数　　経年変化を表す（浮動小数点値）

[例]　　T1=TIME(730.0);

①日数は、負であっても構わない。

②日数が空欄または数値として解析できない文字列の場合にはゼロとする。

③引数が２以上あった場合には、最初の引数を日数として解釈する。

④同一ラベルで再定義が行われた場合、パラメータ（日数）だけを更新する。この更新は、そのラベルを用いて時間を定義している全てのシーンに遡って適用される。

⑤時間が設定、修正された場合には、時間に依存するモデルの再構築が行われる。

（２）カメラ情報 CAMERA

視点データを作成する。三次元で表現された地物を、二次元のディスプレイに透視図として表示（レンダリング）するためには、視点位置が必要である。景観シミュレータにはユーザーが視点位置を移動することにより様々な位置・視角で地物を表示する機能が備わっているが、地物自体には、「見どころ」の視点場などを保存する機能はなく、ＬＳＳ－Ｓで記述する。

[記法]視点名称＝CAMERA(EYE_X, EYE_Y, EYE_Z,
　　CENTER_X, CENTER_Y, CENTER_Z,
　　UP_X, UP_Y, UP_Z,
　　FOVY, ASPECT, ZNEAR, ZFAR);

EYE_X	視点のＸ座標値
EYE_Y	視点のＹ座標値
EYE_Z	視点のＺ座標値
CENTER_X	注視点のＸ座標値
CENTER_Y	注視点のＹ座標値
CENTER_Z	注視点のＺ座標値
UP_X	天頂方向ベクトルのＸ座標値

UP_Z	天頂方向ベクトルのＺ座標値
FOVY	焦点距離
ASPECT	アスペクト比
ZNEAR	視点からの見え始める距離
ZFAR	視点からの見えなくなる距離

パラメータはＯpenＧＬの gluLookAt()及び gluPerspective()関数に適用する。

カメラ位置は、視点位置（カメラの位置)、注視点（中心に見えるポイント)、アップベクトル（カメラの視点－注視点を軸とする回転角）を含む。注視点は、視点から注視点に結ぶ半直線上のどこに移動しても表示に影響は無いが、景観シミュレータにおける視点移動における「回転」は、注視点を中心として視点位置を回転させているため、以後の視点移動の動作に違いが顕われる。

F（焦点距離）は、35mm カメラの焦点距離に対応するパラメータである。OpenGL で用いる視野角 fovy とは、以下の換算式で関連づけている。

fovy = atan(17.5 / F) * 114.591559026164641753596309628 2

F = 17.5 / tan(fovy * 0.008726646259971647884618453842443 06

焦点距離を小さくとると、広角の表示となり、同じ対象物は中心に小さく描画される。大きくするとズームアップ状態となり、遠くの物体の局部が大きく表示される。

ASPECT は、表示画面の横／縦比率で、通常は画面のプロポーションに合わせる。画面のサイズ変更に際して、この値を正しく追随する必要がある。

zNear－zFar は、表示において、手前にある地物によってその裏側の地物が隠される関係を計算するＺバッファの有効範囲（距離）を定義している。近くの物体、遠くの物体をあえて表示しないために､意図的に設定する場合がある｡全ての地物を表示する場合には、zNear を十分小さく、また zFar を十分大きく設定しなければならない。しかし、グラフィックス能力が低いマシン上では､zNear-zFar のレンジを不必要に大きくとると､前後判定の精度が下がり、辺が鋸線状に表示されるような異常が生じる場合がある。

（３）光源 LIGHT, LIGHTGROUP

光源を定義する。光源は、光源ユニットと、それを組み合わせた光源グループを用いて定義する。ひとつのシーンに対して、ひとつの光源グループが定義される。ひとつの光源グループは最大８の光源から成る(三次元表示に使用している OpenGL ライブラリの制約条件に由来)。

（光源ユニット）

[記法]　光源ユニット名称＝LIGHT(光源タイプ, X, Y, Z, W, R, G, B);

光源タイプ	光源タイプの種類

一般の光源　　：０
緯度経度、月日、時分から自動計算した４光源の場合：
　　太陽光源　　：１
　　副光源太陽の反対側　：２
　　副光源東側　　：３
　　副光源西側　　：４

X　　位置のＸ座標値

Y　　位置のＹ座標値

Z　　位置のＺ座標値

W　　位置のＷ座標値

(X, Y, Z, W)は同次座標。４の値が１コマンドでOpenGLに渡される。その際に、Wが0.0であれば、座標点から原点に向かう平行光源が、また、Wが1.0であれば、座標点から八方に発せられる点光源が定義される。

R　　色：RED

G　　色：GREEN

B　　色：BLUE

(R, G, B)は色で、0.0≦≦1.0とする。

Ver.2.09において、
①パラメータの数が不足していた場合：エラー。そのラベルを用いたカメラを生成するが、内容は保証されない。
②パラメータの数が過剰の場合：余分なパラメータ記述を無視する。
③同じラベルを用いて、光源が再定義された場合、その光源を含む光源グループを用いた全てのシーンに遡って適用する。

（光源グループ）

［記法］　光源グループ＝LIGHTGROUP(光源ユニット名称, 光源ユニット名称, ...);
光源ユニット名称　　定義済みの光源ユニットの名前を参照する
光源グループを作成する。
　最大８個まで光源ユニットを登録することができる

Ver.2.09において、
①光源の数がゼロだった場合：デフォルト光源を一つ設定
②光源の数が８以上定義された場合：警告を出し、８までを採用
③光源グループの定義において、光源に同じ名称が繰り返し使用された場合：同じ名称の光源ユニットを複数生成する（危険な状況）。

［例］L1 = LIGHTGROUP(L1,L1,L1);

④同じラベルを用いた光源グループの再定義はエラーとして処理する。既に定義済みの光源グループは影響を受けない。但し、それを構成する光源ユニットの内部のパラメータは、光源ユニットの再定義によって変化しうる。

（４）効果 EFFECT, EFFECTGROUP

OpenGL で使用することのできる効果等を記述する。シーンを定義するために必須ではなく、何も定義されていなければ設定されない。効果タイプの後の引数の数に制限はない。現在の段階までで実装している機能には次のようなものがある。

```
EFFECT(STDFOG,int 種別);  種別は、霧の濃度を示す１～５の値
EFFECT(GRID, float 間隔);  平面図等の表示におけるグリッド
EFFECT(MEASURE);    平面図等の表示におけるメジャー
EFFECT(STEREO, float 眼距, int SWAP); ステレオ表示モード
EFFECT(SHADOW, int mode, float 影の長さ);  影の表示を行う。
EFFECT(ANTIAREASING, int value);アンチエリアシング
EFFECT(TRIQUAD); 三角四角高速表示
EFFECT(DISPLIST); ディスプレイ・リストの使用
EFFECT(SIZE,x,y); 画面サイズ（背景・前景画像が存在する場合）
EFFECT(VISUAL,x1,y1,x2,y2,NAME); 可視範囲解析結果の表示
```

［記法］ 効果ユニット名称＝EFFECT(効果タイプ、...);
効果タイプ　　効果の種類の指定
効果ユニットを作成する。

定義した EFFECT は、EFFECTGROUP コマンドの中で指定され、その EFFECTGROUP があるシーンに適用されて初めて意味を持つ。EFFECTGROUP から引用されなかった EFFECT や、シーンに引用されなかった EFFECTGROUP から引用された EFFECT は、システム状態や表示には反映されず、ＬＳＳ－Ｓ形式のファイルのロードが終了し、インタープリタがリセットされた時点で、メモリから削除・解放される。

()の中に記述される効果タイプおよび引数の数、内容関して、形式が守られている限りインタープリタによる制約はなく、エラー処理も行わない。形式的な解析結果が、シーン管理ライブラリ(SML)に渡される。それが意味のないものであれば、実際の動作はない。表示段階で未定義の EFFECT コマンドが検出されれば、その段階でエラー表示が行われるが、データ自体は削除されず、再保存に際してもＬＳＳ-Ｓファイルに反映される。一方、ユーザーが「シャッター」を操作して、ひとつのシーンを更新した場合には、システムが現在の表示状態から自動的に EFFECTGROUP を生成するため、そこに元々あった未定義の EFFECT コマンドは記録されるシーンには含まれない。言いかえると、読み込むファイルに未定義の EFFECT コマンドが含まれていた場合、シャッター操作によるデータの更新が行われない限り、再保存される LSS-S ファイルにもそれは継承される。

例：　E0 = EFFECT();　→空のエフェクトが作成される。動作上意味がない。

E1 = EFFECT(NONSENSE,,,,,,,,,,,,,nonsense,,,,,);
->形式的に解釈し、システムに渡される。

[記法] 効果グループ＝EFFECTGROUP(効果ユニット名称、効果ユニット名称、...);
効果ユニット名称　　効果を指定する
効果グループを作成する。複数の効果を組み合わせて一つのシーンに対して指定することができる。効果が一つしかない場合であっても、EFFECTGROUP コマンドを通じてしか、シーンに対して効果を指定することはできない。

Ver.2.07 以前においては、無視される。必要があれば、シーン切換時の挨拶音や、表示中の BGM 等を指定することに利用できるかも知れない。後述の GROUP 定義における属性と同様、将来拡張性のあるコマンドである。

Ver.2.09 における EFFECT は、全てシステムが自動的に生成する。シーンの編集において、あるシーンをシャッターで確定する時点で、画面に設定されていた表示条件を EFFECT コマンドの形式を用いて、そのシーンのデータに対して自動的に記録する。表示するシーンを切り換えた時点や、LSS-S ファイルをロードした時点で、表示すべきシーンに EFFECT コマンドが設定されていた場合には、それに対応する表示方法に切り替える。EFFECT が設定されていなかった場合には、全ての EFFECT に関連した項目を、デフォルト設定とする。

[Ver.2.09 インタープリタの処理]

①インタープリタは、基本的な文法を守った EFFECT 記述を、意味の有無にかかわらず解釈して、結果をシーン管理ライブラリ(４－２(1)参照) に引き渡す。あるシーンを表示する段階で、未定義の記述が検出されれば、そこでエラーを表示する。定義されたコマンドで引数等に不適切な値が設定されていた場合に、エラー処理するか、適当な値を仮定して処理を続行するか等は、それぞれのエフェクトの処理機構に依存している。

②保存段階では、インタープリタは、シーン毎の効果構造体として渡されたデータをそのまま出力する。Ver.2.09 においては、シャッターを操作する段階で、既存の効果構造体をクリアし、その瞬間における表示に係る各種設定を効果構造体として記述する。従って、表示設定に関係のない効果に係る未定義のコマンドは、この段階でメモリ上の効果構造体から削除され、ファイル保存に際しても出力されなくなる。

③同じラベルを用いた効果、効果グループの再定義はエラーとして処理する。既に定義された効果、効果グループは影響を受けない。

（５）モデル MODEL

地物の幾何形状を記述したＬＳＳ－Ｇファイルを指定する。LSS-S ファイルは、異なるシーンに異なるモデルを指定することにより、シーンを切り換えることにより、物体が置

換・変形したような表現を行うことができる。

［記法］　モデル名称＝MODEL(ＬＳＳ－Ｇファイル名称);
ＬＳＳ－Ｇファイル名称　ＬＳＳ－Ｇファイル
モデルデータを作成する。

異なるモデル名称を用いて同じ LSS-G ファイルが再度参照された場合、あるいはロード済みの LSS-G ファイルが、別の LSS-G ファイルから部品として再度参照された場合、インタープリタは繰り返して同じ LSS-G ファイルをロードせずに、ロード済みのデータを再利用する。

2001 年度に、まちづくり・コミュニケーション・システムの一部として、景観シミュレータの機能拡張を行った。その際に、WEB サーバーから公開されるＬＳＳ－Ｓファイルを、HTML ページからリンクできるようにした。景観シミュレータがセットアップされた環境においては、この操作により景観シミュレータが起動し、ＬＳＳ－Ｓファイルをダウンロードし開く。そのＬＳＳ－Ｓファイルの中では、モデルは、例えば以下のように記述されている。

MDL1 = MODEL("http://sim.nilim.go.jp/niigata/geometry/3ddb/08P0487P211200040003.geo");

この情報に基づき、景観シミュレータは、”http://sim.nilim.go.jp/niigata/geometry/3ddb/”というプレフィックスを用いながら、このＬＳＳ－Ｇファイルと、それが更に引用する部品等のＬＳＳ－Ｇファイルをダウンロードして表示する。

［エラー処理の現状］

①モデルは、MODEL コマンドが実行された時点でロードされる。ロードに失敗した場合には、インタープリタのエラーとして処理する。

②同じラベルを用いたモデルの再定義はエラーとして処理する。定義済みのモデルは影響を受けない。

（６）背景・前景画像 IMAGE

背景・前景画像は写真合成に使用する。写真合成において、背景は、設計対象物や地形を記述した三次元オブジェクト(MODEL)が存在しない画面領域にのみ表示される。前景は、常に表示されるが、アルファ値（不透明度）が１ではない画像領域に関しては、モデルや背景と混合し、アルファ値がゼロの場合は完全に透明となる。写真の中に、三次元オブジェクトよりも手前にある被写体が映っている場合、その部分以外の部分を透明（アルファ値ゼロ）となるように加工した画像を前景として設定する。定義した画像を前景とするか背景とするかは、SCENE コマンド（後述）の引数列内の位置により指定する。よく用いら

れる JPEG ファイルではアルファ値が設定できないため、前景として利用する場合は、アルファ値が設定できる SGI 形式などに変換する必要がある。

［記法］ 画像名称＝IMAGE(画像ファイル名称);
画像ファイル名称　　SGI 形式、JPEG 形式、BMP 形式のファイル名。
画像データを作成する。

［例］ I1 = IMAGE(sample.sgi);

Ver.2.09 における処理
①ファイル名称がフルパスで記載されていない場合には、作業環境が設定されている場合には、その場所を、そこになければ環境設定ファイルで定義された画像ファイル格納ディレクトリから検索する。
②画像ファイルをロードする時点は、IMAGE コマンドが実行される時点である。画像ファイルのロードに失敗すると、インタープリタはエラー処理を行う。
③同じ画像名称を用いた画像の再定義が行われた場合には、エラーとして処理する。
定義済みのイメージは影響を受けない。

（7）シーンの生成 SCENE

以上の各コマンドにより定義した情報を用いて、シーンを合成する。

［記法］ シーン名称＝SCENE(シーン・タイプ、背景画像名称、前景画像名称、
モデル名称、光源グループ名称、効果グループ名称、視点名称、時間名称);

シーン・タイプ　図法、表示モードを指定する３桁のコード（表３－１）
背景画像名称　IMAGE コマンドで定義した画像名称
前景画像名称　IMAGE コマンドで定義した画像名称
モデル名称　　MODEL コマンドで定義したモデル名称
光源グループ名称　　LIGHTGROUP コマンドで定義した光源グループ名称
効果グループ名称　　EFFECTGROUP コマンドで定義した効果グループ名称
視点名称　　CAMERA コマンドで定義した視点名称
時間名称　　TIME コマンドで定義した時間名称
シーンを生成するコマンドは、実行されるたびに、新たなシーンを生成し、メモリ上の配列に蓄積される。景観シミュレータでの表示においては、左下のシーン送り／戻しボタンにより、蓄積されているシーンを次々と表示することができる。
最上位の定義である SCENE においては、ファイル読み込み終了後も、その各種設定を名

称で管理しているため、同じラベルの再定義が行われると、編集操作などによりシーンの順序が変更となった場合などに、定義の解釈が難しくなる。そこで、LSS-S の記述に用いる名称は、同じ名称を再定義するとエラーとして処理する。

規則違反のデータに関して、Ver.2.09 では以下のように対処している。

①引数が不足する場合：エラー

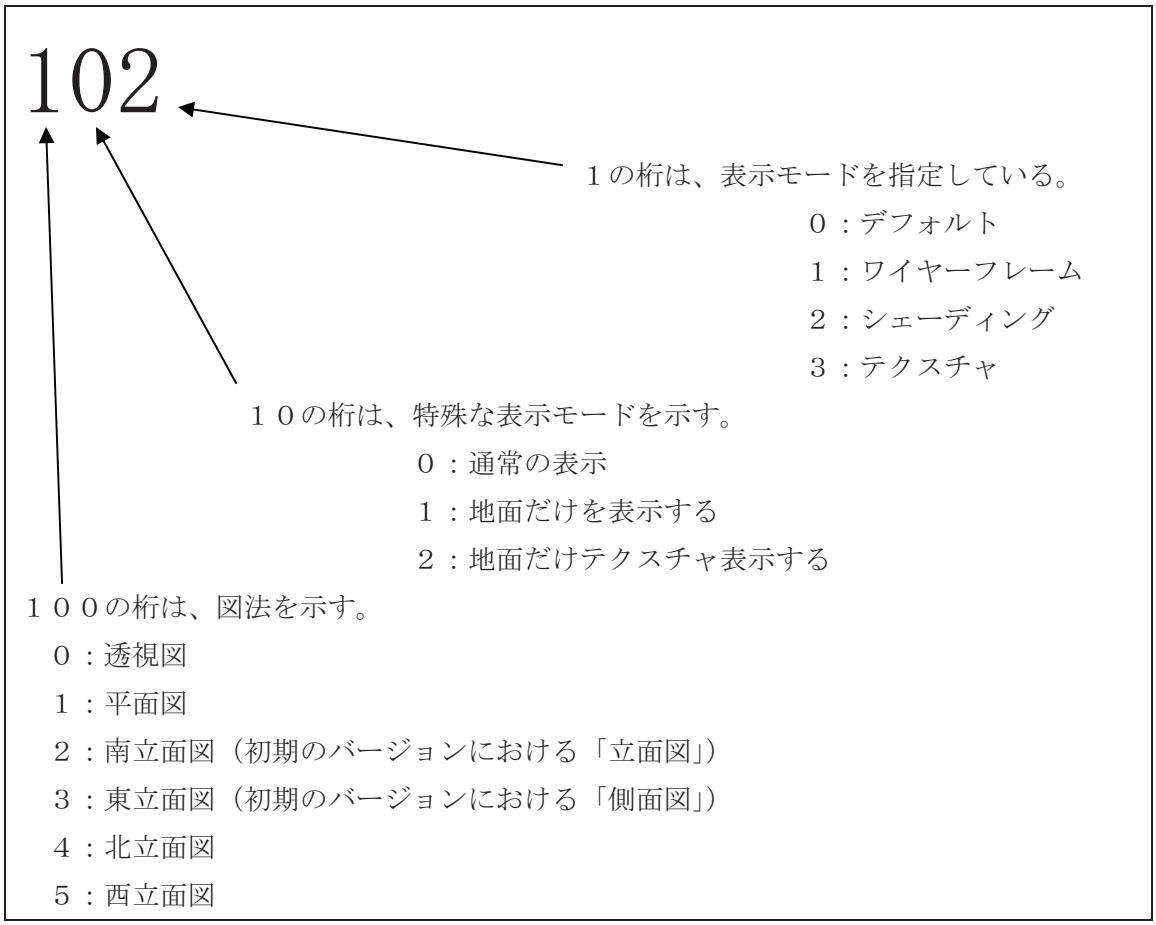
②引数が省略された場合：

1. タイプ：　0 とみなす
2. 背景画像名称：　　背景なし（積極的な意味）
3. 前景画像名称：　　前景なし（積極的な意味）
4. モデル名称：　モデルなし（積極的な意味）
5. 光源グループ名称：光源なし（表示にはデフォルト光源を使用）
6. 効果グループ名称：効果なし（積極的な意味）
7. 視点名称：　エラーではないが、モデルが表示されない。
8. 時間名称：　　エラーメッセージを出し、０を適用。

③同じラベルを用いたシーンの再定義が行われた場合にはエラーとして処理する。

④SCENE コマンドが一つも存在しない場合、エラーとして処理する。

表３－１：シーン・タイプの意味



（8）シーンのファイル出力 OUTPUT_SCENE

シーンの出力を行う。

［記法］ ファイル名称=OUTPUT_SCENE（）；

例） sample＝OUTPUT_SCENE();

３－３ ＬＳＳ－Ｇコマンド

ＬＳＳ－Ｇ（ＬandＳcape Simulation-Geometry）ファイルは、景観構成要素データベースにおける３Ｄ形状データを具現化したファイルである。テキスト・ファイルであり、インタプリタ（ＩＰ、４－２(6)参照）により解釈され、ＤＭＬライブラリ（４－２(2)参照）を介してメモリ上に格納される。

ＬＳＳ－Ｇの拡張子は".geo"である。

表３－２：ＬＳＳ－Ｇファイルに用いられるコマンドの一覧

COORD
NORMAL
TCOORD
COLOR
VERTEX
FACE
LINE
GROUP
LINK
MATERIAL
TEXTURE
FILE
FILE_SUMMIT
FACE_COLOR
FACE_NORMAL
FACE_MATERIAL
FACE_TEXTURE
LINE_COLOR
LINE_NORMAL
LINE_MATERIAL
LINE_TEXTURE
GROUP_FACE

GROUP_MATERIAL
GROUP_TEXTURE
LINK_XFORM
CONCAVE
OUTPUT

３－３－１．データ構築コマンド

（１）座標 COORD

［記法］　座標名称＝COORD(X, Y, Z);

Ｘ　　Ｘ座標値

Ｙ　　Ｙ座標値

Ｚ　　Ｚ座標値

座標データを作成する。

例）　ＶＯＯＯＯ１＝COORD(10.0,20.0,30.0);

（２）法線 NORMAL

［記法］　法線名称＝NORMAL(X, Y, Z);

Ｘ　　法線方向のＸ座標値

Ｙ　　法線方向のＹ座標値

Ｚ　　法線方向のＺ座標値

法線データを作成する。正規化されている必要はない。

例）　ＮＯＯＯＯ１＝NORMAL(10.0,0.0,0.0);

（３）テクスチャ座標 TCOORD

［記法］　テクスチャ座標名称＝TCOORD(S,T);

Ｓ　　テクスチャ座標のＳ値

T　　テクスチャ座標のＴ値

テクスチャ座標データを作成する。

テクスチャ領域は0.0≦ ≦1.0に対応。

例）　ＴＯＯＯＯ１＝TCOORD(0.5,0.5);

テクスチャを面に貼り付ける場合には、面に対して適用するテクスチャの画像ファイル名を定義するだけではなく、各頂点を、画像ファイルのどこに対応づけるかを示すためにテクスチャ座標を指定する必要がある（テクスチャについては、２－９参照)。

（４）色 COLOR

［記法］　色名称＝COLOR(R,G,B,A);

R　　色データの RED 値

G　　色データの GREEN 値

B　　色データの BLUE 値

A　　色データの ALPHA 値

色データを作成する。各値は 0.0≦ ≦1.0 の浮動小数点値とする。

例)　ＣＯＯＯＯ１＝COLOR(1.0,0.5,0.0,1.0);

色は、面や線に対して一括して指定することも、頂点毎に指定することもできる。一つの面や線を構成する点に異なる色を指定すると、その中間部は補間した色となる。

（５）頂点 VERTEX

最大で、座標、法線、テクスチャ、色の４属性[1)]を有する頂点を生成する。同じ位置にあっても、異なる法線ベクトル等をもつ頂点は、別の頂点として定義する必要がある。

[記法]　頂点名称＝VERTEX(座標名称、法線名称、テクスチャ座標名称、色名称);

座標名称　　COORD コマンドで定義された座標名称

法線名称　　ＮＯＲＭＡＬコマンドで定義された法線名称

テクスチャ座標名称　　TCOORD コマンドで定義されたテクスチャ座標名称

色名称　COLOR コマンドで定義された色名称

頂点データを作成する。

例)　ＰＯＯＯＯ１＝VERTEX(V0001,N0001,T0001,C0001);

法線名称、テクスチャ座標名称、色名称は省略可能である。

例)　ＰＯＯＯＯ１＝VERTEX(V0001,,T0001);

例)　ＰＯＯＯＯ１＝VERTEX(V0001);

1)：Ver.2.07 までのインタープリタ（ＩＰライブラリ）では、５以上の属性が定義されていた場合においても、最初の４属性のみを解釈し、以降の属性は無視している。Ver.2.09 においては、仮想線の出発点となる頂点を明記する必要がある場合に、ファイル出力時点で第５引数に、その仮想線の終点の座標名称を第５引数として記述する。入力時点では、面を定義するコマンドにおいて、次の頂点の座標名称が、第５引数で指定された頂点の座標名称と一致する場合には、当該頂点に仮想線の出発点であることを示すフラグを立てる。このフラグが無い場合で、仮想線が存在しうる場合（８頂点以上の面）に関しては、従来通り面の定義が行われた時点で、使用されている頂点名称の比較分析により仮想線を判定する。この属性が意味を有するのは、図形演算が途中で失敗ないし中断し、図形の内部に、外周との接続がなく、面積の無い線状の切り込みが残された状態で、検査等の目的のためにこれをファイル保存する場合である。

（６）面 FACE

[記法]　面名称＝FACE(頂点名称、頂点名称、...);

頂点名称　　VERTEX コマンドで定義された各属性名称

面データを作成する。

例)　ＳＯＯＯＯ１＝FACE(P0001,P0002, P0003);

面は無制限の数の頂点を持つことができる。最後の頂点と最初の頂点は自動的につながれるため、例えばｎ角形の場合にはｎ個の頂点を指定する。最初の頂点と同じ頂点を最後に定義すると、表示は同等であっても長さゼロの辺が一つ余分に定義された「病的な」面が生成するので注意を要する。

面の場合、次に述べる線の場合とは異なり、頂点の順序を巡回的に変更しても、生成される面は同じものとなる。

（７）線 LINE

［記法］ 線名称＝LINE(頂点名称、頂点名称、...);

頂点名称　　VERTEX コマンドで定義された各属性名称

線データを作成する。

例)　ＳＯＯＯ１＝LINE(P0001,P0002, P0003);

線は、無制限の数の頂点を持つ折れ線である。面とは異なり、閉じたループを作る場合には、最後の点と最初の点の頂点は同じものでなければならない。折れ線は、面とは異なり、同一平面上にない場合でも支障はない（例えば、ジェットコースター線路の中心線軌跡)。

（８）グループとその属性の定義 GROUP

［記法］ グループ名称＝GROUP(属性 1,属性２,・・・);

グループを生成する。

属性：属性を記述する文字列。

引数なし　　通常のグループ定義

&GROUND グループに地面属性を付ける

&TREE3　３枚の面で作られた近似的な樹木であることを示し表示を制御

&INFO:文字列情報　（グループを選択したときに情報の表示を行う）

&EXFO:ファイル名　（そのグループを選択したときに、テキスト・ファイルが表示される

&HOUSE:住宅情報　（住宅情報を見る際に表示される構造化された情報）

&PHISI:物理的属性

&CHEMI:化学的属性

&DB:データベース参照情報（景観データベースの登録情報へのタグ）

例)　ＧＯＯＯ１＝GROUP();

同じグループ名称が二度定義されると、属性部分が追加される。例えば、（１２）ＦＩＬＥコマンドを用いて構築されたグループに属性情報を付加する場合には、同じグループ名称を用いたＧＲＯＵＰコマンドを用いて、属性情報を付加することができる。

G01=FILE(物体 1);

G01=GROUP(属性 1);

G01=GROUP(属性 2);

というコマンドが実行された場合、これをファイル保存すると、次のように保存される。

G01=FILE(物体 1);

G01=GROUP(属性 1,属性 2);
というコマンド列として保存される。

（9）リンクの宣言 LINK

二つのグループを関連づける

[記法]　リンク名称＝LINK(グループ名称、グループ名称);

グループ名称（第１引数）　　親となるグループの名称

グループ名称（第２引数）　　子となるグループの名称

グループ間の階層を定義する。第一パラメータで示されるグループが親となり、第二パラメータで示されるグループが子となる。

例)　L0001＝LINK(G0001.G0002);

リンクには、マトリクスが定義されており、リンクが定義された時点では単位マトリクスに初期化されている。続くコマンドでこのマトリクスが変更されると、子グループの位置・スケールが変化する。

同じ親グループと子グループの間には、複数のリンクを定義することができる。それぞれのリンク・マトリクスが異なっていれば、リンクの数だけ子グループが異なる位置に表示される。この機能は、街路樹など、同じ物体を、メモリ消費を増大させることなく、多数配置する場合に有用である。言い換えると、画面に表示されているオブジェクトは、実はリンクを見ている、と理解することができる。

（１０）マテリアルＩＤ　MATERIAL

[記法]　マテリアル ID＝MATERIAL(マテリアル名称);

マテリアル名称　マテリアルファイル（３－６参照）に定義されているマテリアルの名称（ラベル）で、以下その内容に関する記述（カラー値、テクスチャ等）が続く。マテリアル・ファイルは、ＤＢＩＬライブラリの dbLoadMaterial 関数により行われる。

例)　M001 = MATERIAL(SILVER);

（１１）テクスチャＩＤ　TEXTURE

[記法]　テクスチャ ID＝TEXTURE(テクスチャファイル名);

テクスチャファイル名　　テクスチャとして適用する画像ファイル名（３－８参照)。

画像ファイル名がフルパスで指定されている場合を除き、環境設定ファイル kdbms.set(１２－１参照)で指定したディレクトリから取得する。

例)　I001 =TEXTURE (renga) ;

I002 = TEXTURE(renga.sgi);

I003 = TEXTURE(renga.jpg);

I004 = TEXTURE(“D:¥MyProject¥image3.bmp”)

Ver.209 においては、
①拡張子が省略された場合には、.sgi を補う。SGI 形式(RGB)形式はマイナーであるが、景観シミュレータの開発開始頃(1993 年)に利用可能であった、透明度（アルファ値）まで表現しうる唯一の画像であった。構造は RGB または RGBA をバイナリでシリアルに並べただけのシンプルな形式である（３－８参照)。
②テクスチャは、TEXTURE コマンドが実行された時点でロードされる。同じファイルが別のラベルで参照された場合には、同じデータで多重にメモリを消費しないようにバッファリングを行っている。

（１２）外部関数、外部ファイル参照 FILE

[記法] グループ名称＝FILE(ファイル名称、パラメータ、パラメータ、...);

ファイル名称　ＬＳＳ－Ｇファイルまたは、EXT.TAB ファイルに登録されている外部コマンドファイル

パラメータ　EXT.TAB ファイルで定義された引数リスト。ＬＳＳ－Ｇファイルを指定したときは、パラメータはない。

ファイル名称として固定的なＬＳＳ－Ｇファイル名が指定された場合、データファイルを読み込む。引数は無い。

ファイル名称として、EXT.TAB ファイル（３－１０、５－４参照）に登録してある外部関数を指定した場合は、そのコマンドが実行される。その場合引数として指定するパラメータの数と形式は EXT.TAB に定義されている引数と同じでなければならない。

例１）　G0001 = FILE(BRG_ARCH.geo);

例２）　G0001 = FILE(SPHERE, 10.0, 20.0, 30.0, 3.5);

半径 3.5、中心点座標が(10.0, 20.0, 30.0)の球。

パラメトリックな図形を外部関数により作成し、あるいはＬＳＳ－Ｇ形式のファイル(固定的図形）を読み込み、このグループ名称を有するグループを構築する。ＦＩＬＥコマンドに使用するグループ名称が既に使用されているものである場合、エラーとなる。

ファイル名称で参照されたファイル（上記の例１）の場合、BRG_ARCH.geo)の中で使用されているグループ名称（例えば）G0001 と、参照元で直接定義するグループ名称との重複・一致・衝突を避けるために、FILE コマンドで参照されたファイルの中で使用されるグループ名称には、G0001.G0001 のように、親グループ名称をプレフィックスとして追加している。

（１３）ルートグループにおける外部関数、ファイル参照 FILE_SUMMIT

強制的にグループ名プレフィックスを付加しないFILE コマンド。ファイル保存に際して、必要な場合には、通常のＦＩＬＥコマンドに代わり、システムが自動的に使用する。

［記法］ グループ名称＝FILE_SUMMIT(ファイル名称, パラメータ, パラメータ,...);

ファイル名称　ＬＳＳ－Ｇファイルまたは、EXT.TAB ファイルに登録されている外部コマンドファイル

パラメータ　EXT.TAB ファイルで定義された引数リスト。ＬＳＳ－Ｇファイルを指定したときは、パラメータはない。

ファイル名称として固定的なＬＳＳ－Ｇファイル名が指定された場合、データファイルを読み込む。引数は無い。

ファイル名称として、EXT.TAB ファイル（３－１０、５－４参照）に登録してある外部関数を指定した場合は、そのコマンドが実行される。その場合引数として指定するパラメータの数と形式はEXT.TAB に定義されている引数と同じでなければならない。

FILE_SUMMIT コマンドでは、ロードに際して、指定したファイルの中で定義されたグループの名称に関して、親グループ名称のプレフィックスを付けない。

最上位の親グループとして、FILE コマンドで定義されたグループしかないような場合、保存・読み込み等を繰り返す度に無用なプレフィックスが追加されて長いグループ名となってしまうような事態を防ぐために用いる。

例）G0001.G0001.G0001.G0001

３－３－２．データ定義／更新コマンド

面、線関連

（１）色の面への適用　FACE_COLOR

面データに色を割り当てる。

［記法］ FACE_COLOR（面名称、色名称）；

面名称　FACE コマンドで定義された面名称

色名称　COLOR コマンドで定義された色名称

例）　FACE_COLOR（S0001、N0001);

内部処理においてはd3Face 構造体のパラメータを設定する。

（２）法線の面に対する適用 FACE_NORMAL

面データに法線を割り当てる。

［記法］ FACE_NORMAL（面名称、法線名称）；

面名称　FACE コマンドで定義された面名称

法線名称　NORMAL コマンドで定義された法線名称

例）　FACE_NORMAL（S0001、N0001);

内部処理においては d3Face 構造体のパラメータを設定する。

（3）マテリアルの面への適用 FACE_MATERIAL

［記法］ FACE_MATERIAL（面名称、マテリアル ID）；

面名称　　FACE コマンドで定義された面名称

マテリアル ID　MATEREAL コマンドで定義されたマテリアルＩＤ

面にマテリアルを割り当てる。

例）　FACE_MATERIAL（S0001、M0001)；

ＬＳＳ－Ｇコマンドにおいては、面にマテリアルを定義する際に、マテリアル名称（鉄、コンクリート、あるいは色見本帳番号など）を直接指定するのではなく、マテリアルＩＤを一度定義した上で、これを用いて間接指定しているため、単純なケースでは、冗長な定義となっている。

しかし、例えば複雑な形状の壁を有する建物の場合、同一の仕上げの面が多数存在する。これらに対して、同一のマテリアル「M-kabe」を定義しておくと、個別の面へのマテリアル適用に先行して M-kabe を定義している１行を、例えば「M-kabe=MATERIAL(リシン吹付);」から「M-kabe=MATERIAL(レンガタイル);」に変更するだけで、全ての壁の仕上げを一括変更することができる。

内部処理においては d3Face 構造体のパラメータを設定する。

（4）面へのテクスチャの適用　FACE_TEXTURE

［記法］ FACE_TEXTURE（面名称、テクスチャ ID）；

面名称　　FACE コマンドで定義された面名称

テクスチャ ID　TEXTURE コマンドで定義されたテクスチャＩＤ

面にテクスチャを割り当てる。

例）　FACE_TEXTURE（S0001、I0001)；

面にテクスチャを適用する場合、あらかじめ面を構成する頂点の定義(VERTEX コマンド)に際して、テクスチャ座標が定義されていなければならない。

内部処理においては d3Face 構造体のパラメータを設定する。

（5）線への色の適用　LINE_COLOR

線分データに色を割り当てる。

［記法］ LINE_COLOR（線分名称、色名称）；

線分名称　　LINE コマンドで定義された線分名称

色名称　　COLOR コマンドで定義された色名称

例）　LINE_COLOR(LN0001、C0001)；

線分に適用した色は、光源の条件に関わりなく、例えば闇夜であっても、その色で画面

表示される。指定されない場合であっても、明るい灰色で表示される。従って、電線などの細長い物体を線としてモデリングすると、不自然な表示となる場合がある。むしろ、補助線・寸法線などの標示に用いるのに適している。

一方、線分として定義された高架道路などの長尺物の断面を用いて実際の三次元形状を生成する道路生成機能や外部関数「掃引体１面」においては、線に適用されていた色を生成された立体に適用する。このような断面の定義には線に適応した色が有用である。

内部処理においては d3Face 構造体のパラメータを設定する。

（６）線への法線の適用 LINE_NORMAL

［記法］ LINE_NORMAL（線分名称、法線名称）；

線分名称　　LINE コマンドで定義された線分名称

法線名称　　NORMAL コマンドで定義された法線名称

線分データに法線を割り当てる

例）　LINE_NORMAL(LN0001、N0001)；

線に適用された法線は、その線自体を表示する際には意味を持たない。しかし、長尺物の断面を定義する線に法線を定義することは、線が掃引されて生成される面の表裏を区別する上で意味をもつ。

内部処理においては d3Face 構造体（２－２参照）のメンバ変数（法線ベクトル）に値を設定する。

（７）線へのマテリアルの適用 LINE_MATERIAL

［記法］ LINE_MATERIAL（線分名称、マテリアル ID）；

線分名称　　LINE コマンドで定義された線分名称

マテリアル ID　　MATEREAL コマンドで定義されたマテリアルＩＤ

線分データにマテリアルを割り当てる

例）　LINE_MATERIAL(LN0001、M0001)；

線が断面を定義するデータの場合、これを用いて生成した立体に反映される。

内部処理においては d3Face 構造体(２－２参照)のメンバ変数 material を設定する。

（８）線へのテクスチャの適用 LINE_TEXTURE

［記法］ LINE_TEXTURE（線分名称、テクスチャ ID）；

線分名称　　LINE コマンドで定義された線分名称

テクスチャ ID　　TEXTURE コマンドで定義されたテクスチャＩＤ

線分データにマテリアルを割り当てる

例）　LINE_TEXTURE(LN0001、T0001)；

表示に反映するためには､線の各頂点 VERTEX にテクスチャ座標が定義されている必要が

ある。

内部処理においては d3Face 構造体（２－２参照）のメンバ変数 texture を設定する。

（9）グループへの面の割り当て GROUP_FACE

［記法］ GROUP_FACE(グループ名称、面名称、面名称、...);

グループ名称　　GROUP コマンドで定義されたグループ名称

面名称　　FACE コマンドで定義された面名称

例）　GROUP_FACE(G0001,S0001);

一つのグループには無制限個の面を割り当てることができる。グループに既に面が割り当てられている場合には、これに追加される。

実装しているインタープリタにおいては、GROUP_FACE コマンドが実行された後に、その面を構築する際に定義された面の属性､面を構成する頂点､各頂点の属性が変更されても、一度グループに対して割り当てられた面に遡って影響は及ばない。

内部処理においては このコマンドが実行された時点で、インタープリタの解析木の上にそれまでに定義されてきた頂点やカラー等の構成要素を用いて、面を記述するＤＭＬ空間（４－２(7)参照）の d3Face 構造体（メモリ・ブロック）を構築し、これを、ＤＭＬ空間に定義された d3Group 構造体に追加する。この追加は、d3Group 構造体のメンバ変数 f（d3Face*）を起点としてポインタつなぎで定義する､グループに帰属する一群の面のリストの末尾に、この面を加えている。

この面を定義する構成要素がこの時点で確定することから、以後、インタープリタの中で、この面の定義に用いられた各要素の変更・再定義が行われても、ＤＭＬ空間に既に移管・構築された面に遡っては影響が及ばない。

この仕様を利用して、例えば定義済みの面を利用して、それを構成する頂点の座標だけを変更し、再度同じ名称の面としてグループに割り当てることにより、効率的に複数の面を定義することができる。このことはコンバータの作成等においてきわめて有用である。

（１０）グループへの線分の割り当て GROUP_LINE

［記法］ GROUP_LINE(グループ名称、線分名称、線分名称、...);

グループ名称　　GROUP コマンドで定義されたグループ名称

線分名称　　LINE コマンドで定義された線分名称

例）　GROUP_LINE(G0001,LN0001,LN0002);

機能や仕様は、GROUP_FACE コマンドと同等である。グループに面と線を混在させて割り当てることができる。

（１１）グループへのマテリアルの定義 GROUP_MATERIAL

［記法］ GROUP_MATERIAL(グループ名称、マテリアル ID);

グループ名称　GROUP コマンドで定義されたグループ名称

マテリアル ID　MATEREAL コマンドで定義されたマテリアルＩＤ

例)　GROUP_MATERIAL(G0001,M0001);

このマテリアルは､グループ内のデフォルトのマテリアルとなる｡グループに属する面、このグループにリンクする子グループにマテリアルが定義されていない場合、このマテリアルが適用される。

内部処理においては d3Group 構造体のパラメータを設定する。グループは、DML 空間に既に登録されているが、そのグループへのリンク（ポインタ）はインタープリタの解析木に残されているので、これを用いてグループの属性を変更することができる。

グループを定義している d3Group 構造体のメンバである int material メンバに ID 番号として登録される。実際のマテリアルのデータは、インタープリタがこのコマンドを実行するに際して、d3RegisterMaterial(マテリアル名称)関数を起動することにより、DML 空間のスタティックなテーブル mtltab に登録される。但し、この段階では、マテリアルの名称だけが登録されており、実際のデータはまだ登録されていない。実際にマテリアルのロードが行われるのは表示段階である。一度ロードされたマテリアルは、このテーブルから取得されることとなる。なお、マテリアルには経年変化するものがあることから、時間が変更された場合には、mtltab は見直される。

（１２）グループへのテクスチャの定義 GROUP_TEXTURE

グループにテクスチャを割り当てる。

［記法］ GROUP_TEXTURE(グループ名称、テクスチャ ID);

グループ名称　GROUP コマンドで定義されたグループ名称

テクスチャ ID　TEXTURE コマンドで定義されたテクスチャＩＤ

例)　GROUP_TEXTURE(G0001,I0001);

このテクスチャは、グループ内のデフォルトのテクスチャとなる。

内部処理においては d3Group 構造体のパラメータを設定する。グループは、DML 空間に既に登録されているが、そのグループへのリンク（ポインタ）はインタープリタの解析木に残されているので、これを用いてグループの属性を変更することができる。

グループを定義している d3Group 構造体のメンバである int texture メンバに ID 番号として登録される。実際のテクスチャのデータは、インタープリタがこのコマンドを実行するに際して、d3RegisterTexture(テクスチャ名称)関数を起動することにより、DML 空間のスタティックなテーブル textab に登録される。但し、この段階では、テクスチャの名称だけが登録されており、実際のデータはまだ登録されていない。実際のテクスチャのロードは表示段階で g3LoadMateiral によって果たされる。一度ロードされたテクスチャは、このテーブルから取得されることとなる。

（１３）リンク・マトリクスの定義 LINK_XFORM

リンクに変換マトリクスを定義する。

［記法］ LINK_XFORM(リンク名称、変換タイプ、変換名称、パラメータ、...);

変換タイプ

LOAD　それまでのマトリクスを初期化する。

PRE　それまでのマトリクスの前に乗ずる。

POST　それまでのマトリクスの後に乗ずる。

変換名称及びそれに付随するパラメータは以下の通り

IDENTITY　パラメータ=なし（対角だけ１の単位マトリクス）

TRANSLATE　パラメータ=X、Y、Z（平行移動）

ROTATE_X　パラメータ=X 軸周り回転角(deg)

ROTATE_Y　パラメータ=Y 軸周り回転角(deg)

ROTATE_Z　パラメータ=Z 軸周り回転角(deg)

ROTATE_A　パラメータ=回転軸ベクトル（X、Y、Z)、回転角

SCALE　パラメータ=X、Y、Z

MATRIX　パラメータ=行列要素 0、1、...、１５

例）　LINK_XFORM（L001、LOAD、ROTATE_X､30.0）

LINK_XFORM コマンドが実行されるまでは、リンクが定義された時点で単位マトリクスが定義されている。

マトリクスの詳細については、４－２（３）マトリクス／ベクトル関数の、d3SetLinkMatrix 関数の項を併せて参照されたい。上記の MATRIX により１６個の倍精度実数値で記述する場合には、

0　4　8 12

1　5　9 13

2　6 10 14

3　7 11 15

の順でマトリクスの要素を記述し、変換されるベクトルは列ベクトルである。頂点座標の変換計算は、XYZ に１を加えた同次座標に変換マトリクスを乗じることにより、回転と移動を一挙に計算する。法線ベクトルについては、平行移動はないため、３×３の部分だけが計算に用いられるのでなければならない、ということはデバッグ、検査に際して留意すべき点である。

（１４）凹ポリゴンの指定 CONCAVE

［記法］ CONCAVE(パラメータ);

ＯＮ　凹多角形として表示処理を行う

ＯＦＦ　凹多角形表示処理を行わない

凹多角形処理の開始、終了を指示する（２－２（６）参照)。
例)　CONCAVE(ON);
CONCAVE スイッチが ON になっている間、インタープリタは、入力されメモリ空間に構築する面に凹ポリゴンの属性(D3_SHP_CONCAVE)を付ける。表示段階では、この属性が付いた面に関しては、三角形に分割して表示を行うため、無い筈の場所に現れる意味のない黒影で惑わされることが防がれる。

（１５）グループのファイル出力 OUTPUT

データファイルに書き込む
［記法］ ファイル名称=OUTPUT（グループ名称）;
例)　sample＝OUTPUT(G0001);
指定されたグループ名称から下位の情報が書き込まれる

３－４．　共通コマンド

LSS-G と LSS-S に共通に用いられる、システム的なコマンドである。

表３－３：共通のコマンド

CLEAR
DELETE
RESET
COMMENT
UNIT

（1）インタープリタのリセット RESET

インタープリタ空間をリセットし、ラベルの辞書等に占有していたメモリを解放する。
［記法］ RESET();

（2）データの初期化 CLEAR

未実装（廃止)。1998 年の仕様書には記されていたが、使われたことがなく、今後も使う予定はない。

（3）データの削除 DELETE

［記法］DELETE（*名称）;
名称で指定されたデータを削除する
例)　DELETE（S0001); //この例では面 S0001 を削除する
例)　DELETE（G0001); //この例ではグループ G0001 を削除する
例)　DELETE（L0001); //この例ではリンク L0001 を削除する

（4）注釈領域の始まりと終わり COMMENT

コメントの開始、終了を指定する。

[記法] COMMENT（パラメータ）；

ON　　コメントの開始

OFF　　コメントの終了

COMMEN(ON)以降はCOMMENT(OFF)が現れるまでコメントとして扱う。ファイル中、情報としては残しておたいが、表示には使用したくないようなデータに用いる。システム開発者が、モデリング機能やコンバータの開発途上でのテスト・デバッグにおいて、データと表示の対応を検査する際に有用である。

例)　COMMENT(ON);

（5）長さ精度の単位 UNIT

座標値の精度、即ち最小単位を、メートルを尺度として指定する。単位よりも小さな値は四捨五入される。従って、データの誤差の下限を示している。単位よりも小さい距離にある二つの異なる点は、同じ点として理解される。図形の内外判定において、単位は線の太さを示し、線上の判定に用いられる。例えば1,000ｍを指定すると、線の太さは１km となり、多くの建物は点に包摂される。

[記法] UNIT（長さの単位）；

（長さの単位）；長さの最小単位を指定する。0.001＝1mm がデフォルト有効である。

例)　UNIT(1e-3)

３－５　ＬＳＳデータ構築の実際

（1）ＬＳＳ－Ｓファイル

ＬＳＳ－Ｓデータを構築するには、まず、シーンコマンドで設定するのに必要な各要素（時間、視点、光源、光源グループ、モデル、イメージ）の設定をはじめに行う。ただし、すべての要素を設定する必要はなく、最低限必要な要素は、時間・視点・光源・光源グループである。

①┌まず、必須である要素の設定を最初に行う。

リスト３－４：必須要素の設定

```
例)
time = TIME( 1 );
camera = CAMERA(
        15, 105, 18,
        28, 10, 18,
        0, 0, 1,
        60, 1.566, 1, 1000);
```

```
l = LIGHT(
        0,
        20, 10, 7, 0,
        1, 1, 1);
lg = LIGHTGROUP( l );
```

②次に、モデルやイメージデータがある場合は、それらの設定を行う。

リスト３－５：モデルとイメージデータの設定

```
例)
img = IMAGE(haikei1.sgi);
model1 = MODEL(BRG_ARCH.geo);
```

③最後に、シーンとして先程設定した要素を割り付ける。

リスト３－６：シーンへの各要素の割り付け

```
例)
scene_1 = SCENE( 0, img, , model1, lg, , camera, time );
```

これで少なくとも一つのシーンを持った、ＬＳＳ－Ｓファイルとなる。

④ひとつのＬＳＳ－Ｓファイルの中に、複数の要素、複数のシーンデータを設定する事ができる。

リスト３－７：複数シーンの設定

```
例)
time = TIME( 1 );
camera1 =CAMERA(
        15, 105, 18,
        28, 10, 18,
        0, 0, 1,
        60, 1.566, 1, 1000);
camera2 =CAMERA(
        20, 80, 30,
        28, 10, 18,
        0, 0, 1,
        60, 1.566, 1, 1000);
l = LIGHT(      0,
        20, 10, 7, 0,
```

```
        1, 1, 1);
lg = LIGHTGROUP( 1 );
img = IMAGE(haikei1.sgi);
model1 = MODEL(BRG_ARCH.geo);

scene_1 = SCENE( 0, img, , model1, lg, , camera1, time );
scene_2 = SCENE( 0, img, , model1, lg, , camera2, time );
```

この例では、視点情報を２つ設定している。これを用い、シーンも２つ定義し、それぞれに違う視点情報を設定している。つまり、背景や光源情報、時間情報、モデル情報は同一だが、視点情報だけ異なる２つのシーンが設定されたことになる。プレゼンテーションにおいては、シーンを切り換えることにより、予め設定してある視点場から別の視点場へと景観表示をすばやく切り換えることができる。

あるいは、上の例と同様に、視点情報以外の要素でも複数設定できる。例えば、異なるモデルを同じ視点から見たシーンを用意しておき、現況と建て替え後の、同じ視点から見た比較表示を行うことができる。

（２）ＬＳＳ－Ｇファイル

①グループの作成

ＬＳＳ－Ｇデータの形状定義は、大きく４段階から成る。

座標設定（図４）→面設定（図３）→グループ設定（図２）→リンク設定（図１）

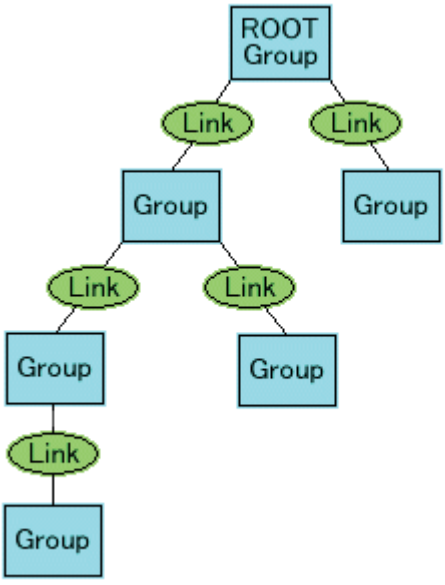
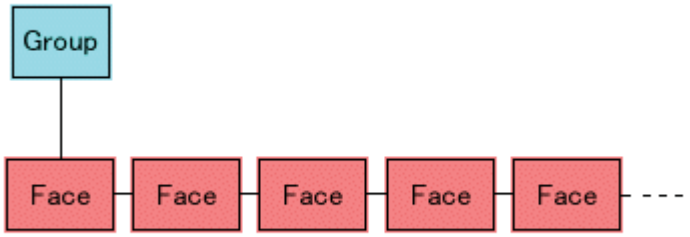
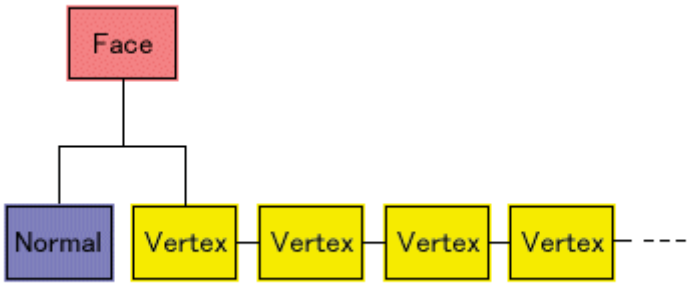


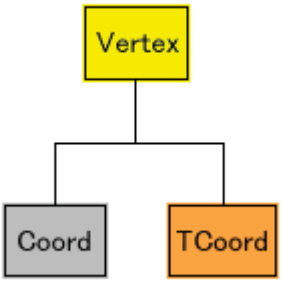
図３－１：リンク設定



·図３－２：グループ設定



・図３－３：面設定



・図３－４：座標設定

最も簡単な例を挙げながら、各設定の流れを説明する。

例）立方体を作る

①まず、座標を任意の座標名称に設定する。

立方体は、8 頂点あるので８個分設定する。

リスト３－８：頂点座標の設定

```
v01=COORD(0, 0, -5);
```

```
v02=COORD(5, 0, -5);
v03=COORD(5, 5, -5);
v04=COORD(0, 5, -5);
v05=COORD(0, 0, -10);
v06=COORD(5, 0, -10);
v07=COORD(5, 5, -10);
v08=COORD(0, 5, -10);
```

②次に、先程設定した座標名称を使って頂点を設定する。

リスト３－９：頂点の設定

```
p01=VERTEX(v01);
p02=VERTEX(v02);
p03=VERTEX(v03);
p04=VERTEX(v04);
p05=VERTEX(v05);
p06=VERTEX(v06);
p07=VERTEX(v07);
p08=VERTEX(v08);
```

この場合、テクスチャ座標等がないため、VERTEX は座標値しかもたない。

③先程設定した頂点名称を使って面データを設定する。

面数は６面なので６面分設定し、面は頂点が４つなので４つの頂点名称を設定する。

リスト３－１０：面の設定

```
s01=FACE(p01, p02, p03, p04);
s02=FACE(p02, p06, p07, p03);
s03=FACE(p06, p05, p08, p07);
s04=FACE(p05, p01, p04, p08);
s05=FACE(p04, p03, p07, p08);
s06=FACE(p05, p06, p02, p01);
```

④ここで、法線を設定する。

６面あるので、６つの法線を設定する。

リスト３－１１：法線の設定

```
n01=NORMAL(1, 0, 0);
n02=NORMAL(-1, 0, 0);
n03=NORMAL(0, 1, 0);
n04=NORMAL(0, -1, 0);
n05=NORMAL(0, 0, 1);
n06=NORMAL(0, 0, -1);
```

⑤先程設定した各面に法線名称を登録する。

リスト３－１２：面への法線の登録

```
FACE_NORMAL(s01, n05);
FACE_NORMAL(s02, n01);
FACE_NORMAL(s03, n06);
FACE_NORMAL(s04, n02);
FACE_NORMAL(s05, n03);
FACE_NORMAL(s06, n04);
```

⑥グループを設定する。

リスト３－１３：グループの設定

```
g01=GROUP();
```

⑦そのグループに先程設定した面データを登録する。

６面体なので、６面登録する。

リスト３－１４：グループへの面の登録

```
GROUP_FACE(g01, s01, s02, s03, s04, s05, s06);
```

⑧以上で、一つのグループが出来た。

これをファイルに保存し、景観シミュレータで表示すると以下のように表示される。

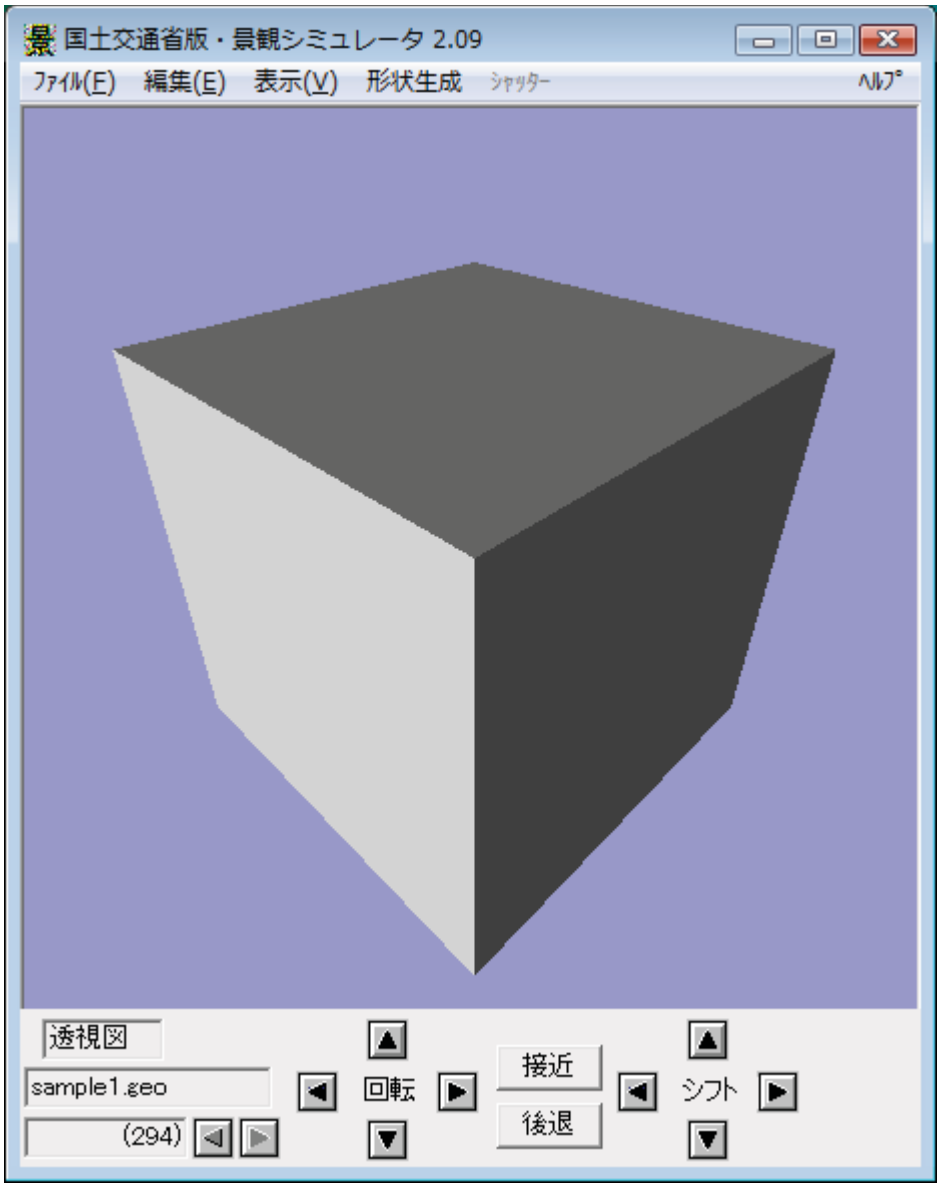


図３－５：６面から成る立方体の表示

（2）マテリアル

今度は、このデータにマテリアルを設定する。

①マテリアルを設定したい物体に適用するに先立って、予め MATERIAL コマンドを使って、マテリアル ID を定義しておく。

リスト３－１５：マテリアルＩＤの定義

```
m01 = MATERIAL(RED);
m02 = MATERIAL(WHITE);
```

②面にマテリアルを付けたい場合は、以下のように FACE_MATERIAL コマンドで登録する。

リスト３－１６：マテリアルＩＤの面への登録

```
FACE_MATERIAL(s01, m01);
FACE_MATERIAL(s02, m02);
FACE_MATERIAL(s03, m02);
FACE_MATERIAL(s04, m02);
FACE_MATERIAL(s05, m02);
FACE_MATERIAL(s06, m02);
```

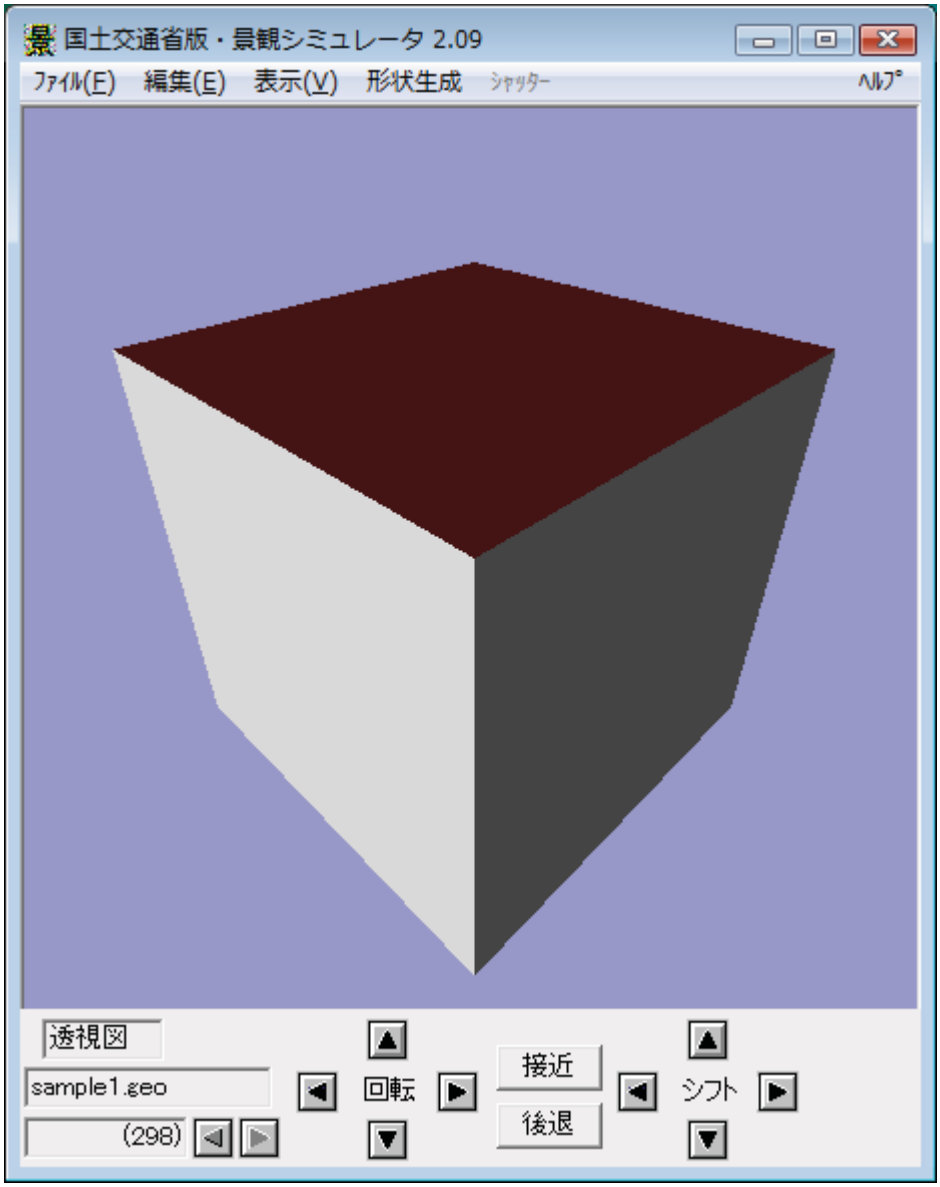


図３－６：面にマテリアルを設定する

③グループにマテリアルを付けたい場合は、以下のように GROUP_MATERIAL コマンドで設定する。

リスト３－１７：マテリアルＩＤのグループへの登録

```
GROUP_MATERIAL(g01, m01);
```

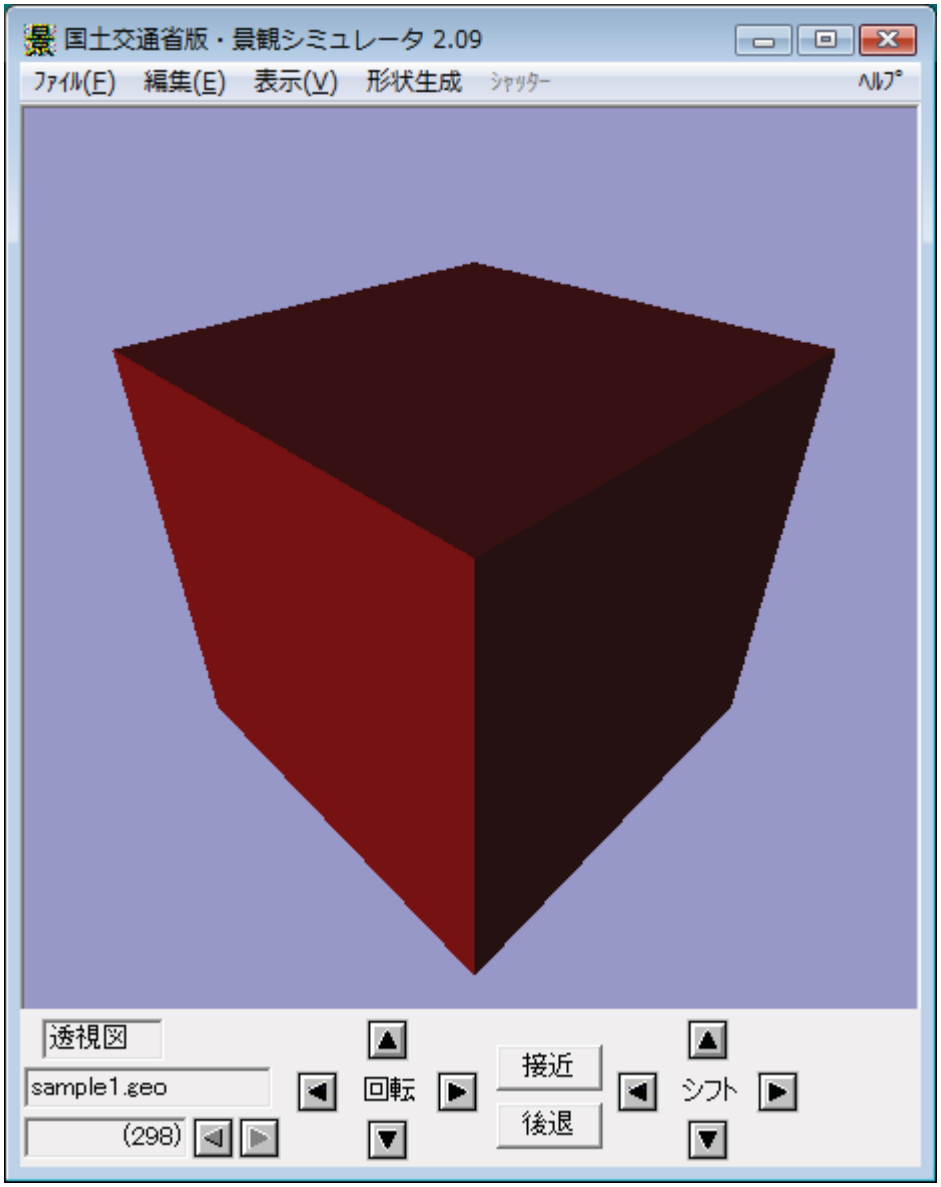


図３－７：グループへのマテリアルの設定

（４）テクスチャ

今度は、このデータにテクスチャを設定する。

①┌グループを設定する。

リスト３－１８：グループの設定

```
g01 = GROUP();
```

②座標、テクスチャ座標を１面分（４点分）設定する（この面にテクスチャを貼りつける）。

リスト３－１９：頂点座標とテクスチャ座標の設定

```
V000000 = COORD(0.000000, 0.000000, -10.000000);
T000000 = TCOORD(0.000000, 0.000000);
V000001 = COORD(5.000000, 0.000000, -10.000000);
T000001 = TCOORD(5.000000, 0.000000);
V000002 = COORD(5.000000, 0.000000, -5.000000);
T000002 = TCOORD(5.000000, 5.000000);
V000003 = COORD(0.000000, 0.000000, -5.000000);
T000003 = TCOORD(0.000000, 5.000000);
```

③先程設定した、座標名称・テクスチャ座標名称を使って、頂点を設定する。

リスト３－２０：頂点の設定

```
P000000 = VERTEX(V000000, , T000000, );
P000001 = VERTEX(V000001, , T000001, );
P000002 = VERTEX(V000002, , T000002, );
P000003 = VERTEX(V000003, , T000003, );
```

④頂点名称を使って面を設定する。

リスト３－２１：面の設定

```
F000000 = FACE(P000000, P000001, P000002, P000003);
```

⑤法線を設定する。

リスト３－２２：法線の設定

```
N000004 = NORMAL(0.000000, -1.000000, 0.000000);
```

⑥面に法線を登録する。

リスト３－２３：面への法線の登録

```
FACE_NORMAL(F000000, N000004);
```

⑦テクスチャＩＤを設定する。

リスト３－２４：テクスチャＩＤの設定

```
I000004 = TEXTURE("renga_1.sgi");
```

⑧面にテクスチャＩＤを登録する。

リスト３－２５：面へのテクスチャＩＤの登録

```
FACE_TEXTURE(F000000, I000004);
```

⑨この面をグループに登録する。

リスト３－２６：面のグループへの登録

```
GROUP_FACE(g01, F000000);
```

これで１面分を作成し、グループに登録した。以降も同様に１面ずつ作成して、グループに登録する。またこの時、同じ頂点名称（VERTEX）を使用したまま、中味の情報を更新する。更新しても今までこの頂点名称を用いて設定し、GROUP_FACE を実行した面には影響

しない。

一度設定した頂点名称を各面で共用した場合、例えば、テクスチャ座標が設定された頂点座標を共用した場合、テクスチャを貼り付けたくない面にまでテクスチャ座標が設定されてしまい、そのデータを表示した時、他の面にまでテクスチャが貼られてしまうという現象が出る。そこで、同じ名称の情報を、テクスチャの無い座標値だけの頂点定義に更新して使用する（⑩の例）か、別の頂点名称に新たな情報を設定してその名称の方を使用する事により、この現象を回避できる。

⑩以降の面は、テクスチャを貼り付けない設定を行う。

リスト３－２７：テクスチャのないその他の面の設定

```
g01 = GROUP();

V000000 = COORD(0.000000, 5.000000, -5.000000);
V000001 = COORD(5.000000, 5.000000, -5.000000);
V000002 = COORD(5.000000, 5.000000, -10.000000);
V000003 = COORD(0.000000, 5.000000, -10.000000);
P000000 = VERTEX(V000000, , , );//テクスチャ座標の定義をしない
P000001 = VERTEX(V000001, , , );
P000002 = VERTEX(V000002, , , );
P000003 = VERTEX(V000003, , , );
F000001 = FACE(P000000, P000001, P000002, P000003);
N000004 = NORMAL(0.000000, 1.000000, 0.000000);
FACE_NORMAL(F000001, N000004);
GROUP_FACE(g01, F000001);

V000000 = COORD(0.000000, 0.000000, -10.000000);
V000001 = COORD(0.000000, 0.000000, -5.000000);
V000002 = COORD(0.000000, 5.000000, -5.000000);
V000003 = COORD(0.000000, 5.000000, -10.000000);
P000000 = VERTEX(V000000); //二番目以降のパラメータを省略
P000001 = VERTEX(V000001);
P000002 = VERTEX(V000002);
P000003 = VERTEX(V000003);
F000002 = FACE(P000000, P000001, P000002, P000003);
N000004 = NORMAL(-1.000000, 0.000000, 0.000000);
FACE_NORMAL(F000002, N000004);
```

```
GROUP_FACE(g01, F000002);

V000000 = COORD(5.000000, 0.000000, -10.000000);
V000001 = COORD(0.000000, 0.000000, -10.000000);
V000002 = COORD(0.000000, 5.000000, -10.000000);
V000003 = COORD(5.000000, 5.000000, -10.000000);
//VERTEX の定義を省略（定義済み）
F000003 = FACE(P000000, P000001, P000002, P000003);
N000004 = NORMAL(0.000000, 0.000000, -1.000000);
FACE_NORMAL(F000003, N000004);
GROUP_FACE(g01, F000003);

V000000 = COORD(5.000000, 0.000000, -5.000000);
V000001 = COORD(5.000000, 0.000000, -10.000000);
V000002 = COORD(5.000000, 5.000000, -10.000000);
V000003 = COORD(5.000000, 5.000000, -5.000000);
N000004 = NORMAL(1.000000, 0.000000, 0.000000);
FACE_NORMAL(F000003, N000004);//F000003 を流用し、面の定義を省略
GROUP_FACE(g01, F000003);

V000000 = COORD(0.000000, 0.000000, -5.000000);
V000001 = COORD(5.000000, 0.000000, -5.000000);
//座標値が変化しない頂点に関して座標の定義を省略
V000003 = COORD(0.000000, 5.000000, -5.000000);
GROUP_FACE(g01, F000003);//面に垂直な法線を省略
```

上記の、テクスチャを貼らない面の定義において、順次記述を省略する方法を例示した。座標値やテクスチャ座標、その他の属性設定は、GROUP_FACE コマンドの実行により、確定する。以後、それ以前に定義したラベルを再定義しても、確定した面の定義は変化しない。このことを利用して、COORD から始めて FACE を構築するのに用いた以前の諸定義を、いわばテンプレートのように活用し、変化した数値（例えば座標値）だけを書き変えて、同じ面を用いて GROUP_FACE コマンドを実行することにより、効率的にデータを構築することができる。このことは、ファイル・コンバータを開発する際に有用である。

データのパターンに応じて、LSS－G ファイルの冒頭部分にいくつかのテンプレートとなる面を定義しておき、変化する部分だけを更新しながら、GROUP_FACE を繰り返していくと

いう方法が、コンパクトなデータに変換するために、よく用いられている。

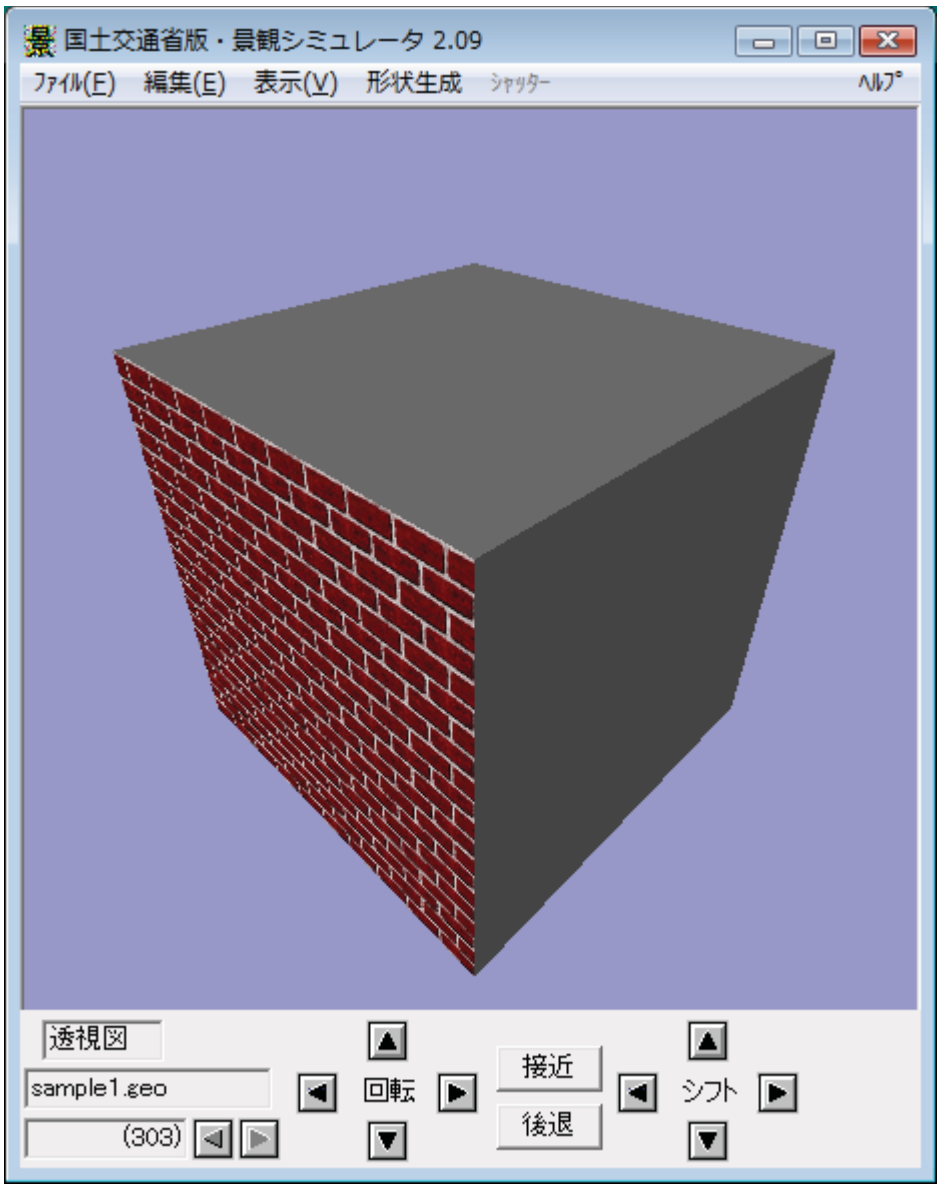


図３－８：一面だけにテクスチャを貼った立方体

（５）グループの作成（FILE コマンド）

グループを作成するもう１つの方法について説明する。

①グループを作成する方法は、以上のような作成方法の他に、すでにあるＬＳＳ－Ｇファイルを読み込んで一つのグループとして扱う方法がある。

リスト３－２８：ＦＩＬＥコマンドによるグループの設定

```
g10 = FILE(BRG_ARCH.geo);
```

FILE()コマンドは、group()コマンドの機能を一部含んでいるので、FILE()コマンドの前に

```
g10 = GROUP();
```

という設定をしておく必要はない。

（6）リンクの設定

今までは、グループを１つだけ定義した。次に、複数のグループを設定し、親子関係を設定する方法について説明する。

①┌グループを複数設定する。

リスト３－２９：リンクする複数グループの設定

```
g01 = GROUP();
g02 = GROUP();
g03 = GROUP();
g04 = GROUP();
g05 = GROUP();
```

②g01 の子供として g02 を登録する。

リスト３－３０：g01 を g02 の親とするリンクの設定

```
L000000 = LINK(g01, g02);
```

③g01 の子供として g03 を登録する。

リスト３－３１：g01 を g03 の親とするリンクの設定

```
L000001 = LINK(g01, g03);
```

④g02 の子供として g04 を登録する。

リスト３－３２：g02 を g04 の親とするリンクの設定

```
L000002 = LINK(g02, g04);
```

⑤g02 の子供として g05 を登録する。

リスト３－３３：g02 を g05 の親とするリンクの設定

```
L000003 = LINK(g02, g05);
```

以上のようなリンクを設定すると、以下のような親子関係となる。

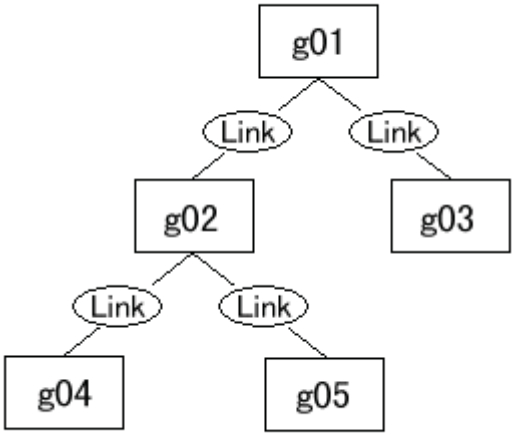


図３－９：リンク

これらのリンクによる親子の座標関係は、子の座標系が、グローバル座標系ではなく、親の座標系に準拠する。つまり、親の座標系の原点と子供の座標系の原点が一致する。

また、このリンクに変換マトリクスを定義する事も出来る。この場合、親の座標系に対する子の座標系の変換（移動・回転・拡大縮小）がマトリクスにより記述される。親がグローバル座標に関して、最上位グループとのリンク・マトリクスにより移動した場合、子はこれに付いて動く。リンクが設定されただけで、リンク・マトリクスが明示的に定義されていない場合には、デフォルト値として単位マトリクス（移動・回転・拡大縮小なし）が設定されている。

リスト３－３４：リンク・マトリクスの設定

```
例)
    LINK_XFORM(L000000, LOAD, MATRIX,
            0.707107, 0.707107, 0.000000, 0.000000,
           -0.707107, 0.707107, 0.000000, 0.000000,
            0.000000, 0.000000, 1.000000, 0.000000,
            2.500000, 2.500000, 0.000000, 1.000000);
```

LINK_XFORM コマンドの詳細に就いては、＜３－３．ＬＳＳ－Ｇコマンド＞で解説する。

（７）異なる設定が同時に行われた場合の優先順位

マテリアルおよびテクスチャの優先順位について説明する。

①┌同一属性が異なるレベルで設定された場合の優先順位

マテリアルおよびテクスチャは、面・グループそれぞれに設定できる。また、カラーと法線は、頂点・面それぞれに設定できる。

これらは、下位の方（マテリアル・テクスチャ：面、法線：頂点）の設定が、優先順位

が高く処理される。つまり、マテリアルの例を挙げると、面・グループのどちらにも同時に設定されている時には、面の設定が優先され有効となる。

②異なる属性が同一レベルで設定された場合の優先順位

面に色を設定する方法には、MATERIAL と COLOR の 2 種類がある。

カラーの定義を含むマテリアルとカラーが同じ面に同時に定義された場合、カラーが優先される。ただし、このことはマテリアル定義の内の色(RGB)についてのみであって、マテリアルの中にテクスチャ、輝度、反射率等が別途定義されている場合には、それらはそのまま適用され、カラーだけが無視されて、別途単独で COLOR 設定された色情報が用いられる。

また、MATERIAL の中には、テクスチャを含むものが存在する。同じグループまたは面に対して、テクスチャ付きマテリアルとテクスチャが同時に設定された場合、マテリアルのテクスチャが優先される。

これらのコマンドを上手に使う事により、より本物に近い質感で表示をさせる事ができる。例えば、TEXTURE コマンドで材質感を表現し、COLOR コマンドで色を微調整するといった使い方である。

（8）マテリアル・テクスチャのデフォルト値と継承

マテリアル・テクスチャが定義されていない場合には、デフォルトのマテリアル（色が{0.8,0,8,0.8,1.0}で、テクスチャその他の設定なし）が適用される。

親グループにマテリアル･テクスチャが定義された場合､そのグループ自身に属する面、これとリンクで結ばれた子グループ（更にその下の孫グループ以下)、及びその子グループに属する面に継承され､それぞれにマテリアル･テクスチャが定義されなかった場合のデフォルトの設定となる。

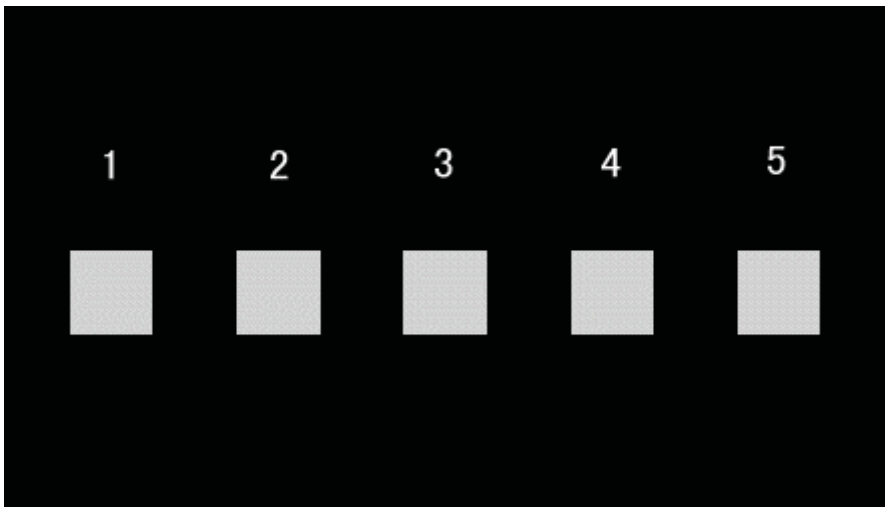


図３－１０：デフォルト状態

マテリアル・テクスチャが設定されない場合、デフォルト設定で表示される。

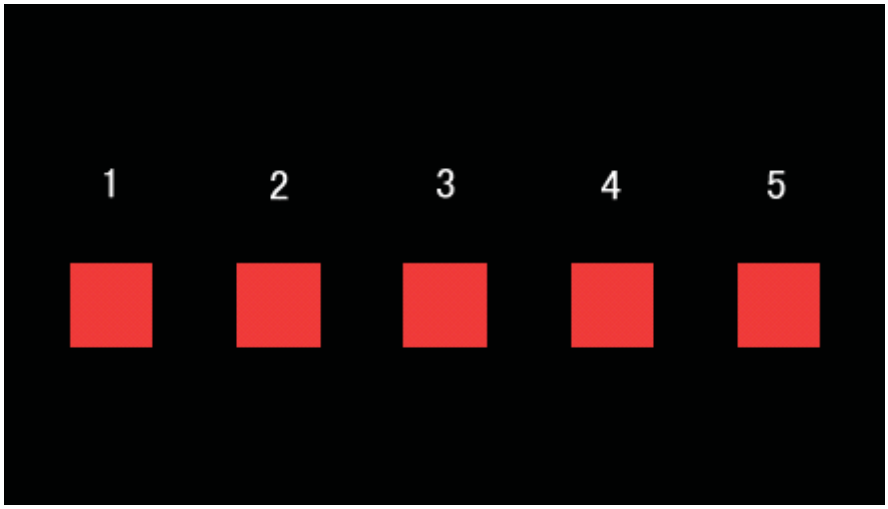


図３－１１：親への設定

親グループにマテリアル「RED」が設定されると、その下に属する、マテリアル等が特記されていない一般の面や子グループ全てに影響が及ぶ。

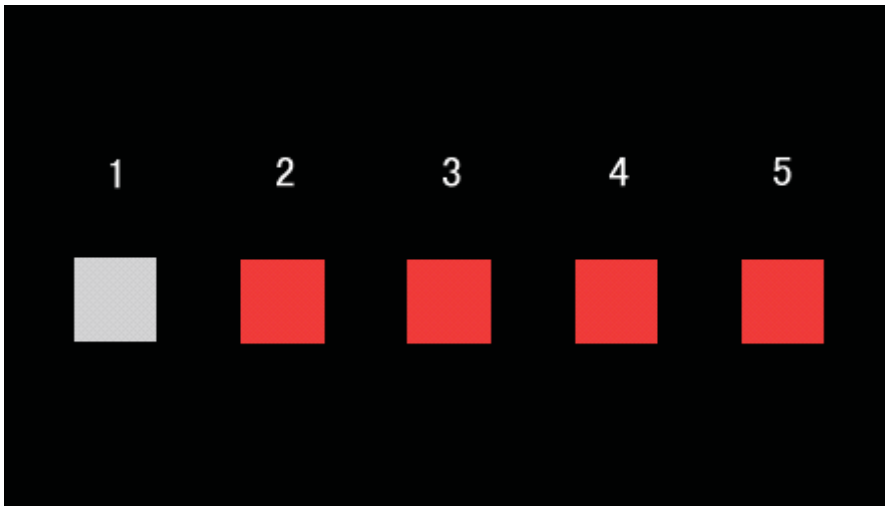


図３－１２：ユニークな設定

親グループから継承されたマテリアル「RED」が存在している状況下で特定の面またはグループ１だけ別のマテリアル「WHITE」を設定すると、その面だけユニークなマテリアルとなり、それ以外の面は親から継承した設定となる。

３－６．ファイル参照とリンクを併用したデータ構築

サンプルデータ１と、サンプルデータ２（これらは長いためデータ自体のリストは省略する）を FILE コマンドで参照して組み合わせることにより、サンプルデータ３を構成した例を示す。

（１）サンプルデータ１：街灯の支柱部分

・　　005_1.geo 表示例

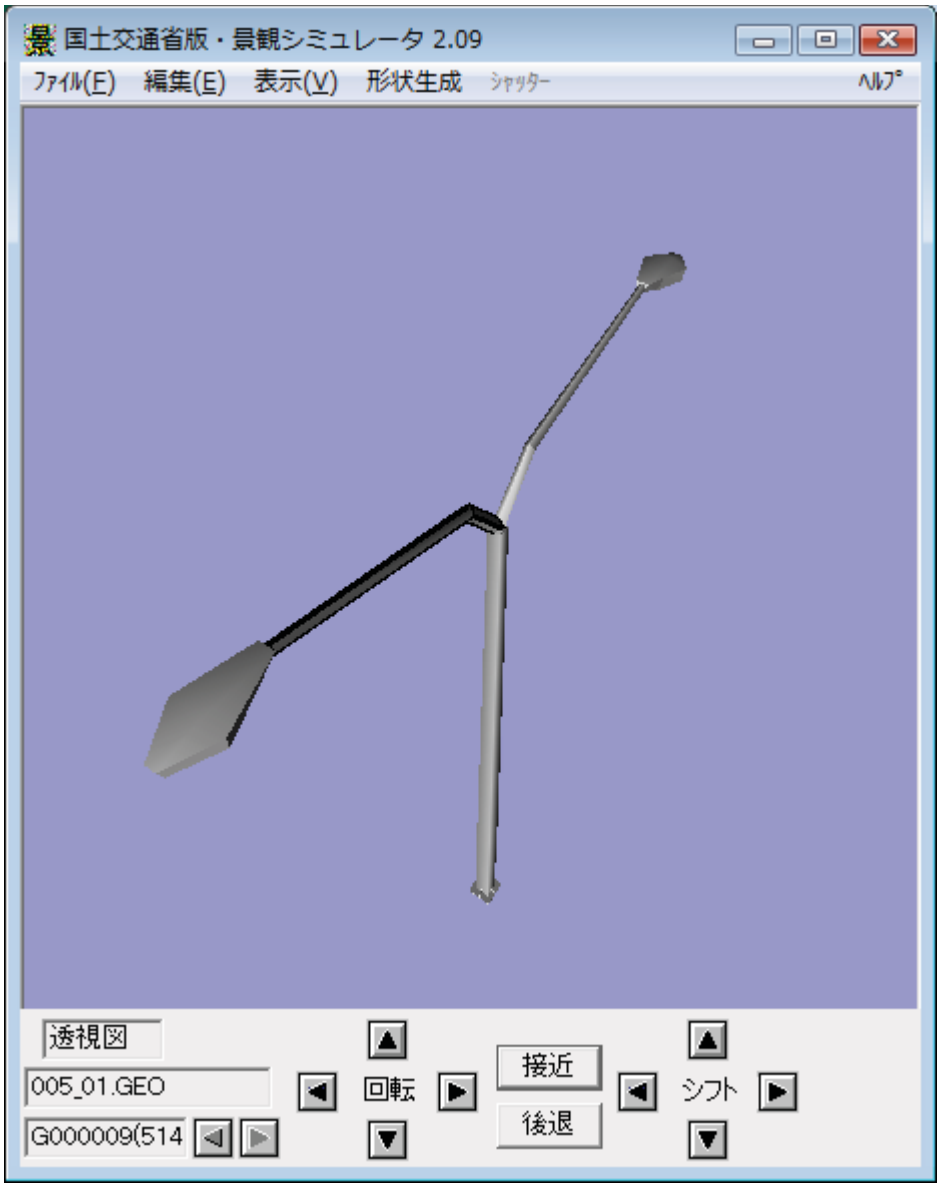


図３－１３：サンプル・データ１

（２）サンプルデータ２：街灯のランプ部分

・　　005_2. geo

表示例）

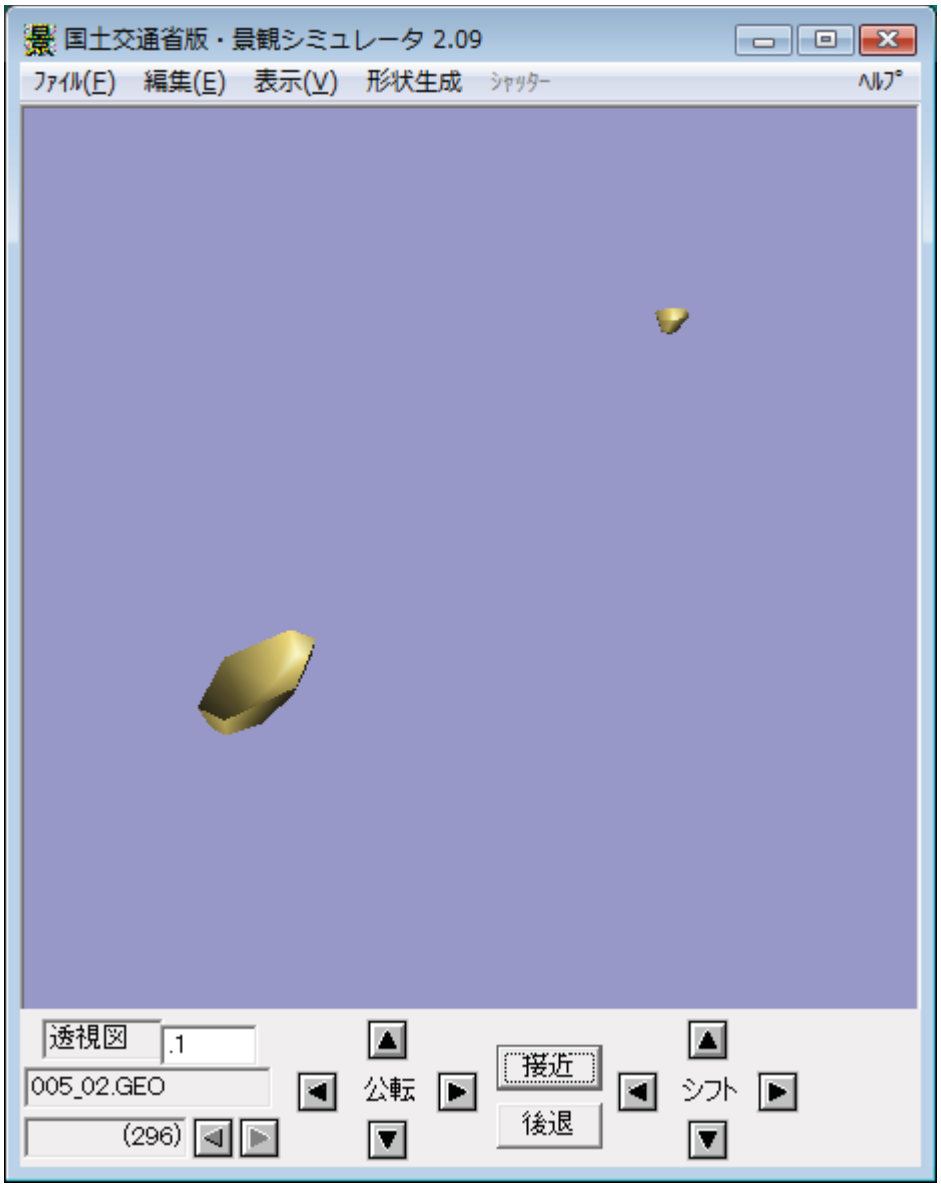


図３－１４：サンプル・データ２

（３）サンプルデータ３：街灯全体

リスト３－３５：total005.geo

```
#
    ROOT = GROUP();
#
#    Child model link
#
    G000000 = FILE(005_01.geo);
    L000000 = LINK(ROOT, G000000);
```

```
#
    G000001 = FILE(005_02.geo);
    L000001 = LINK(ROOT, G000001);
```

表示例)

図３－１５：ファイル参照とリンクによる合成

この、FILE コマンドと LINK コマンドで構築することにより、複雑な景観構成をコンパクトなファイルで記述することができる。データベースとして予め用意してある構成要素データを配置しながら構築したデータは、一般にこのような構成を有しており、景観検討のためのデータ構築作業としては極めてよく用いられる方法である。

３－７．マテリアル・ファイル

（1）　マテリアル・ファイル

マテリアル・ファイルは、景観材料データベース内のマテリアル情報を格納したデータとする構想が開発初期には存在し、一部の変数名などに名残を留めているが、実際に開発されたシステムにおいては環境設定ファイルの「FILE_PATH_MATERIAL」エントリーで定義されたディレクトリ（標準： kdb/material）にテキスト・ファイルとして並列的に置くこととなり、Ver.2.09まで継承している。

その役割は、鉄鋼、コンクリート、舗装、芝生、壁材料など、固有的形状を持たず使用される場所・状況により形状が変化する材料の固有の光学特性を記述するためのファイルである。

ひとつのマテリアル・ファイルには、一群のマテリアルが定義されている。例えば、日塗工の色見本帳は、色コードをマテリアル名とするマテリアル定義の集合体として、一つのファイル「nittoko.mtl」にまとめられている。

マテリアル・ファイルは、テキスト・ファイルであり、マテリアル名称をキーとし、材料の表面特性に関するパラメータが記述される。

マテリアル・ファイルにおいては、LSS-G ファイルの中で直接記述することができるカラー、テクスチャの他に、鏡面反射率と輝度（発光）も定義することができる。LGG-G ファイルにおいては、カラーは４つの値（RGB と透明度を表すα）で指定するが、マテリアルファイルにおいては、色彩を表す RGB または Lab 値と、透明度を表す TRANSPARENCY を分離している。

全てのパラメータに、期間を定義することができる。色彩、鏡面反射率およびテクスチャを期間毎に定義することにより、経年変化のシミュレーションを簡単に実行することができる。

マテリアル・ファイルの仕様を以下に示す。

表３－４：マテリアル・ファイルの仕様

･１カラム目が’#’の行はコメント行である。 ･マテリアル名称の宣言は１カラム目から始まる。 ･マテリアルの定義内容を関するキーワード行は１カラム目が空白またはタブであり、これにキーワード、期間の定義、パラメータが続く。 ・キーワードには次の種類がある。 ①RGB 拡散反射係数（通常のカラー）を直接ＲＧＢの３値で指定する（各 0.0≦≦1.0)。 ②Lab 拡散反射係数をＬａｂの３値で記述し、内部で変換して適用する。 ②SPECULAR 鏡面反射係数は、正反射により、表面が平滑な材料のツヤを表現する（各 0.0≦≦1.0)。

指定されなかった場合のデフォルトは0.0である。

④TRANSPARENCY

透明度は0.0≦≦1.0の範囲で指定する。0.0が透明で、指定されなかった場合のデフォルトは1.0（不透明）である。

⑤TEXTURE

テクスチャを記述する画像ファイル名称を指定する。

⑥EMISSION

放射光係数は０≦≦１の範囲で指定する（EMISSION)。

・期間の定義は、適用される時刻の範囲を指定できる（省略可)。
・期間指定には0.0以上の浮動小数点値を用い、[開始時点-終了時点]の形式で記述する。
・期間範囲がオーバーラップしないキーワードを任意個数記述できる。

リスト３－３６：マテリアルの記述例

```
ORENGE
        RGB [0-365]      1.0  0.5  0.0
        RGB [366-1000]   0.8  0.4  0.0
        SPECULAR [0-1000]        0.0
        TRANSPARENCY [0-1000]    1.0
SPECULAR_96
        RGB[0-500000]            1.0      1.0      1.0
        SPECULAR[0-500000]       0.96
        TRANSPARENCY[0-500000]   1.0
NORI
        RGB[0-99]                1.0      1.0      1.0
        RGB[100-10000]           1.0      1.0      1.0
        TEXTURE[0-99]            slop_1.sgi
        TEXTURE[100-10000]       slop_2.sgi
WHITE3
        RGB                      1.0      1.0      1.0
        SPECULAR                 0.0
        TRANSPARENCY             1.0
        EMISSION                 1.0
```

マテリアル・ファイルのロードは、ＤＢＩＬライブラリのdbLoadMaterial関数(dbms.c)により行われる。マテリアルのロードは、期間の定義が、現在のシステムの時刻に合致するキーワードに関してのみ実行される。

LSS-G ファイルのロードに際しては、インタープリタにより、LSS-G ファイルから参照されたマテリアルのリストが作成され、ロードの最終段階にリストされた全てのマテリアルが、dbLoadMaterial 関数によりロードされる。

マテリアル編集ダイアログ(４－４(21),(24))で、あるマテリアルが選択された場合には、そのマテリアルがロードされる。また、グラフィックなマテリアル編集ダイアログ(４－４(25))では、選択したマテリアルファイルで定義される全てのマテリアルをロードし、一覧表示する。

経年変化ダイアログ（４－４(32)）の操作や、シーンの切換に伴い、システムの時間が変更された場合にも、新たに設定されたシステム時間を用いてマテリアルの再ロードを行う。

Ver.2.09 のマテリアル・ファイルのロードに際して、以下のようにエラー処理している。
・名称の中にキーワードが一つもない場合：警告(3345)を発し、マテリアルは適用しない。
・無効なキーワードが宣言された場合：警告(3343)を発し、有効キーワードのみを適用。
・期間の定義がない場合：システム時間にかかわりなく常にロードする。
・同じマテリアル名称が重複して登録されていた場合：最初に検出したものを採用する。
・有効なキーワードが複数存在する場合：最後の有効なキーワードが適用される。
・拡散反射係数に係のキーワードが存在しない場合：デフォルト値 0.8 0.8 0.8 を適用。
・パラメータが必要数に不足する場合：存在する分のみ適用し、残りはデフォルト値。
・パラメータが必要数を超える場合：必要な数のみ適用し、残りは無視する。
・期間の開始時点が負数であった場合には、警告(3338)を発し、そのマテリアルを適用しない。

LSS-G 形式のファイルにおいて、木・石・鉄・アスファルト・コンクリート・ガラスなどの基本的な素材の光学特性を直接数値定義するのではなく、マテリアルとして間接定義しておくことにより、将来の表示処理技術の高度化に際して、LSS-G 形式のファイルを変更することなく対応が可能となる。また、表示に直接関係しない、重量や物質含量や価格等の集計分析も可能なデータとなる。

３－８．画像ファイル

景観シミュレーション・システムの開発に着手した 1993 年当時は、TIFF、BMP 形式などもされていたが、基本的な画像保存形式として、SGI 形式（グラフィック・ワークステーションと OpenGL を提供していた Silicon Graphics 社による公開性の高い単純な形式）を採用した。RGB という拡張子も用いられる場合がある。IMGSGI.lib ライブラリで入出力処理を行っている。

Ver.2.03 以降、Windows 上で背景画像等を利用するために、BMP 形式のファイルを読み込めるように拡充した。Windows 系の開発環境で入出力関数が用意されている。

TIFF 形式は、Aldus 社から仕様書を入手することができる公開性の高い形式であったが、

圧縮方法を含め様々なバリエーションを含む形式であるため、フルスペックでは対応していない。しかし、衛星画像を配布する際に用いられている GeoTiff 形式は、この内の単純な部分仕様に従っているため、支障なく使用することができる。

その後、圧縮効果の高い JPEG 形式が、デジタルカメラ等の普及に伴い広く利用されるようになったため、背景画像やテクスチャとして取り込めるようにした。また、表示画面をファイル保存し、報告書等に使用する際にも便利であるため、出力ファイル形式としても選択可能としている。しかし、圧縮に伴い、ファイルは小さくなるものの、画像データとしての同一性は失われるため、画像解析的な目的には適さない。JPEG.lib で入出力を行っている。

GIF 形式は、入出力プログラムにライセンスの制約があるため採用していない。景観シミュレーションに使用するためには、画像編集ソフトなどを用いて別途変換する必要がある。PNG 形式は国土地理院による電子国土等に用いられている。サーバー側の機能開発では、画像の WEB 配信に使用したが、クライアント側で動作する本システムでは、まだ対応していない。これも別途変換する必要がある。

SGI 形式は、現在では殆ど流通に用いられておらず、ファイル形式は単純で固定されているため、将来的に仕様変更される危険性が小さく、データベースに登録した基本的構成要素のためのテクスチャなどを変更することなく長期に保存するためには最も適していると言える。最大の特徴は、アルファ値を有する画像を記述できる点であり、樹木等を長方形にテクスチャを貼ったオブジェクトとして表現することは、JPEG などの形式では不可能である。SGI 形式における圧縮は、同じビットパターンが連続する場合に、そのパターンと繰り返し数に置き換えるという単純な方法であるが、面や背景が、同じカラーまたは透明部分を有するような CG 画像をコンパクトに保存するためには効果的である。

３－９．エラーメッセージ定義ファイル

テキスト・ファイルであり、ASCII コードまたはこれを拡張したシフト JIS コード等により記述する。システム起動時、または言語切換時に z3LoadMessage()関数によりロードされ、以後メモリ上に置かれ、各種メッセージ表示が必要となる場合に利用される。メッセージは、種別と番号によりアクセスされ、z3Message(メッセージ番号,引数・・・)等の関数により表示に使用される。ファイルのフォーマットは以下の通りである。

① 行単位でエラーメッセージを定義する。

② 行の最初の１文字が、アルファベット大文字の E、W、I、C で始まる行のみをデータとして解釈し、それ以外の文字で始まる行や、単に改行のみの行はコメント解説等として読み飛ばす。

③ 上記の１文字コードは、メッセージの種類を区別する。

E:エラー(Error)

W:警告(Warning)

I:情報(Information)

C:確認(Confirmation)

原則として、システムの側に責任が帰する障害の場合に E を、またユーザーの誤操作等に帰する障害の場合に W を用いる。C(確認)は、通常は z3Confirmation 関数で用い、「はい」または「いいえ」の選択結果(戻り値)を続く処理に反映させる。

④　1 文字コードの後にスペース区切りで、メッセージ番号を記述する。

メッセージ番号は、メッセージの種類に関わらずユニークでなければならない。整理のために、エラーは 1000 番台、警告は 3000 番台、情報は 5000 番台、確認は 7000 番台としているが、必ずしもこの慣習に従う必要はない。また番号が昇順に並んでいる必要もない。

⑤　メッセージ番号の後にスペース区切りで改行まで続く文字列をエラーとして表示する。

この文字列は、printf 文のフォーマット文字列と同様に、引数で指定する整数値(%d)、浮動小数点値(%f)、文字列(%s)を含めることができる。④　W　の場合、警告として表示する。ERR_MSG.TXT ファイル中におけるメッセージの定義例をリストに、またこれを用いたエラーメッセージの表示のプログラム例をリストに示す。

リスト３－３７：ERR_MSG.txt による定義

```
W 3006 %s のセットに失敗しました
```

リスト３－３８：エラーメッセージのプログラム例

```
CString s;
int i;
s.Format("EFFECT(%s",e->params[0]);
for(i=1;i<e->count;i++){
        s += _T(",");
        s += _T(e->params[i]);
}
s += _T(")");
z3Message(3006,s);
```

(3)実際の表示

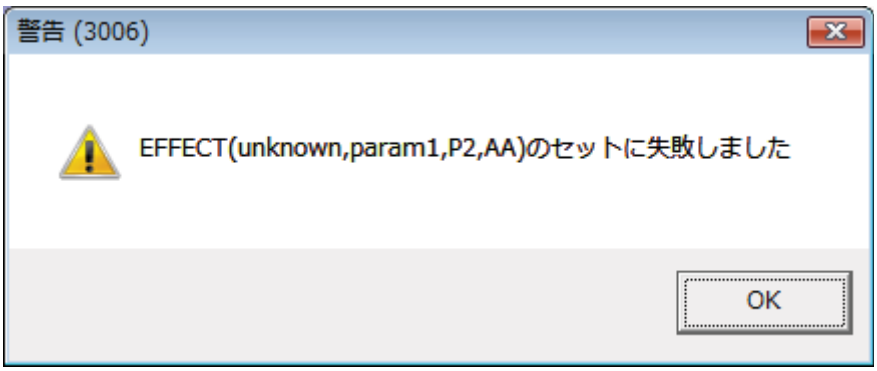


図３－１６：警告の表示例

３－１０．選択項目定義ファイル

ユーザーが選択するアイテムを、プログラム中に固定的に記述するのではなく、外部テキスト・ファイルにおいて定義しておき、これを用いて選択ダイアログやメニューの表示を行うものである。これには以下のものがある。

（1）ＥＸＴ．ＴＡＢ

外部関数の一覧を格納する。ksim/bin に置かれる。選択可能な外部関数の範囲を定めると共に、引数の型を指定する。新たな外部関数を作成し追加する場合には、このテーブルに登録する。

改行を単位とするレコードである。#から始まる行はコメントとして無視される。

行のはじめはFILE から始め、()の中に、外部関数名とこれに続く引数をデータ型で記述し、「;」（セミコロン）で終了する。

データ型は、表３－４に示す通りである。

表３－５：EXT.TAB の引数のデータ型

INT	整数
FLOAT	浮動小数点
DOUBLE	倍精度実数
STRING	文字列
FILE	ファイル名称
FACE	指定されたＬＳＳ－Ｇファイルを開き、最初のグループの最初の面を取得する。
LINE	指定されたＬＳＳ－Ｇファイルを開き、最初のグループの最初の線を取得する。
TIME	現在の経年（日数）を倍精度実数の形式でシステムから受け取る。

Ext.tab ファイルは、システム起動時に読み込まれ、メニューの[形状生成][オプション]が選択された時に、選択ダイアログを開き、各関数の名称の一覧を表示する。

リスト３－３９：EXT.TAB

```
#外部関数一覧
FILE(CUBE,DOUBLE,DOUBLE,DOUBLE,DOUBLE,DOUBLE,DOUBLE);
FILE(SPHERE,DOUBLE,DOUBLE,DOUBLE,DOUBLE);
FILE(CYLINDER,DOUBLE,DOUBLE,DOUBLE,DOUBLE,DOUBLE,DOUBLE,DOUBLE);
FILE(CONE,DOUBLE,DOUBLE,DOUBLE,DOUBLE,DOUBLE,DOUBLE,DOUBLE,DOUBLE);
FILE(FLATCYLI,DOUBLE,DOUBLE,DOUBLE,DOUBLE,DOUBLE,DOUBLE,DOUBLE,INT);
FILE(FLATCONE,DOUBLE,DOUBLE,DOUBLE,DOUBLE,DOUBLE,DOUBLE,DOUBLE,DOUBLE,INT);
#FILE(SWEEP,FACE,DOUBLE,DOUBLE,DOUBLE);
FILE(STEEL,DOUBLE,DOUBLE,DOUBLE,DOUBLE,DOUBLE);
FILE(SWEEP1,FACE,LINE);
FILE(SWEEP2,FACE,FACE);
#FILE(SWEEP1,FILE,FILE);
#FILE(SWEEP2,FILE,FILE);
FILE(HSTEEL,DOUBLE,DOUBLE,DOUBLE,DOUBLE,DOUBLE);
FILE(CSTEEL,DOUBLE,DOUBLE,DOUBLE,DOUBLE,DOUBLE);
FILE(TSTEEL,DOUBLE,DOUBLE,DOUBLE,DOUBLE,DOUBLE);
FILE(LSTEEL,DOUBLE,DOUBLE,DOUBLE,DOUBLE,DOUBLE,DOUBLE,DOUBLE);
FILE(SAMPLE,DOUBLE,DOUBLE,DOUBLE);
FILE(STRING,STRING,STRING,STRING);
```

```
FILE(STAIR,DOUBLE,DOUBLE,DOUBLE,DOUBLE);
FILE(TIMESPHERE,DOUBLE,DOUBLE,DOUBLE,TIME);
FILE(PERIOD,FILE,DOUBLE,DOUBLE,TIME);
FILE(VRML2LSS,FILE);
FILE(URL,STRING);
FILE(HAKOBUIL,DOUBLE,DOUBLE,DOUBLE,FILE);
FILE(BS2LSS,STRING);
FILE(TAMENTAI,INT,DOUBLE);
FILE(CONE,DOUBLE,DOUBLE,DOUBLE,DOUBLE,DOUBLE,DOUBLE,DOUBLE,DOUBLE);
FILE(STEEL);
FILE(SEGITIGA,DOUBLE,DOUBLE,DOUBLE);
FILE(FIRE2LSS,FILE);
FILE(SCADEC2,STRING,STRING);
```

Ver.2.09 においては、原始図形とユーザー定義の外部関数の内部処理を一元化したが、メニュー構成においては、従前の操作環境を維持している。このため、上記におけるメニューのオプションから起動するダイアログにおいては、ext-tab に登録されている関数の全てを表示するのではなく、この内、原始図形及び基本構成要素：型鋼以外のものについてのみ、表示している。

（2）ＰＬＵＧＩＮ．ＴＡＢ

プラグインＤＬＬの一覧を格納する。ksim/bin に置かれる。選択可能なプラグインＤＬＬの一覧を記述する。新たな外部関数を作成し追加する場合には、このテーブルに登録する。コンマ「,」区切りで、表示する名称と、DLL の名称を１行単位で記述する。

リスト３－４０：PLUGIN.TAB

```
地形編集,land.dll
トンネル,TUNNEL.dll
道路法面生成,nori.dll
園路生成,ParkRoad.dll
日本語版,test.dll
```

基幹部分のメニューから[形状生成][プラグイン]を選択すると、このファイルに登録されている見出し文字列が、サブメニューとして一覧表示される。

Ver.2.09 の多言語環境においては、メニュー項目として表示する文字列の部分のみ各言語に翻訳し、ファイル名称は、autotex.set のままで、各言語ディレクトリ下に置く。

（3）ＲＯＡＤ＿ＳＥＣ．ＳＥＴ

道路生成ダイアログ（[形状生成]［基本構成要素］[道路］で起動）で選択する道路断面ファイルの一覧を記述する。道路断面ファイルはＬＳＳ－Ｇ形式で、断面の構成要素をＬＩＮＥコマンドで定義している。ROAD_SEC.SET ファイルは、kdb/geometry ディレクトリに置かれる。平面生成ダイアログ（[形状生成]［原始図形］[平面]）で作成した図形を道路断面として保存した場合には、ファイル名が、ROAD_SEC.SET ファイルの末尾に自動的に追加される。ファイルの内部は、改行区切りで名称を列挙している。

リスト３－４１：ROAD_SEC.SET

```
roadsection1.geo
roadsection2.geo
roadsection3.geo
1117doro.geo
Sarijadi.geo
```

```
senro.geo
```

（4）ＲＩＶＥＲ＿ＳＥＣ．ＳＥＴ

河川生成ダイアログ（[形状生成[]基本構成要素]［河川]で起動）で選択する河川断面ファイルの一覧を記述する。河川断面ファイルはＬＳＳ－Ｇ形式で、断面の構成要素をＬＩＮＥコマンドで定義している。RIVER_SEC.SET ファイルは、kdb/geometry ディレクトリに置かれる。平面生成ダイアログ([形状生成][原始図形[]平面]で起動)で作成した図形を河川断面として保存した場合には、ファイル名が、RIVER_SEC.SET ファイルの末尾に自動的に追加される。ファイルの内部は、改行区切りで名称を列挙している。

リスト３－４２：RIVER_SEC.SET

```
riversection1.geo
riversection2.geo
riversection3.geo
riversection4.geo
1117kawa.geo
Sarijadi.geo
senro.geo
```

（5）ＴＵＮＮＥＬ＿ＳＥＣ．ＳＥＴ

トンネル生成ダイアログ（Ver.2.09 においてはプラグインＤＬＬとして分離）で選択するトンネル断面ファイルの一覧を記述する。トンネル断面ファイルはテキスト形式で、断面の断面形を独自形式で定義している。TUNNEL_SEC.SET ファイルは、kdb/geometry ディレクトリに置かれる。ファイルの内部は、改行区切りで名称を列挙している。

リスト３－４３：TUNNEL_SEC.SET

```
riversection1.geo
riversection2.geo
riversection3.geo
riversection4.geo
1117kawa.geo
Antapani.geo
Ankyo.geo
```

（6）ＥＮＲＯ＿ＳＥＣ．ＳＥＴ

園路生成ダイアログ（Ver.2.09 においてはプラグインＤＬＬとして分離）で選択する園路断面ファイルの一覧を記述する。園路断面ファイルはＬＳＳ－Ｇ形式で、断面の構成要素をＬＩＮＥコマンドで定義している。ENRO_SEC.SET ファイルは、kdb/geometry ディレクトリに置かれる。ファイルの内部は、改行区切りで名称を列挙している。

リスト３－４４：ENRO_SEC.SET

```
roadsection1.geo
roadsection2.geo
roadsection3.geo
roadsection4.geo
1117doro.geo
```

（7）ＡＵＴＯＴＥＸ．ＳＥＴ

テクスチャ編集画面のメニューにある［自動貼り付け］を選択した時に表示するメニューを階層的に定義する。

リスト３－４５：AUTOTEX.SET

```
1 水面
        2 速い流れmizu-D1.sgi 2 2
        2 やや速い流れmizu-E1.sgi 2 2
        2 普通の流れmizu-C1.sgi 2 2
        2 遅い流れmizu-B1.sgi 2 2
        2 静止水面mizu-A1.sgi 2 2
1 法面
        2 コンクリートslop_1.sgi 3 3
        2 芝１slop_2.sgi 3 3
        2 芝２slop-E1.sgi 1 1
1 ブロック
        2 石１rock_1.sgi 1 1
        2 石２slop-G1.sgi 1 1
        2 石３slop-G2.sgi 1 1
        2 煉瓦renga_1.sgi 1 1
```

ファイルの内部は、改行で区切られた行単位のデータで構成される。一つのデータは、スペース区切りで、メニューの階層を示す数値、メニュー表示に用いる各言語の文字列、テクスチャファイル名、横倍率、縦倍率を記述する。Ver.2.09 においては、メニュー項目として表示する文字列の部分のみ各言語に翻訳し、ファイル名称は、autotex.set のままで、各言語ディレクトリ下に置く。

３－１１．その他の一時的ファイル

景観シミュレーションの操作を行う際に、一時的ファイルが動的に生成される。

（1）景観データベースとのインターフェース

基幹部分における配置ダイアログにおいて、配置するオブジェクトを選択する際に、検索のために景観データベース検索画面(別実行形式)を起動することができる。検索された結果は、一時的ファイル tmp001.txt により基幹部分に返される。このファイルは、スペース区切りで、データベース種類、ファイル種類(画像、モデル等)、およびファイル名を示す１行のデータから成るテキスト・ファイルであり、検索画面の起動時には、

０ ０ NULL

に初期化される。検索が行われると、

１ ８ M_3a.geo （景観構成要素、LSS-G 形式、ファイル名「M_3a.geo」の意味）

のように、具体的なファイルを示す内容が記入されて基幹部分に返される。

検索画面において、検索された物件の３次元データを確認表示する際に、ファイルを指定して、別プロセスとして基幹部分が起動される。この時、基幹部分が初期表示するファイルを指定するために、tmp002.txt が使用される。このファイルの内容は、tmp001.txt と同様である。

この他に同様の形式の tmp003.txt が、ファイル保存操作の記録に用いられている。

これらの一時的ファイルは、ksim/temp ディレクトリに作成される。入出力は、アプリケーション・ライブラリ関数（４－３）の WriteToFile 関数、及び ReadFromFile 関数により処理している。

（２）成熟都市シミュレータ等とのインターフェース

市街地自動生成を行う成熟都市シミュレータ（文献 16）は、景観シミュレータを三次元表示のための出力装置として使用する。データ送出は、設定ファイル timerpath.txt の１行目のフラグファイルと２行目のデータファイルを用いて行う。市街地生成の側では、フラグファイルの内容がゼロであることを確認した上で、データファイルに新しいデータを記録し、フラグファイルを１にセットする。景観シミュレータの側では、フラグが１であることを確認して、データファイルを読み込み、フラグをゼロにセットする。データファイルの内容は、ＬＳＳ－Ｇ形式である。

（３）外部関数ダイアログとのインターフェース

ユーザーが外部関数（原始図形およびユーザー定義）のパラメータ設定ダイアログを操作し、OK で終了した場合には、「関数名.geo」という名称の１行だけから成る、パラメータを記述したファイルが ksim/temp ディレクトリに作成され、次の形状生成過程で使用される。また、過去に外部関数を用いて生成したオブジェクトが選択され、パラメータが再設定される場合にも、同様のファイルが作成され、ダイアログ部の初期表示のパラメータの受け渡しに使用される（５－２参照)。

（４）外部関数による形状生成時

パラメータの設定が終了し、外部関数が実際に形状を生成する際には、外部関数により「外部関数名.g」というファイルが、ksim/temp ディレクトリに作成される。制御が戻ってきた段階で、基幹部分でこのファイルを開き、内容を解析して、地物の中に追加する（５－３参照)。

４．プログラム構成

本章では、景観シミュレーション・システム全体のプログラムの基本的構造を解説する。これは、システム全体管理、内部のデータ構造とデータ管理、ユーザー・インターフェース、ファイル入出力、ＯＳとのインターフェース等の要素を含む、一般的な三次元データを扱うシステムに必要な基本的構成を有している。この上に様々な景観検討のための機能を実現した。基本的な部分の理解は、特定の目的のための機能を今後開発していくために欠くことができない基礎となっている。

例えば、シーンを平面図として表示し、そこにデータベースからオブジェクトを選んで、配置操作を行う画面を取り上げてみよう。この画面には、以下のような機能が関係している：

①この編集画面構成のデザインに従って、ウィンドウを初期化し表示する機能
②シーンを構成する地物全体のデータを取得し、その内の指定された表示範囲を、指定された寸法の画面に配置図としてグラフィックに表示する機能
③配置するオブジェクトをデータベースから検索・選択する機能
④配置する場所、路線区間、エリアなどを画面で設定する機能
⑤配置する密度や間隔、向きや位置やサイズの揺らぎ等を設定する機能
⑥各配置地点の地面の標高を計算する機能
⑦配置の処理を実行し、その結果に基づいて地物構成を変更する機能
⑧結果が期待通りでなかった場合に、取り消して元の状態に戻す機能
⑨処理の途中でエラーが発生した場合に、メッセージを表示する機能
⑩ユーザーの理解を助けるためのヘルプの表示

これらを根底で支えているのが、ライブラリ関数群であり、システムを運用する土台となる基本的なデータ構造を管理している。

一方、ユーザーから最も近いところで、画面に表示されたボタン等のコントロールに対する操作や、画面上でのマウス操作を受け付け、対応する機能をライブラリから起動し、処理結果を画面表示に反映させるハンドラ・ルーチンがある。

更に、これらの間をつなぐ、中間的なアプリケーション・ライブラリ関数がいくつか介在している。

４－１．概要

（1）景観シミュレータと景観データベース

景観シミュレータは、単独で起動し動作するアプリケーションとして使用することもできるが、データベースと組み合わせて、データベースから起動し三次元モデルを表示したり、ウェブ・ブラウザと組み合わせて、インターネット上に公開されている三次元データを表示したりするような使い方にも対応している。

景観データベースには、国土交通省がデータを蓄積する事例データベース、景観検討データを構築する上で頻繁に利用される一般的な地物の三次元モデルを集めた、構成要素データベース、及びマテリアルを集めた景観材料データベースから成っている。

90 年代の初期のバージョンにおいては、これら全てをユーザーが操作する PC の中に置き、連携しながら作業を進める環境を前提としていた。2001 年に開発したまちづくり・コミュニケーション・システムにおいては、データを WEB 上に置き、必要に応じてダウンロードし、作業を行うような環境も実現した。

これら全体構成の内、サーバー側で動作するデータ配信機能や GIS 機能のソフトウェアは、別稿に譲ることとし、本書では、クライアント側（PC）で動作する景観シミュレータを中心とするシステムの構成について解説している。

アプリケーションとしては、景観シミュレータ（sim.exe）、３種類の景観データベースブラウザ（yuu.exe, kou.exe, zai.exe）が主なものである。それ以外に、ファイル・コンバータや、データベース入力機能などの支援プログラムが存在している。

更に、景観シミュレータと一体で動作する多くの外部関数(第 5 章)、及びプラグインＤＬＬ(第 6 章)が、それぞれ独立したビルドとなっている。

表４－１：景観シミュレーション・システムを構成する実行形式

Sim.exe　景観シミュレータ基幹部分
Yuu.exe　景観事例データベース検索
Kou.exe　景観構成要素データベース検索
Zai.exe　景観材料データベース検索
Editor.exe データベース入力編集機能
貿易.exe および関連するコンバータ別実行形式
外部関数
プラグイン DLL

（２）ソースコードの全体構成

景観シミュレーションのシステムは、大きく見て、８のライブラリと、ダイアログ毎の８０程度のハンドラ・ルーチンがその大半を占めている。前者は、開発初期の頃から一貫したデータ構造を処理するために維持・改良されてきたものである。可搬性が高いＡＮＳＩ－Ｃ言語をベースとして記述されている。初期の頃と比較すると、データ規模も大きくなってきており、またマシンのメモリ容量、ハードディスク容量、処理速度などは比較にならない程進歩したが、基本的な構造は何も変わっていない。

一方、後者は、現場のニーズや、新たなＯＳや開発環境に適応しつつ、柔軟に改良・追加を加えた結果、現状に至っている。全体の構成ダイアグラムは、図４－１に示したような構成となっている。

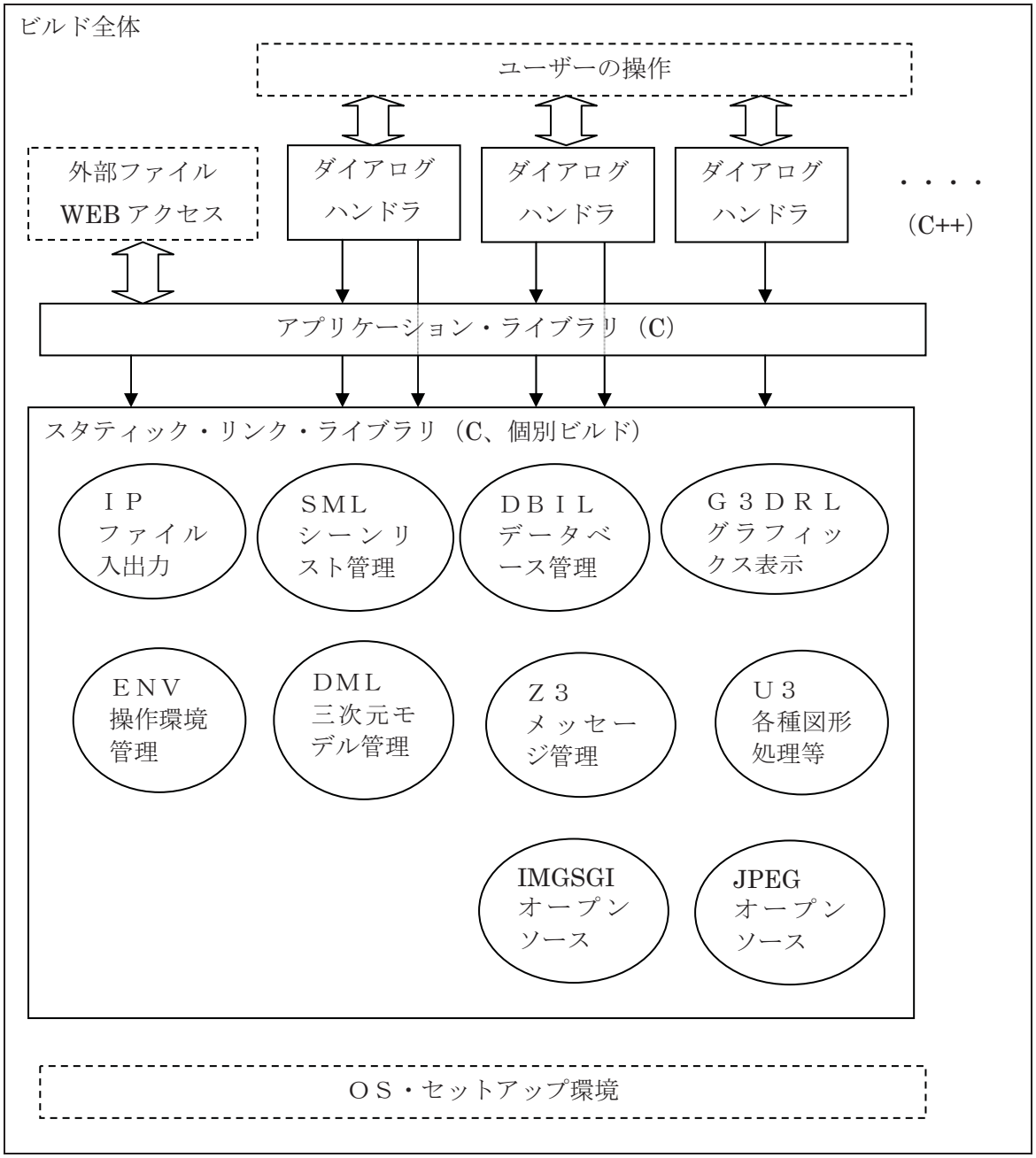


図４－１：景観シミュレータのビルド構成（参照元→参照先）

開発環境のビルドの編成は、まず C 言語による共通のライブラリが、基本的なデータ構造の管理機能を提供しており、スタティック・リンク・ライブラリ(.lib)としてライブラリ単位のビルドとなっている。景観シミュレータ、３種類の景観データベース検索機能などの実行形式(.exe)のビルドは、ユーザー・インターフェースを構成する各ダイアログのハンドラ（C++のクラス）のソースコード（Ｃ＋＋）と、アプリケーション・ライブラリのソースコード（Ｃ）から構成されているビルドに、予めビルドされた上記のライブラリ関数群をスタティック・リンク・ライブラリ(.lib)として組み込む流れとなっている。

①ライブラリ関数（C)

１章で示した開発当初のマルチ・プラットフォームの環境から、ライブラリはOSに依存しない機能としてC言語で記述されている（本章第2節)。共通ライブラリとしては、

Sml.lib：シーンリストを管理する（付録B-1　シーン管理ライブラリ）

Dml.lib：三次元モデルを管理する（付録B-2　データ管理ライブラリ）

Dbil.lib：景観データベースを管理する（付録B-3　景観データベースライブラリ）

G3drl.lib：OpenGL画面の描画処理を管理する（付録B-4　レンダリングライブラリ）

Ip.lib：外部ファイル入出力を管理する（付録B-5　インタプリタライブラリ）

Env.lib：動作環境を管理する（付録B-6　環境設定ライブラリ）

U3.lib：各種図形演算処理などのパッケージ（付録B-7　ユーティリティライブラリ）

Z3err.lib：エラーメッセージを処理する（付録B-8　メッセージライブラリ）

がある。この他に、公開オープンソースを利用しているものとして、

JPEG.lib：JPEG形式の画像ファイルの入出力

IMGSGI.lib：SGI形式の画像ファイルの入出力

をリンクしている。

ライブラリ関数からアプリケーション・ライブラリ関数やダイアログのハンドラを起動することはない。しかし、ライブラリのうちのいくつかは依存関係にある。これについては、（６）で解説する。

②アプリケーション・ライブラリ関数（C）

この他、C言語のライブラリと、C++言語のダイアログをつなぐ役割を果たす、C言語の共通関数群を用意している。主要なものは、主としてシステム状態を記憶するスタティック変数を管理するcommon.cと、データ変換を行っているdataope.cに記述された関数群があり、これらから起動する下請け的な関数を提供するいくつかのC言語のソースコードがある（本章第３節)。

この内、言語別のバージョン枝分かれの問題を解決するためにVer.2.09で新たに追加した、多言語対応のMULTILANG機能については、第１１章で解説している。

アプリケーション・ライブラリ関数からダイアログ・ハンドラを呼び出すことはない。

③ダイアログ・ハンドラ（C++)

景観シミュレータの外見を構成する大小８０程度のダイアログに関しては、C++のクラスとして記述された、それぞれを処理するハンドラが存在している（本章第4節)。これらからは、ライブラリ関数を直接起動するか、アプリケーション・ライブラリ関数を起動し、そこから間接的にライブラリ関数を起動する。

更に、景観シミュレータには、基幹部分(sim.exe)から派生的に起動する外部関数、及び連携動作するプラグインDLLを用いることができ、すでに多くの実装がある。このような外部関数やプラグインDLLを今後追加的に開発し連携動作するためのひな形やライブラリを用意している。これらの詳細については、第５章、第６章で解説している。上記のスタティック・リンク・ライブラリは、プラグインDLLのビルドにスタティックにリンクした

場合、グローバル変数等が基幹部分とは別のものとなるため、動作の一体性が失われる。そこで、基幹部分にリンクしたライブラリを直接利用するために、景観シミュレータで使用しているライブラリ関数は、DLL エクスポートするように設定してある。このため、各プラグイン DLL の開発においては、SIM.LIB だけをリンクすることで、基幹部分の実行形式に組み込まれたライブラリ関数を DLL インポートして利用することができる。

（3）動作環境

景観シミュレータは、単独で三次元モデル（LSS-G 形式のファイル）を見るだけのブラウザとして使用することもできる一方、大きなデータベースと共に本格的な三次元データ制作環境としてセットアップすることもできる。このため、景観シミュレータで操作する各種のファイル（三次元モデル、画像、テキストファイル等）は、目的に応じて特定のディレクトリに仕分けて置くことができる。とりわけ、景観データベースを構成する多くの共用ファイルと、現在設計検討中の建設プロジェクトに関係した固有のデータを区別することは整理のうえで有効である。更に、使用言語に依存したテキストファイルも、整然と整理する必要がある。

このディレクトリ編成は、環境設定ファイル kdbms.set の設定により行う。様々の目的のためのディレクトリを定義している。簡単な用途のためには、多くのデータを一つのディレクトリに集約するのが便利であろう。一方、大量のデータを扱う場合には、モデルやイメージなど、データの種類による仕分けや、プロジェクト単位の仕分けを行う方が便利である。このような必要とする操作環境に応じて柔軟に設定することができる。

システム起動直後にまず、env ライブラリがこの環境設定ファイルを最初に読み込み、様々なデータにアクセスできるように準備する。

（4）景観シミュレータの起動条件

単独のアプリケーションとしてユーザーが起動する場合には、OS から渡される引数はない。この場合には独立した操作環境として起動する。一方、景観データベースやウェブ・ブラウザから起動する場合には、引数として、表示すべきファイルや、データベースの種類に関する情報が渡される。

起動条件に応じた動作の場合分けは、CSimApp::InitInstance()関数の中で行っている。

単独で起動した場合には、CsplashWnd で初期画像を一定時間表示し、その間に初期化処理を行う。

データベースからの CreateProcess 関数による起動の場合には、「-f DB」という引数が渡される。この場合、初期化において CalledDB フラグ 1 を立て、以後の処理を行う。この場合、データベースによりテンポラリ・ディレクトリにある一時的ファイル「tmp000.txt」にデータベースの種類、表示すべきファイル等が格納されているので、そこから情報を取得し、ファイルの表示を行う。その際、部品として引用される三次元オブジェクト、画像

などは、データベースの種類に応じたディレクトリから取得する。

上記の一時的ファイルで指定された起動元のデータベースの種類は、dbms ライブラリのグローバル変数

```
int     DBno;
```

に登録される（０：事例、１：構成要素、２：材料）。この設定により、フルパスで指定されていない各種の参照ファイルの取得先ディレクトリを環境設定情報に基づいて定める。

Web ブラウザからの起動の場合には、フラグ２を立てる。初期化が終了すると、渡されたファイルをロードし、表示を行う。WEB から取得したシーン・ファイルの中の三次元モデルのファイルの所在は、http://から始まるアドレスで記述されているため、このサイトにアクセスして、関連するデータをダウンロードし表示する。ダウンロード先は、規定の作業用ディレクトリとするため、二回目からの表示にはダウンロードの必要がない。

（５）景観シミュレータが操作するデータのタイプ

景観シミュレータで編集し、表示するファイルには、三次元データをモデリングする LSS-G データ（ファイル拡張子.geo）と、モデリング結果に基づいて、光源や時間や視点設定を組み合わせたシーン群を構築する LSS-S データ（ファイル拡張子.scn）がある。扱うデータのタイプ区別は、［ファイル］以下のドロップダウンメニューから、新規作成またはファイルを開く選択を行う際に決定する。

現在編集しているデータのタイプは、common.c が記憶している。common.c は各種条件設定をスタティック変数に格納しており、各種ソースコードから共通に参照される。現在設定されているモードは、GetLssFileType()　関数で調べることができ、戻り値が K_LSS_S であればシーン、K_LSS_G であれば、ジオメトリと識別される。

LSS-S（シーン）タイプのデータは、SML ライブラリによって管理される。一つのシーンの構成は、s3Scene 構造体によって定義され、編集の結果は、この構造体の配列として、SML ライブラリの中で定義されたスタティック変数 scntab[]により管理されている。現在ロードされているシーンは、メイン画面の左下の操作ボタンにより逐次的に表示することができる。

LSS-G（ジオメトリ）タイプのデータは、

グループの一覧表 d3db を DML ライブラリの中で管理している。グループの作成(d3CreateGroup 関数による)や削除(d3DeleteGroup 関数による)の処理の中では、メモリブロックの取得や解放と同時に、この台帳への登録・削除を行っている。全てのグループからの検索は、このテーブルを利用して行っている。

一つのシーンは、表示する三次元モデルを持つことができる。更に、異なるシーンは異なるモデルを持つことができる。一つのシーンに定義されるモデルは、そのモデルを描画する起点となる最上位のグループへのポインタとして、シーン構造体に登録する。表示するシーンの切換に際して、表示すべきモデルが変わる場合は、モデルがない場合も含め、

表示しようとするシーンに登録されているルート・グループのアドレスを、da[ ]構造体にコピーすることにより実行される。

一方、DML ライブラリにおいては、このような複数の互いに独立したグループの集合体も、一つの共通テーブルで管理している。従って、表示において互いに無関係の、別シーンに属するモデル（それ以下の全てのグループ）も共通のテーブル上に登録される。登録に際して、FILE コマンドにより定義されるグループの内、外部関数ではなく LSS-G ファイルを直接参照している場合には、ロードの過程の中で起動される I3 ライブラリの i3LoadLssg 関数の中でこのテーブルの検索を行っている。もし、同一のファイルであれば、内容は固定的であり、再び同じ内容をロードする意味はないので、新たなグループは生成せずに、このグループを取り回して使用する。

リスト４－１：LSS-S 編集時におけるモデルのロード過程

LSS-S タイプの編集中で、シーンで定義するモデルのロードにおける呼び出し順序 s3CreateModel 関数→dbLoadGeometry2 関数→i3LoadLssg 関数

リスト４－２：LSS-G 編集時におけるモデルのロード過程

LSS-G タイプの編集中、LSS-G ファイルをロードする場合の呼び出し順序 common.c の LoadGeometry 関数→dbLoadGeometry2 関数→i3LoadLssg 関数

（6）リンクにおけるライブラリの依存関係

上記のように、景観シミュレータのビルドにおいては、８のライブラリをリンクしている。これらは互いに独立しているのではなく、ライブラリ間での相互依存関係が存在する。このため、例えば外部関数を作成するためにあるライブラリを利用すると、その前提条件として別のライブラリもリンクする必要が生じる場合がある。ライブラリの依存関係を図４－２に示した。

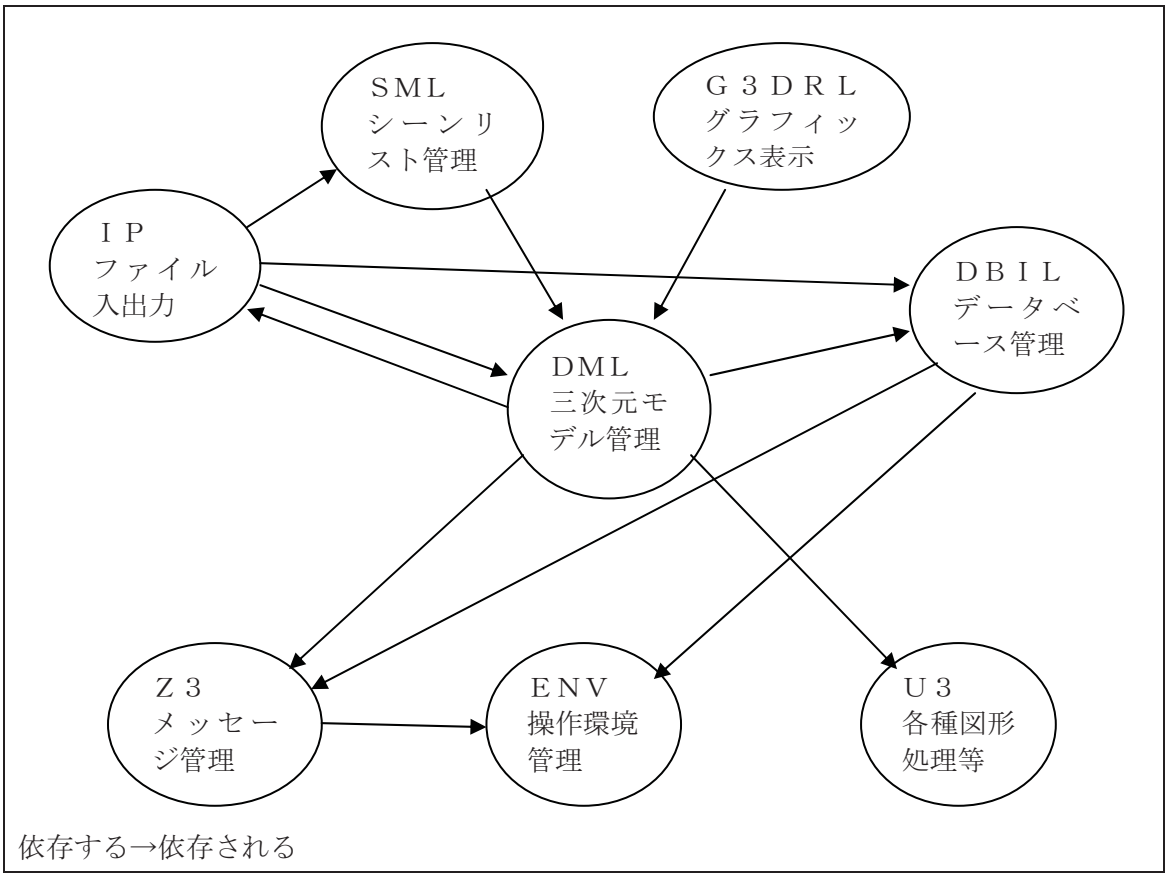


図４－２：ライブラリの依存関係

①┌　シーン管理ライブラリ（ＳＭＬ）

画像やモデルのロードに、ＤＭＬライブラリの関数を使用している。

②　データ管理ライブラリ（ＤＭＬ）

エラーメッセージにＺ３ライブラリの関数を使用している。

マテリアル、テクスチャのロードにＤＢＩＬライブラリの関数を使用している。

d3Gwaktu 関数,d3FilingFiles 関数が、ＩＰライブラリを使用している。

③景観データベースライブラリ（ＤＢＩＬ）

エラーメッセージにＺ３ライブラリの関数を使用している。

データの格納先ディレクトリを知るためにＥＮＶライブラリを使用している。

④グラフィックス表示ライブラリ（Ｇ３ＤＲＬ）

モデルのデータ構造を辿るために、ＤＭＬライブラリを使用している。

図形処理に、u3 ライブラリの関数を使用している。

エラーメッセージに z3err ライブラリの関数を使用している。

⑤インタプリタ（ＩＰ）

ファイル構築の途上で、SML、DML ライブラリの関数を使用している。

画像等の取得に際して、DBIL ライブラリの関数を使用している。

⑥環境設定ライブラリ（ＥＮＶ）

１箇所だけ z3err の関数を使用している

⑦ユーティリティライブラリ（Ｕ３）

u3grid.c が、DML、z3err の関数を使用している。

u3same~1.c１が、DML の関数を使用している。

u3slant.c が DML の関数を使用している。

u3trans.c が DML の関数を使用している。

⑧メッセージライブラリ（Ｚ３ＥＲＲ）

ENV でメッセージ・ファイルの所在ディレクトリを取得している。

（7）メモリ空間の相互関係

各ライブラリは、データ管理のためのスタティックなポインタまたは配列を有しており、これを用いて動的に取得・解放される各種メモリブロックを登録・管理するメモリ空間を有している。例えば、ENV ライブラリにおける環境変数テーブルや、z3err におけるエラーメッセージテーブルのように、独自に構築され、独自に開放される完結したデータは、ライブラリ間、あるいダイアログのハンドラとの間で相互に干渉することなく各ライブラリにおいて一貫して管理するこができる。

一方、例えばインタープリタが外部ファイル読み込みに際して動的にメモリブロックを取得し生成するデータの一部は、処理後に SML や DML ライブラリに移管され、シーン配列やグループ配列などの中に組み込まれる。これらについては、除却・解放に関する役割分担が明確に定められ協調的な処理が行われないと、使用中のメモリブロックの解放や、解放済みのメモリブロックを再度解放しようとすることによるシステム障害や、いかなるライブラリの管理からも遊離したメモリブロックの発生（メモリー・リーク）の原因となる。

Ver.2.09 においては、これを、以下のように整理した。

① IP と SML

インタープリタが、SCENE コマンドを処理した後に、一つのシーンに対応するメモリ・ブロック(s3Scene*)を、SML ライブラリが管理するテーブル(scntab)に登録する。

一つのシーンに関連づけられた光源グループ、効果グループ等の各メモリブロックは、移管されたシーンから参照されているものについては、インタープリタが終了した時点で解放しない。

別の処理において、ユーザーの編集操作によりシーンが削除された時点、新規作成が選択された時点、またはシステム終了時点で、シーンと、それに関連付けられたメモリブロックを解放する。シーンにおいては、一つの光源グループや効果グループを複数のシーンから共通で使用することができる。従って、一つのシーンの解放に際し

ては、それが参照している光源グループ、効果グループ等が、他のシーンから参照されているかどうかを検査し、該当するものがなければ、シーンと同時に解放する。

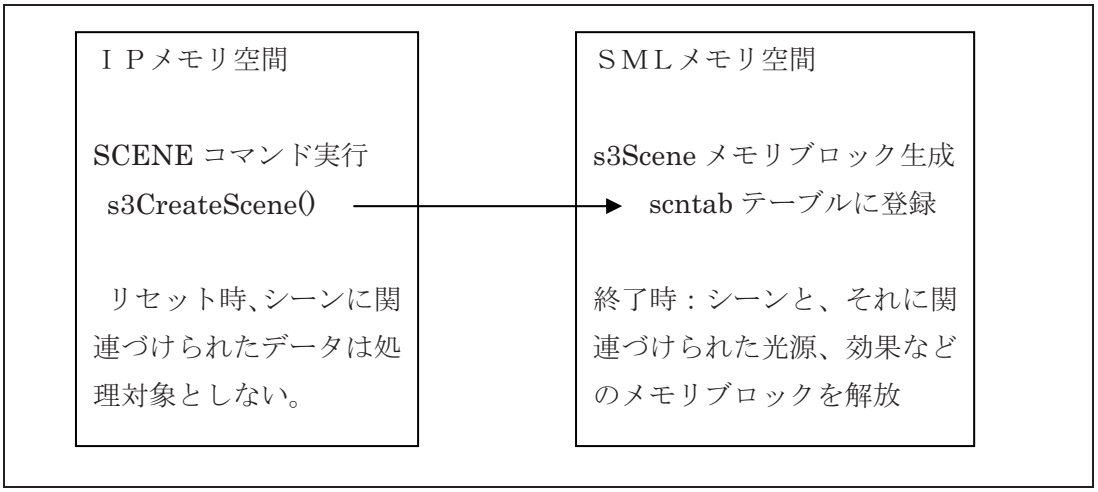


図４－３：IP と SML のメモリ管理

② IP と DML

DML は、グループを登録する d3db、マテリアル ID を登録する mtltab、テクスチャ ID を登録する textab をテーブルとして管理している。

インタープリタが、GROUP コマンド、または FILE コマンドを実行した後に、新たに作成されたメモリ・ブロック(d3Group)を DML が管理するテーブル d3db に登録する。

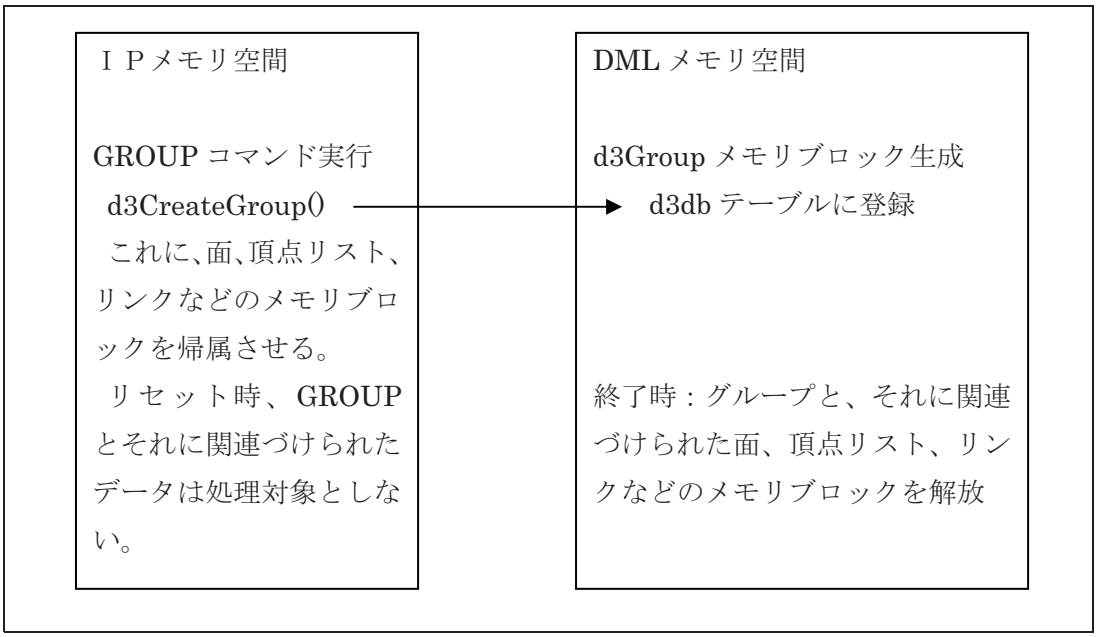


図４－４：IP と DML が管理するメモリ空間の関係

そのグループに属する面は､GROUP_FACE コマンドが実行された時点でコピーを作成し、グループの要素として登録する。それに際して面に帰属する名称、頂点リストなどのメモリブロックも全てコピーを作成する。一つの面（メモリー・ブロック）

は、別のグループから共有されることはなく、常に唯一のグループに帰属する。

LINK コマンドが実行された時点で作成されるメモリ・ブロック(d3Link, d3LinkUP)は、それぞれ関連する親グループ、子グループが保有するリスト（next ポインタでつなぎ）に登録する。

インタープリタが終了した時点で、グループ、リンク以外のメモリブロックを解放する。

マテリアル､及びテクスチャに関しては､インタープリタがMATERIALコマンド、TEXTUREコマンドを処理した後にDMLが管理するテーブルにレコードだけを新規作成する(d3RegisterMaterial 関数等)。この時点では画像等データのロードは実行せずメモリブロックは作成されない。GROUP_MATERIAL、FACE_MATERIAL などのコマンドが実行された時点で､適用対象となる面､グループに ID 番号を記入する。

実際のテクスチャ画像データ等の取得と画像データを格納したメモリブロックの作成は、インタープリタによるファイル読み込みが終了した後、表示段階で G3DRL ライブラリ(g3LoaMaterial, g3LoadTexture 関数)から参照される DBIL ライブラリが行っており、状況に応じた適切な参照先のディレクトリから画像データのロードを行う。その際にキャッシングを行っており、異なる ID で同じテクスチャ・ファイルが複数回参照されても、一つしかメモリブロックを作成しない。また、表示する時間が変更された場合にも、経年劣化するマテリアル等のデータの再ロードを行っている。

以上の機構により、テクスチャを多用した多くのモデルを参照するシーン・ファイルのロード時間やメモリ消費が非常に重くなることも防いでいる。

ユーザーの編集操作、新規作成または終了時点でグループが削除されると、グループ及びそれに帰属するメモリブロックが解放される。その際に、グループにリンクで関連づけられた子グループに関しては、別グループからの参照の有無を検査し、該当するものがなければ、リンクと同時に削除する。マテリアルおよびテクスチャのテーブルは、個々のグループの削除操作に際しては変更せず、新規作成または終了の処理の中で再初期化している。

③DML と DBIL

第３章で解説したように、グループは、リンクによって親子関係が設定され、一つのシーン中に存在するすべてのグループが、最上位のルート・グループを起点とする親子関係の連鎖として構造化されている。表示に際しては、ルート・グループからリンクを辿り、リンクで結び付けられた子グループの全てを表示する。

OpenGL による表示を管理する G3DRL ライブラリは、OpenGL 表示を行う全てのダイアログ画面を配列化したテーブル da[ ]を管理している。一つの表示画面には、様々な表示条件と共に、グループとリンクの全体を辿る起点となるルート・グループ（最上位のグループ）が登録される。描画は、このグループから、リンクを辿って行われる。LSS-G タイ

プのデータを編集する場合には、ルート・グループは通常は不変で、全ての表示画面に共通している。

しかしながら、地物を構成する一つのグループは、複数の親グループを持つことができ、従って、このリンクを辿る方法では、一つのグループが複数回ヒットする可能性がある。実際、リニア配置された街灯や、エリア配置された樹木などは、メモリの上では一つのグループとして格納されているデータが何回も位置を変えて再描画される。したがって、リンクを辿る方法では、特定の属性を有するグループを検索するような処理には無駄が多い。そこで、表示に関連したリンク構造とは別に、DML において、別途テーブルとして全てのグループを管理している（図４－５）。

d3CreateGroup()関数を使用せずに、例えば

d3Group *g = (d3Group*) d3Malloc(sizeof(d3Group));

のような方法で、グループを作成することが可能であるが、このようにして作成されたグループは、DML が管理するテーブルには登録されていないことに注意する必要がある。例えばデータ変換の途上でこのような方法で一時的に作成したグループのメモリブロックを、地物を構成するグループとリンクの中に組み込むような処理が行った場合には、終了時点での一貫性は保証されない。

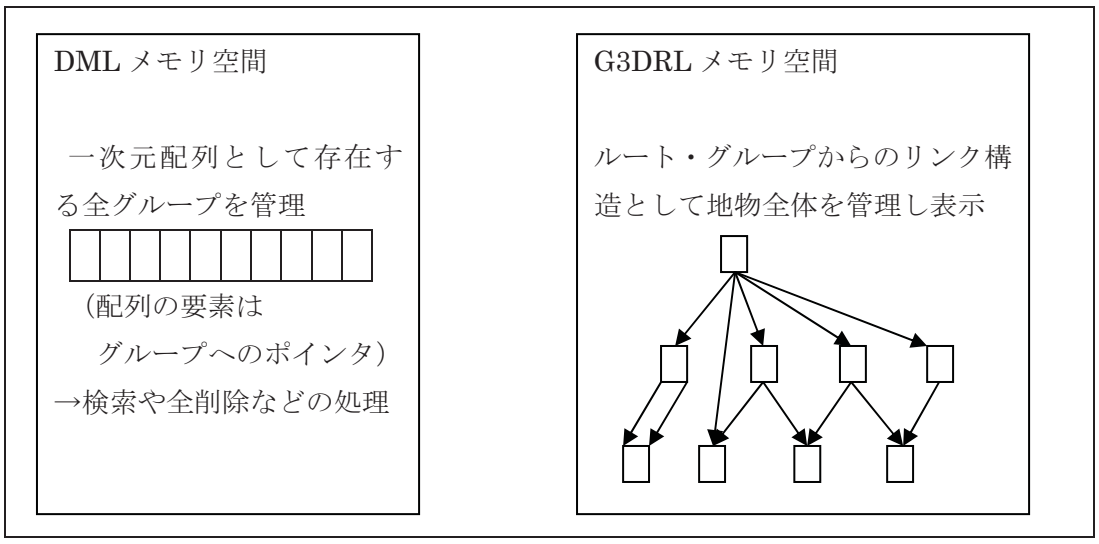


図４－５：DML と G3DRL が管理するメモリ空間の関係

４－２．ライブラリ関数

ライブラリ関数は、マルチ・プラットフォーム（UNIX 系と Windows 系を含む）の環境において、OS に依存しない景観シミュレータのデータ構造の管理と処理を行う諸関数のパッケージである。大きく、概要で述べたように８のライブラリが存在する。

それぞれのライブラリを構成する関数群は、全体のビルドにおいては、共通のヘッダーファイルに登録されているが、ソースコードは複数に分かれている。ソースコードはライブラリ毎に一つのディレクトリに置く一方、ヘッダーは、INCLUDE ディレクトリに一括してある。それぞれのライブラリを構成するソースコードは、コンパイル・リンクによって、それぞれのスタティック・リンク・ライブラリを生成する。

各ライブラリを構成する関数は、所属するライブラリがわかるように、共通のプレフィックスを持つ名称を有している。

以下に、各ライブラリの機能を解説する。　個別のライブラリ関数の解説を付録 B に掲載した。

（１）シーン管理ライブラリ（SML）

①概要

一つの景観画像を生成するために必要な背景・前景画像、三次元モデル、視点、光源、効果、時間を定義したシーンを構築する。更に、多くのシーンを作成し、一連の評価用画像として次々と提示できるように、シーン・リスト（配列）として管理する。

②シーン構造体

最も基本となるシーン構造体は、リスト４－３のように定義されている。

リスト４－３：シーン構造体の定義(sml.h)

```
typedef struct _s3Scene {
        char            *name;
        int             type;
        s3Image         *img_b, *img_f;
        s3Model         *mdl;
        s3Camera        *cam;
        s3LightGroup    *lg;
        s3EffectGroup   *eg;
        s3Time          *time;
} s3Scene;
```

名称（name）：シーン毎にユニークな名称を定義する。

タイプ(type)：　図法（透視図、平面図・・・）、表示モード（テクスチャ、シェーディング、ワイヤーフレーム）等を示すコード。

背景(img_b)・前景(img_f)：モデルの前後に表示する画像。

モデル(mdl)：地物を表す三次元モデル。
視点情報(cam)：視点、注視点、天頂ベクトルなどを記述する。
光源グループ(lg)：最大８の光源から成る光源グループを一つ指定する。
効果グループ(eg)：無制限個の効果を束ねた効果グループを一つ指定する。
時間(s3Time)：時間を記述する構造体を指定する。

名前とタイプ以外は、ファイル名やメモリブロックのアドレスを直接指定するのではなく、定義を記述した構造体（メモリ・ブロック）のアドレスを指すポインタとなっている。一つの定義を、複数のシーンで共通に使用することができる。

実際の運用においては、共通の背景を用いて様々のモデルを比較検討したり、共通のモデルで、様々な視点場からの検討を行ったり、あるいは光源条件（季節・時刻・天候）や築後年数のみを変更したヴァリエーションを一連のシーンとして用意し、評価セッションに臨む。このような様々のヴァリエーションの作成に際して、変更のないデータや設定内容に関しては、複数のシーンで共用することが、このような間接的な参照により可能となっている。

シーンの編集操作、あるいはＬＳＳ－Ｓ形式のファイルをロードすることにより、一連のシーンから成る表示用のデータ（メモリ・ブロック上の配列）を構築する。このブロックのアドレスを示すスタティック変数をＳＭＬライブラリが管理しており、通常はこのメモリブロックに対する様々な処理をライブラリ関数で実行することにより、シーンに対する様々な処理を行う。

③ ソースコード(s3xxx.c の名称)

ＳＭＬライブラリは、リスト４－４に示したソースコードから成る。

リスト４－４：SML ライブラリのソースコード一覧

S3sml.c	(構築)
S3set.c	(データのセット)
S3get.c	(データの参照)
S3delete.c	(データの削除)

（２）データ管理ライブラリ（DML）

①概要

三次元の地物（形状、表面の光学特性、属性）をデータとして構築する。更に、複数の地物の構成要素を、リンクで親子関係に結び、地物全体を構造的にデータ化し、管理することにより、様々な編集操作を下支えする。

モデルの記述

②グループ構造体

グループは、名称、ID やマテリアル、テクスチャ、親の数、子の数等の属性と、面

のリストを持っている

リスト４－５：グループ構造体の定義(dml.h)

```
struct _d3Group {
/*private:*/
        int grpid;
        int dbno;
        int num_u;/* numbers of parents */
        int num_d;/* numbers of children */
        int material;
        int texture;
        double xmin, xmax, ymin, ymax, zmin, zmax;/* bounding box */
        d3Link *link;
        d3LinkUp *lup;
        /*INT CONSTANT OOCURS HERE*/
        d3Face   *f;
        /**/
        void *display;
        void *user[D3_USERDATA_MAX];
/*public:*/
        char *name;
        char *kind;
        int type;
};
```

③面の構造体

面は、名称を持たず、頂点リスト、マテリアル、テクスチャ、法線ベクトル、カラー等の情報を持つ。next のポインタでつながれたリストとして、グループに所属する。その先頭に当たる面を、d3Group の f メンバが指し示している。

リスト４－６：面の構造体の定義(dml.h)

```
struct _d3Face {
/*private:*/
        d3Vertex *vl;
        d3Face *next;
/*public:*/
        unsigned long va_f;
        unsigned long va_v;
        int vnum;
        int material;
        int texture;
        float n[3];
        float c[4];
        int display;
        int shape; /* 1996.1.30 D3_SHP_xxxx */
};
```

③頂点リストの構造体

頂点は、座標、法線ベクトル、テクスチャ座標、カラーを有する点をリスト化した配列（メモリ・ブロック）として定義され、面に帰属する。

リスト４－７：頂点の構造体の定義(dml.h)

```
struct _d3Vertex {
/*private:*/
/*public:*/
        unsigned long va;
        double v[3];
        float n[3];
        float t[2];
        float c[4];
};
```

④ソースコード（d3xxxx.c の名称）

リスト４－８：DML ライブラリを構成するソースコード

d3dml.c	(データ構築)
d3malloc.c	(メモリ管理)
d3mat.c	(リンク・マトリックス計算)
d3pick.c	(オブジェクトの選択)

（３）データベース管理ライブラリ（DBIL）

①概要

多くは、景観データベースの検索機能において用いられるが、景観シミュレータの基幹部分(sim.exe)のビルドにおいても、画像ファイルの処理など、部分的な機能を利用している。

②ソースコード（dbxxx.c の名称）

リスト４－９：DBIL ライブラリを構成するソースコード

dbdata.c	(データベース検索)
dbms.c	(構築)
dbread.c	(データベース検索)
image.c	(SGI 形式の画像の読み込み)
isave.c	(SGI 形式の画像の保存)

（４）動作環境管理ライブラリ（ENV）

①概要

システムの動作環境を表す様々の環境変数を管理する。システム起動時に、環境設定ファイル kdbms.set を読み込むことによって初期化される。

②ソースコード(e3xxx.c の名称)

リスト４－１０：ENV ライブラリを構成するソースコード

e3env.c	(環境変数の管理)
e3kerja.c	(ユーザー作業環境)

（５）画像表示処理ライブラリ（G3DRL）

①概要

主に、三次元モデルを、各種の OpenGL 画面に描画する機能を提供している。表示の

ほかに、地面の高さを用いて処理を行う場合には、デプス・バッファを用いたOpenGLの描画機能を用いて地面の高さを抽出している。OpenGLによる描画は、メイン画面における透視図の表示や、配置・コピー、平面生成、可視範囲解析など、地物の平面図表示（オルソ画面）を主要部分とするダイアログにおいて用いられる。この他に、テクスチャ選択画面など、いくつかの編集ダイアログでは、OpenGLの小さな画面を用いて、ユーザーが選択しようとしているカラーやテクスチャ等を表示している。

②ソースコード(g3xxx.c、wg3xxx.c等の名称)

リスト４－１１：Ｇ３ＤＲＬライブラリを構成するソースコード

g3drl.c	(メイン画面、各編集画面への透視図・平面図の描画)
g3font.c	(文字の表示)
g3load.c	(マテリアル、テクスチャのロード)
g3road.c	(デプス・バッファによる道路法面の形状生成)
g3sdl.c	(マテリアル、光源のダイアログの色球の描画)
g3steel.c	(型鋼のダイアログの見本の描画)
g3stenci.c	(ステンシル・バッファの利用)
g3utl.c	(テクスチャ自動貼付等)
StereoSight.c	(ステレオ表示のパラメータ管理)
wg3.c	(メイン画面、オルソ画面の初期化
wg3swal.c	(ダイアログの小さなOpenGL画面の初期化)

（6）ファイル入出力ライブラリ（IP）

①概要

外部ファイルの入出力を行う。

②ソースコード(i3xxx.c等の名称)

リスト４－１２：ＩＰライブラリを構成するソースコード

i3ip.c	(構築)
i3ipbase.c	(コマンドの解析)
i3ipcall.c	(外部ファイル参照)
i3iperr.c	(エラー処理)
i3ipex.c	(外部関数処理)
i3iplssg.c	(LSS-Gのデータ構築)
i3iplsss.c	(LSS-Sのデータ構築)
i3ipout.c	(ファイル出力)
fopen_.c	(URLアクセス)

（7）図形演算処理ライブラリ（U3）

①概要

様々なダイアログにおいて必要となる図形演算処理のうち、共通性の高いものをまとめたライブラリである。

②ソースコード(u3xxx.c等の名称)

リスト４－１３：Ｕ３ライブラリを構成するソースコード

u3cross.c	(二次元図形演算)

u3cut.c	(二次元図形演算)
u3dml.c	(バックアップ・リストア)
u3grid.c	(メッシュデータ)
u3hsvrgb.c	(表色系)
u3ofs.c	(断面)
u3rand.c	(乱数)
u3same~1.c	(点の一致)
u3slant.c	(法面)
u3spline.c	(スプライン曲線)
u3trans.c	(メッシュの座標変換)

（8）メッセージ管理ライブラリ（Z3)

①概要

エラー、警告、情報提供、選択肢の提示などを行う。

②ソースコード(z3xxx.c 等の名称)

リスト４－１４：Ｚ３ライブラリを構成するソースコード

z3err.c	(環境変数の管理)
z3nt.c	(メッセージボックス表示)
z3unix.c	(メッセージウィンドウ表示)

４－３．アプリケーション・ライブラリ関数

アプリケーション・ライブラリは、各種ライブラリと、次に述べるダイアログ・ハンドラの中間に位置し、システム全体を管理している。これには、以下のような機能・役割がある。個別のライブラリ関数に関する解説は、付録 B に掲載した。

（1）システム管理機能

①システム全体の初期化処理を行う

②システムの状態をモニタリングし、問い合わせに回答する。

③システム全体の終了処理を行う

④エラー対策を行い、Z3 ライブラリにメッセージを送出する。

ソースコードは、

Common.c

Dataope.c

の二つの長いファイルにほぼ集約されている。

（2）ライブラリ支援機能

各ライブラリの機能を動かす上で必要となる、Windows 固有の条件に対応する機能を細くする。例えば、ファイルやディレクトリの記述法が、UNIX 系と Windows 系では異なっている。また、グラフィックス表示画面の初期化の方法なども、OS により異なっている。

①OpenGL 画面の初期化

pixel.c

wg3.c

wg3swal.c

②画面の印刷処理

b3bitmap.c

copy.c

print.c

（3）ダイアログ・ハンドラ支援機能

複数のダイアログ・ハンドラ（C++のクラス）に共通する機能を提供する。操作・処理の前後での画像のバックアップとリカバリーを行う。また、外部プロセスの起動を行う。

Multilang.cpp（多言語処理）

Pembantu.c （ヘルプ表示）

Genshi.c （原始図形計算）

Fopen_.c （URL アクセス）

（4）個別単発的な特殊演算処理機能

数値計算処理やデータ形式変換処理などで、一つのライブラリを形成する程の規模ではない専門性の高い計算処理は、アプリケーション・ライブラリとして分類している。画像の視点抽出計算や、異なる色彩の表色法の間での数値換算処理などがこの中に分類されている。

①異なる形式でのファイル出力

dxfout.c

vrmlout.c

fire.c

②カラー表色系の相互変換

cnvcol.c

③ビットマップによる動画作成等

bmp2dib.cpp

AviComposeWnd.cpp

MyImage.cpp

MyAvi.cpp

dib31.cpp

④画像視点抽出の解析計算

kamera2.c

４－４．ダイアログ・ハンドラ

Windows 固有のダイアログ画面に対するハンドラ・ルーチン群は、ユーザーによる画面操作を通じての様々なリクエストに対応する処理を実行する。これらの処理プログラムは、ＯＳに大きく依存しており、ステップ数の多くは、エラー対応処理やメッセージ、ヘルプなどを含め、ユーザーによる想定外の操作に対する対応に充当されている。

ダイアログの画面構成（デザイン）の大半は、リソースによって定義され、開発環境の中ではＧＵＩ環境でデザインされている。多言語環境において、画面レイアウトには、漢字系（日中韓）のダイアログと、アルファベット系のダイアログの二種類のデザインを用意してあり、選択された言語がいずれに属するかによって選択的に表示を行っている。ユーザーの操作は、マウスクリックおよびキーボード操作によってインプットされる。一つの画面に関して、初期化処理、様々な操作に対する応答と、終了時の処理から成るハンドラを、Ｃ＋＋のクラスとして構成している。

本節においては、各操作画面ごとに、主要な機能を解説した上で、デザインを定義しているリソース、これを処理する関数群をパッケージ化したクラスと、それを定義しているソースコードファイル名を示す。更に、必要に応じて、それぞれのダイアログの呼び出し元や、それぞれから起動するダイアログとの関係、要求された機能を実行するために使用されるライブラリ関数との関係を解説する。

（1）メイン画面

様々な図法（透視図、平面図、立面図その他）で地物を表示すると共に、多くの編集機能をメニューから起動する。下部にシンプルな視点移動ボタン、シーン切換ボタンがある他、OpenGL の表示画面におけるマウス操作で操作対象となるオブジェクトを選択し、情報を見たり編集を加える。

また、Ver.209 においては、マウスの右ボタンを押したままドラッグすることにより、高速でスムーズな視点移動を行う。枝分かれバージョンの統合により、ステレオ表示、影の表示を一つのバージョンで使い分けることができる。更に、多言語機能により、走りながら、メニュー、ダイアログ、ヘルプ、エラーメッセージの言語を切り換えることができる。

図４－６：メイン画面

構成要素

①メニュー：IDR_MAINFRAME　ハンドラ：CMainFrame (mainfrm.cpp)

メニュー構成
［ファイル］
＋［新規作成］
＋［LSS-S］　　LSS-S（シーン）データを新規作成
＋［LSS-G］　　LSS-G（ジオメトリ）データを新規作成
＋［開く LSS-S］　　LSS-S（シーン）ファイルを開く
＋［開く LSS-G］　　LSS-G（ジオメトリ）ファイルを開く
＋［読み込み LSS-G］　LSS-S（シーン）の編集でモデルを読み込む

＋［JPEG 形式で出力］表示されている画面を JPEG 形式で保存する
＋［上書き保存］　　読み込んだファイルに LSS-S または LSS-G を上書き保存する
＋［名前を付けて保存］編集中のデータ(LSS-S または LSS-G)に名前を付けて保存する
＋［最適化保存］　　LSS-G（ジオメトリ）データを最適化処理付きで保存する
＋［作業環境設定］　ファイルを取得／保存する作業用フォルダを指定する
＋［報告書執筆］　　現在編集中のデータに関するサマリーをテキスト出力する
＋［ファイル整理］　編集中のデータに関連するファイルを指定場所にコピー集約する
＋［スキャナー入力］　スキャナー入力のためのアプリケーションを起動する
＋［画面印刷］　　表示されている画面イメージをプリンターに出力する
＋［高解像度印刷］　プリンターの解像度で印刷する
＋［印刷プレビュー］　高解像度印刷をプレビューする
＋［プリンタの設定］　プリンターの選択と印刷条件の設定を行う
＋［使用言語の選択］　メニュー、ダイアログ、メッセージ、ヘルプの言語を選択する
＋［アプリケーションの終了］　終了する

［編集］
＋［他選択］
＋＋［次候補］　　クリックした点に係る、もう一つ裏側のオブジェクトに選択を変える
＋＋［親グループ］　選択したオブジェクトの親グループを選択する
＋＋［子グループ］　選択したオブジェクトの子グループを選択する
＋＋［情報を見る］　選択したグループの諸元素を表示・編集する
＋［選択取り消し］　選択・強調表示を取り消す
＋［選択モード］
＋＋［グループ］　　グループを選択する（通常のモード）
＋＋［面］　　面を選択する（マテリアルの編集対象として）
＋＋［同色面］　　同色面を選択する（同上）
＋［移動/回転/スケール］選択したオブジェクトの上方リンクのマトリクスを編集
＋［配置/コピー］　位置、線形、領域を指定して、オブジェクトを添加する
＋［削除］　　選択されているオブジェクトを削除する
　　オブジェクトが選択されていない場合には、面が無く、下方リンク
　　ンクもないグループ（「幽霊」と表記）を全て削除する
＋［視点設定］
＋＋［視点座標］　　視点座標・注視点座標等の視点情報を数値で表示・編集する
＋＋［可視範囲解析］選択したオブジェクトが見える範囲を解析する
＋＋［視点設定］　　平面図上の指定した位置に視点を設定する
＋＋［移動経路設定］移動経路と注視方向を指定し、動画を表示・記録する

＋［マテリアル／テクスチャ］選択したオブジェクトの表面仕上（光学特性）の編集
＋［エフェクト］シーンの効果（霧、ステレオ表示、影等）の設定
＋［画像視点抽出］背景画像の視点パラメータを求め、立体を位置合わせする
［表示］
＋［視点］
＋＋［全体視界］全てのオブジェクトが見える位置に移動する
＋＋［初期表示］最初の視点（シーンに定義された視点）に戻る
＋［透視図］　透視図として表示する
＋［平面図］　平面図（配置図）として表示する（真上から見る）
＋［南立面］　Y 軸の負から正の方向に見た立面（正面図）
＋［東立面］　X 軸の正から負の方向に見た立面（側面図）
＋［北立面］　Y 軸の正から負の方向に見た立面
＋［西立面］　X 軸の負から正の方向に見た立面
＋［経年変化］築後日数を設定する
＋［表示モード］
＋＋［テクスチャ表示］テクスチャ付で面を表示する
＋＋［シェーディング表示］テクスチャを無視して面を表示する
＋＋［ワイヤーフレーム表示］辺・稜を表示する
＋＋［オプション設定］
＋＋＋［地面のみ表示］地面の属性をもつオブジェクトだけを表示する
＋＋＋［地面テクスチャ表示］地面の属性をもつオブジェクトだけテクスチャ表示する
　　（大規模な市街地の編集などにおいて、通常の表示が遅い場合などに使用）
＋＋［オプション解除］オプションの設定を解除する
＋［グリッド］グリッドの表示を ON／OFF する
＋［アンチエリアシング］画面に対して斜の境界線を滑らかに表示する設定を行う
＋［影］
＋＋［影なし］影を表示しない
＋＋［地面以外（通常）］地面以外の物体の影を表示する
＋＋［全地物］全ての地物の影を表示する
＋＋［選択物］選択物の影を表示する
＋＋［影設定］影表示のパラメータ、詳細設定を行う
＋［ステレオ］ステレオ表示の有無とパラメータを設定する
＋［高速表示］表示の高速化のオプションを設定する
＋［シーン選択］シーンを一覧表示し、選択する
［形状生成］
＋［原始図形］

＋＋［直方体］直方体の位置と大きさを指定し、形状生成する
＋＋［球］中心位置と半径を与え、形状生成する
＋＋［円柱］上円と下円の中心位置と半径から形状生成する
＋＋［円錐・円錐台］上円と下円の中心位置とそれぞれの半径から形状生成する
＋＋［角柱］上底と下底の中心位置と半径・角数から形状生成する
＋＋［角錐・角錐台］上底と下底の中心位置とそれぞれの半径、角数から形状生成する
＋＋［掃引体（１面)］断面と、移動経路から立体を形状生成する
＋＋［掃引体（２面)］二つの面（上底と下底）から形状生成する
＋＋［平面］平面、及び押し出しによる立体、面の穴あけ等の編集操作を行う
＋＋［線］　頂点座標の配列から線分、折れ線を形状生成・編集する
＋［基本構成要素］
＋＋［形鋼］典型的な断面の鋼材を形状生成する
＋＋［橋］５種類の橋を配置する
＋＋［草］ランダムな形状として草を生成（未完成）
＋＋［水面］（廃止）テクスチャ編集画面に統合
＋＋［ブロック］（廃止）テクスチャ編集画面に統合
＋＋［道路］断面と中心点軌跡から立体を形状生成する
＋＋［河川］断面と中心点軌跡から立体を形状生成する
＋＋［法面］テクスチャを貼る
＋［道路法面生成］地形と道路断面、法面パラメータから道路と法面を形状生成する
＋［オプション］ユーザー定義によるパラメトリックな部品を選択し、形状生成する
　屋根、階段、正多面体、正面テクスチャ付きビル、各種コンバータ等
＋［プラグイン］プラグイン DLL を選択し、形状生成・編集する
　トンネル、園路、地形編集機能など
＋［シャッター］表示されてい状況を一つのシーンとして記録する
＋［ヘルプ］メイン画面のヘルプを表示する

②ポップアップメニュー：IDR_MENU_POPUP　ハンドラ：CDrawFrm (drawfrm.cpp)
　メイン画面でオブジェクトを選択した状態で、画面を右クリックすると表示される。

メニュー構成
［情報を見る］
＋［情報を見る］ダイアログ(9)を開く
＋［幾何属性］
＋＋［立体情報］ダイアログ(52)を開く
＋＋［面情報］　ダイアログ(51)を開く
＋＋［三次元ソート］オブジェクトの全ての頂点座標をソートし一覧表示する
＋［力学的属性］

＋＋［質量］ダイアログ(64)を開く
＋［化学的属性］
＋＋［炭素含有量］ダイアログ（65）を開く
＋［データベース参照］ダイアログ（66）を開く
＋［住宅情報］ダイアログ(54)を開く
＋＋［詳細を見る］単体の内部俯瞰等の LSS-G ファイルを別プロセスで表示
＋＋［眺望を見る］選択した物件からの眺望を表示
＋＋［物件情報］ダイアログを開く
［選択変更］
＋［次候補］同じクリック地点にある一つ裏側のオブジェクトを選択する
＋［親グループ］選択したオブジェクトの親グループを選択
＋［子グループ］選択したオブジェクトの最初の子グループを選択
＋［選択解除］
［編集する］
＋［移動/回転/スケール］ダイアログ(10)を開く
＋［削除］選択したオブジェクトを削除
＋［このグループを保存］選択したオブジェクトを LSS-G 形式で保存
＋［マテリアル］ダイアログ(21)を開く
＋［色々なマテリアル編集］
＋＋［オリジナル版］ダイアログ(21)を開く
＋＋［グラフィックなマテリアル編集］ダイアログ(25)を開く
＋＋［グラフィックなテクスチャ編集］ダイアログ(26)を開く
＋＋［様々なカラー編集］ダイアログ(28)を開く
［視点場］
＋［これを注視］選択されたオブジェクトの中心に注視点を設定する
＋［ここから見る］選択されたオブジェクトの中心に視点を設定する
＋［ここから見返す］選択されたオブジェクトの中心から現在の視点を注視する
［BOOL 演算］
＋［これを刃物として選択］図形演算の適用オブジェクトを選択する
＋［選択された刃物でこれを切る］図形演算の対象オブジェクトを選択し、掘削する
＋［　取り出す］図形演算の対象オブジェクトを選択し、掘削される部分だけを取り出す
＋［　へこませる］対象オブジェクトを掘削した後に、新たな表面を付け加えて閉じる
＋［　もぎ取る］掘削される部分に新たな表面を付け加えて立体として閉じる
＋［独立する］選択したオブジェクトを親グループから分離し、ルートに置く
＋［統一する］配下の全ての子グループの面を、選択したグループに直接帰属させる。
［オプション］既存のパラメトリックな部品のパラメータを再編集する

メニュー項目の多くに関して、現在のシステム状態により選択可／不可を切り替えている。その条件は、編集対象が LSS-S（シーン）であるか、LSS-G（モデル）であるかで大きく分かれる。表示状態が透視図であるか、平面図・立面図系であるかでも分かれる。更に、編集ダイアログが開いた時点で、同時に開かれると不具合が生じる別の編集ダイアログを選択不可としている。

このようなメニューの選択可／不可を系統的に切り替えるために、CMainFrame クラス(mainfrm.cpp)の関数として、SetSensitiveXXX(int mode)という一群の関数を用意してある。ダイアログの開始や終了に際して、引数として TRUE、FALSE 等を指定してこれらの関数を呼び出すことにより、上記の目的を果たしている。

③下部ダイアログ リソース : IDD_DLG_KBD ハンドラ : COpeDlg::m_opeDlg(opedlg.cpp)
④OpenGL 画面 : CDrawFrm m_pDraw (drawfrm.cpp)
⑤ヘルプ : mainfrm.txt

画面を左クリックすることにより、表示の中からオブジェクトを選択し、次に各種の編集ダイアログを開いて、選択したオブジェクトを編集する、という作業手順が、本システムの操作においてきわめて一般的である。画面左クリックが行われた時点で、上記④の OpenGL 画面をコントロールしている CDrawFrm クラスの、OnLButtonUp 関数を起点として、Ｇ３ＤＲＬライブラリの機能を用いて、クリックした画面上のポイントに表示されている面を探索し、結果を、g3SelPath 構造体のデータとして格納する。この中には、手前に表示されている面の裏側に隠れている面、それらが帰属するグループ、それらの親グループの全てが含まれる。そして、最も手前に表示されている面を含むグループを、強調表示する。この状態でメニューの、[編集][他選択][次候補]が選択されると、その面の裏側に隠れている別の面が帰属するグループに選択対象が変更される。

各編集ダイアログの側では、g3GetSelPath 関数を用いて、現在の選択状態を示す g3SelPath 構造体のデータにアクセスし、編集対象とすべきオブジェクトを特定する。

（2）バージョン情報

メイン画面の右上のヘルプのサブメニューであるバージョン情報から起動し、現在のバージョンを表示する。メイン画面のタイトルに示されたバージョンは、多言語機能により外部化されているため、セットアップが不適切で多言語対応の XML が古いままとなっている場合には、古いバージョンが表記されるのに対して、このバージョン情報で表記する、例えば 2.09 といった数字は、ソースコード中のリソースで表記されたバージョン番号をキーワードとして言語に応じた張替を行うため、旧バージョン番号に張り替えられることはない（意図的に偽の XML ファイルを作成するのでない限り、キーワードが不一致となり、ソースコード中の数字がそのまま表示される）。

ログボタンで、ヘルプと同様の機構により、メモ帳で履歴を表示する。

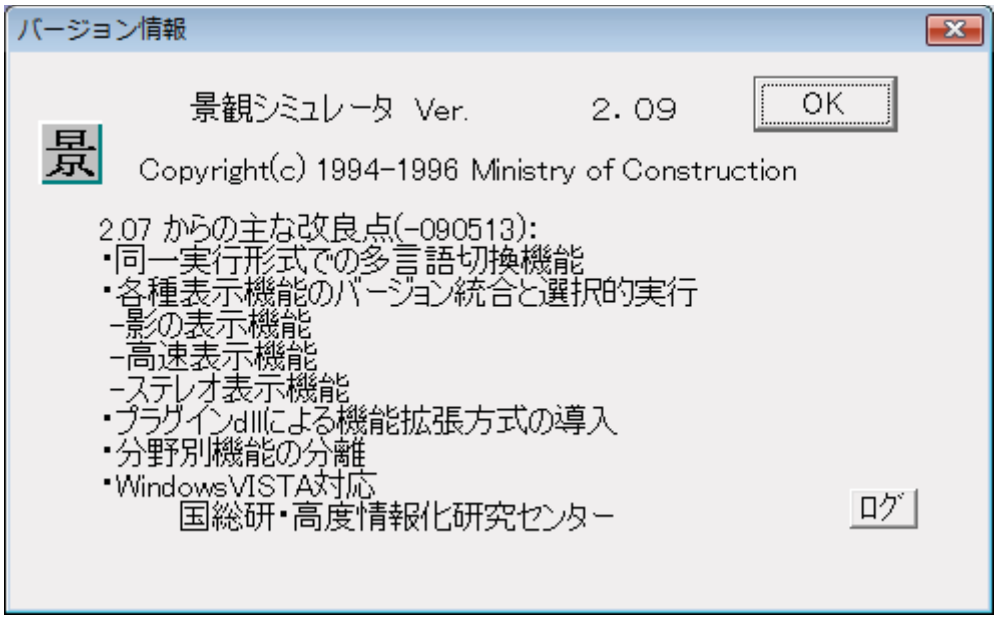


図４－７：バージョン情報表示画面

リソース：IDD_ABOUTBOX

ハンドラ：CAboutDlg(sim.cpp)

ヘルプ：log.txt（ログ　ボタンで開く）

（３）確認その１

重大な処理を実行する前に、ユーザーの再確認を求めるために使用する。二択の選択を簡単に求める場合にも使用可能である。下の例は、NT3.51 以前の WINDOWS 系 OS では使用できないダイアログを表示するに先立って選択を行っている。

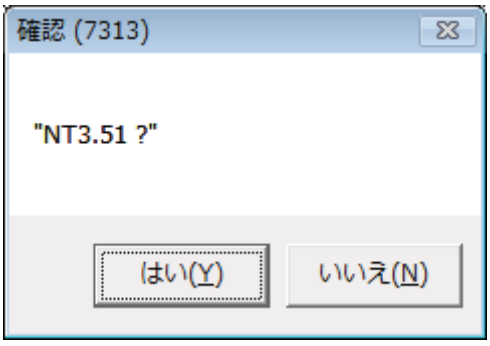


図４－８：確認画面１

z3Confirmation(メッセージ番号)　メッセージ番号は、ERR_MSG.txt による

ERR_MSG.txt に登録する各メッセージは、

コード、番号、テキストの構成となっている。

例「C 7001 データ（%s）を保存しますか？」

コードには、E（エラー）、W（警告）、I（情報提供）、C（確認）は４種類があり、C では、YES か NO の二つの選択ボタンを表示する。結果に基づいて、処理が分岐する。

（4）ファイル選択ダイアログ

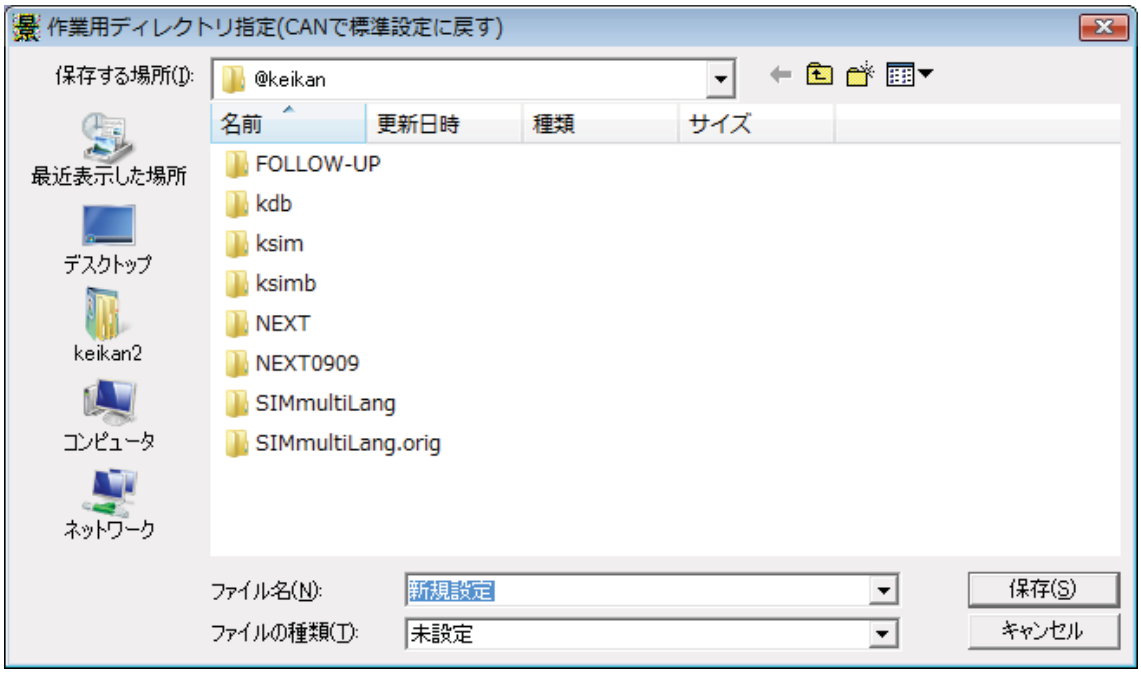


図４－９：ファイル選択画面

CFileDialog　(MFC の組み込みクラス)

ファイルを開く場合、名前を付けてファイルに保存する場合に、ファイル名をユーザーに選択させるために用いる。（5）が使用できない古い OS の場合には、ディレクトリを選択する場合にも用いる。

（5）ディレクトリ選択ダイアログ

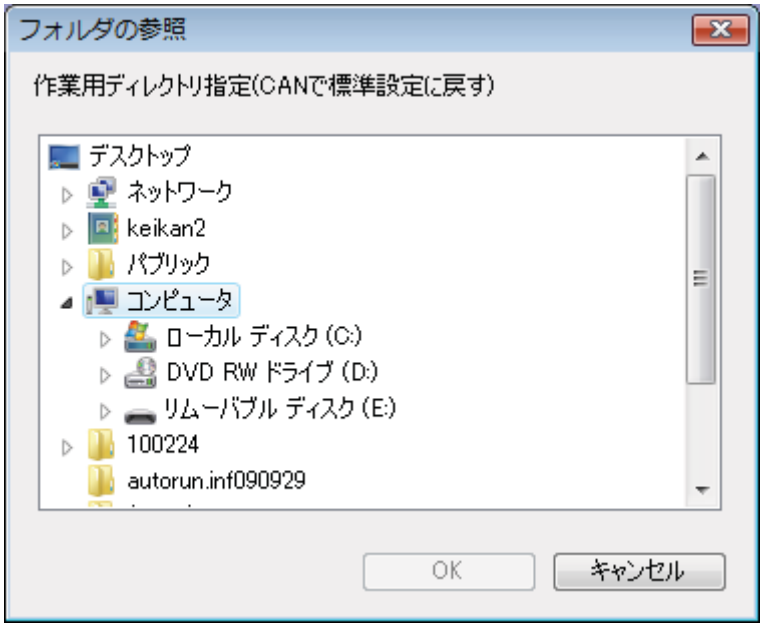


図４－１０：フォルダ選択画面

システム　::SHBrowseForFoler() (OS 組み込み関数)

ディレクトリを選択する場合に用いる。NT3.51 以前の OS では使用できない。

（６）エラー・警告・インフォメーション

あらゆる局面で、ユーザーに情報提供する必要が生じた場合に、現在選択されている言語でメッセージを出力する。原則として、「エラー」は、システムの側に起因する障害、「警告」はユーザーの誤操作に起因する障害、「インフォメーション」は障害ではなく、ユーザーに、現在の状況や操作の結果に関する理解を促すための情報を提示する。

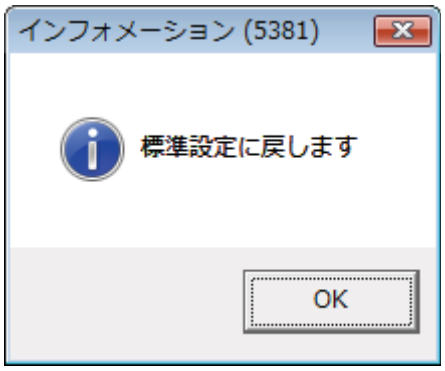


図４－１１：メッセージ表示画面

z3Message(番号)　　番号は、ERR_MSG.[言語コード].txt による（上記の例では、ERR_MSG.ja.txt)。

（７）確認その２

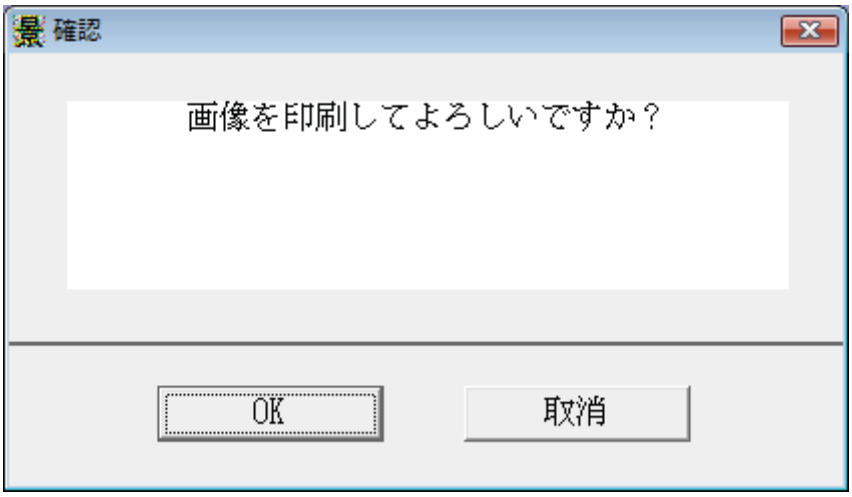


図４－１２：確認画面２

リソース：IDD_KAKUNIN　ハンドラ：CKakunin (kakunin.cpp)

上記（３）の z3Confirmation とほぼ同じ機能であるが、より長いテキストを表示することができる。

（８）言語選択

ダイアログのメニュー、コントロール（ボタンなど）の表示、メッセージ、ヘルプ等に

使用する言語を選択する。選択可能な言語は、Language ディレクトリの下にあるサブ・ディレクトリ名を以て認識する。ディレクトリ名は、ISO639 の言語コード（アルファベット２文字、第 11 章参照）を用いる。

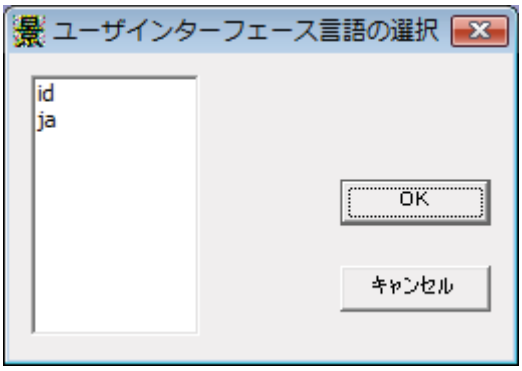


図４－１３：言語選択画面

リソース：IDD_DIALOG_SEL_LANG ハンドラ：CDlgSelLang(dlgsellang.cpp)

（9）選択したオブジェクトの情報を見る

メイン画面で選択し強調表示されているグループ（より正確には一つのリンクの子グループ）に関する情報を表示する。

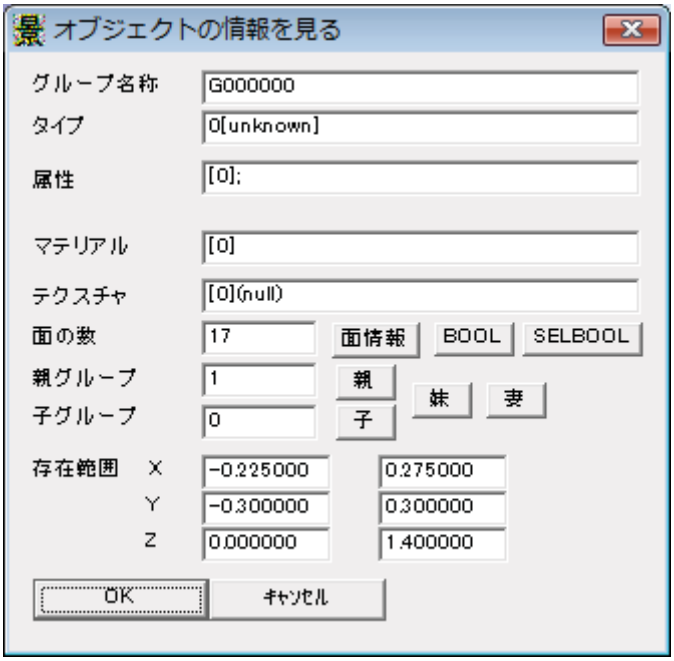


図４－１４：オブジェクト情報表示画面

リソース：IDD_DLG_INFO　ハンドラ：CInfo (info.cpp)

（１０）移動・回転・スケール

メイン画面で選択し強調表示されているグループ（より正確には一つのリンクの子グル

ープ）の上方リンクに定義されたマトリックスを変更することにより、オブジェクトの位置・スケール・角度を変更する。オブジェクトがサミット直下のグループであった場合には、ルートとのリンク・マトリクスを単位マトリクス以外のものに設定することは不適切であるため、中間に新たなグループを一つ挿入して、これにマトリクスを設定する。その場合には、メッセージを発して、ユーザーにその旨を伝える。

図４－１５：移動・回転・スケール編集画面

リソース：IDD_DLG_MOVE　ハンドラ：CEditMove (editmove.cpp)

ヘルプ：editmove.txt

（１１）配置・コピー

LSS-G ファイルを選択して、位置を指定し、モデルに追加する。

配置方法は、単体、リニア、エリアの３種類から選択する。単体の場合には、配置する位置を平面図表示画面上のマウス・クリックで次々と指定する。リニア配置では、平面図上で経路（例えば街路樹であれば、道路脇の歩道上）を指定し、配置間隔を指定した上で一括配置する。エリア配置では、平面図上に領域（例えば公園）を指定し、配置密度を指定した上で一括配置する。

配置する対象物は、５種類まで同時に選択した上で、指定した比率で乱数により混合しながら配置することができる。

配置する対象物の選択方法は、配置オブジェクト選択先選択ダイアログ(14)により、５通りの方法の内のいずれかを用いる。

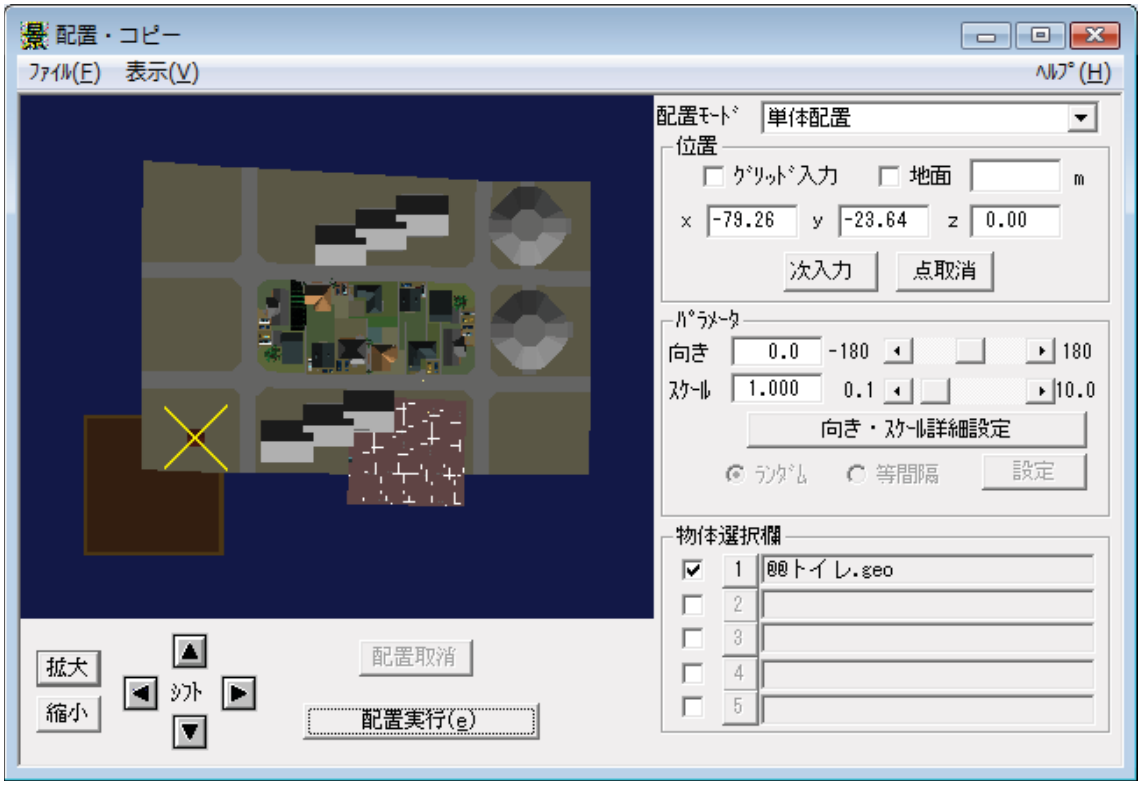


図４－１６：配置・コピー画面

① メニュー：IDR_MENU_HAICHI

メニュー構成：
［ファイル］
＋［終了］終了する
［表示］
＋［グリッド］１m 固定のグリッドの表示の ON／OFF
＋［メジャー］スケール自動設定による縮尺の表示の ON／OFF
＋［全体視界］存在するオブジェクト全てを表示する平面図表示範囲とする
＋［上下範囲］平面図表示する高さ範囲を設定する
＋［縦横範囲］平面表示する水平範囲を設定する
＋［表示モード］
＋＋［テクスチャ表示］平面図をテクスチャ表示する
＋＋［シェーディング表示］平面図を面で表示する
＋＋［ワイヤーフレーム表示］平面図を辺・稜で表示する
＋＋［オプション設定］
＋＋＋［地面のみ表示］地面の属性があるオブジェクトだけを表示する
＋＋＋［地面テクスチャ表示］地面の属性があるオブジェクトだけテクスチャ表示する
＋＋［オプション解除］オプションの設定を解除する

② GL 画面：COrthoVW m_orthoView (orthovw.cpp)
物体を配置する位置や、リニア配置の線形、エリア配置の領域等を指定する
③ 右側ダイアログ：IDD_DLG_HAICHI_R
ハンドラ：CHaichiWnd (haichiwn.cpp)、メンバ変数：CDialogBar::m_Prm_Bar
物体選択欄のボタン（１～５）でポップアップ・メニューを表示する
「向き・スケール詳細設定」ボタンで、詳細設定ダイアログを開く
「設定」ボタンで、配置パラメータ設定ダイアログを開く
④ 下側ダイアログ：IDD_DLG_HAICHI_D
ハンドラ：CHaichiWnd(haichiwn.cpp)、メンバ変数：CDialogBar::m_Ctl_Bar
⑤ヘルプ：haichi.txt

（１２）配置詳細設定パラメータ

配置ダイアログ（１１）の配置の向き・スケール詳細設定ボタンから起動する。単体は位置、リニア配置、エリア配置における、オブジェクトの向きとスケールを設定する。

図４－１７：配置詳細設定パラメータ編集画面

リソース：IDD_DLG_HAICHI_DTL　ハンドラ：CHaichiDtlDlg(haichidt.cpp)
ヘルプ：haichdt.txt

（１３）配置パラメータ

配置ダイアログ（１１）の配置の設定ボタンから起動する。リニア配置、エリア配置における向きやスケールのゆらぎ、ばらつきを設定する。

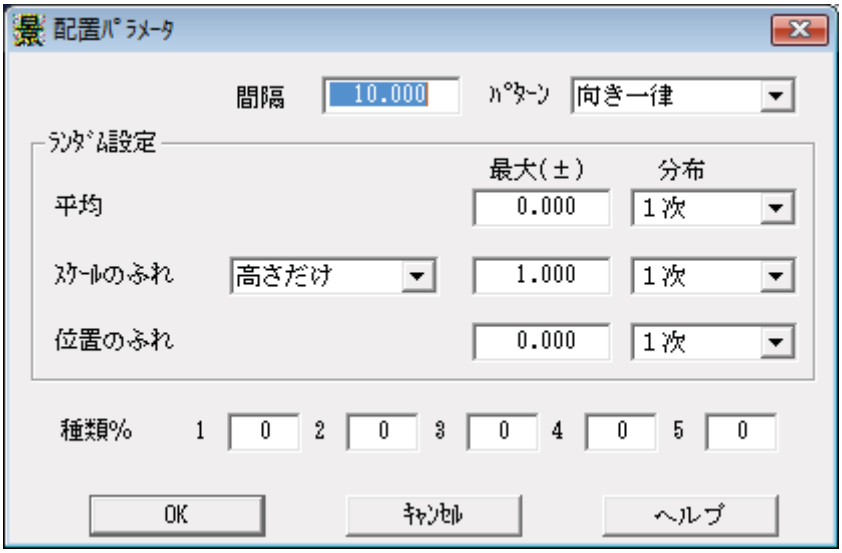


図４－１８：配置パラメータ編集画面

リソース：IDD_DLG_HAI_PRM　ハンドラ：CHaichiPrmDlg (haichipr.cpp)

ヘルプ：haichipr.txt

（１４）配置するオブジェクトの取得先選択

配置するオブジェクトの選択方法を選択する。既にモデル中に存在するグループをコピーして配置する「画面セレクション」、直接ファイル名を指定する「ＬＳＳ－Ｇ」、データベースから検索する「景観構成要素」「景観材料」および「外部関数」(パラメトリックな部品：原始図形及びユーザー定義による部品)が選択可能である。

図４－１９：配置オブジェクト取得先選択画面

リソース：IDD_DLG_HAI_SEL　ハンドラ：ChaiselDlg (haiseldl.cpp)

（１５）視点座標

視点情報(CAM)を構成する視点、注視点、天頂ベクトル、焦点距離、アスペクト比（縦横比）、前後範囲 zNear-zFar を設定する。

a（自動）にチェックを入れておくと、メイン画面下の接近・後退で自動的に zNear-zFar

を調整するため、近づき過ぎて表示から消えることはない。

このダイアログは開いた時点の視点を表示したり、ダイアログで数値を変更するだけではなく、ダイアログを開いたままにしておくと、別の視点設定方法によるＣＡＭ情報の変化を、逐次表示する機能も有している。

図４－２０：視点座標編集画面

リソース：IDD_DLG_SHITEN ハンドラ：CShiten (editshit.cpp)

ヘルプ：editshit.txt

（１６）可視範囲解析

選択したオブジェクトが見える割合を、設定したエリアについて解析し、可視率を色分けで表示する。処理アルゴリスムは、設定したエリアにおいて、解析の精度でメッシュ点を設定し、それぞれの点に関して、地面が存在すれば地面の高さを設定したうえで、各点からオブジェクトだけを表示た場合の表示画面に占めるオブジェクトの面積（ドット数）に対する、全地物を表示した場合のオブジェクトの画面上の面積（ドット数）の割合を計算する。

計算結果は、色分けの画像として表現する。ＬＳＳ－Ｓ編集モードにおいて解析を実行すると、解析結果を画像として保存し、メイン画面を平面図とした場合にも表示する。さらに、このシーンをシャッターで記録した上でファイル保存すると、画像名称と表示範囲を EFFECT コマンドとして保存する。その際に画像には、例えば「VISUAL003.sgi」といった名称を自動的に付し、kdb/image ディレクトリに保存する。これに対応する効果コマンドは、例えば

[例]: E3=EFFECT(VISUAL,62.18, 182, 54.81, 854.81, VISUAL003.sgi);

（効果コマンド名 VISUAL、画像の左下隅の座標と横縦サイズ、画像ファイル名）

という形式で自動的に生成し、LSS-S ファイルに保存する。

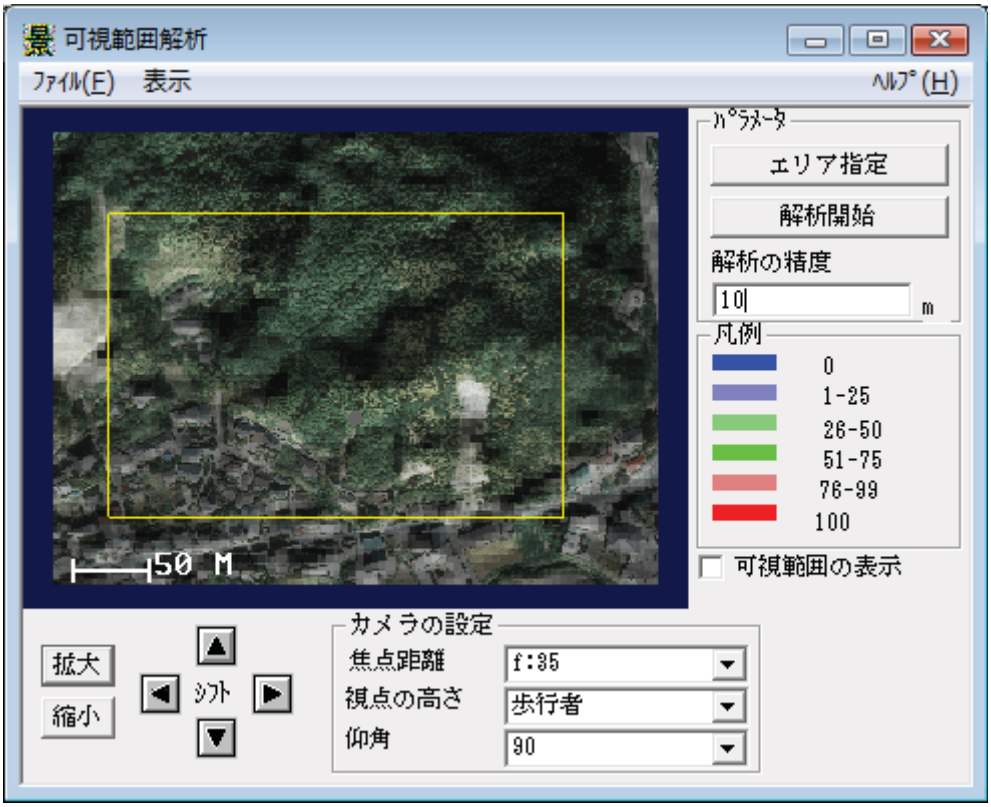


図４－２１：可視範囲解析画面

①　メニュー：IDR_MENU_KEIRO

メニュー構成：
［ファイル］
＋［終了］終了する
［表示］
＋［全体視界］存在するオブジェクト全てを表示する平面図表示範囲とする
＋［上下範囲］平面図表示する高さ範囲を設定する
＋［縦横範囲］平面表示する水平範囲を設定する
＋［表示モード］
＋＋［テクスチャ表示］平面図をテクスチャ表示する
＋＋［シェーディング表示］平面図を面で表示する
＋＋［ワイヤーフレーム表示］平面図を辺・稜で表示する
＋＋［オプション設定］
＋＋＋［地面のみ表示］地面の属性があるオブジェクトだけを表示する
＋＋＋［地面テクスチャ表示］地面の属性があるオブジェクトだけテクスチャ表示する
＋＋［オプション解除］オプションの設定を解除する

②　ＧＬ画面：COrthoVW m_orthoView (orthovw.cpp)

③　右側ダイアログ：リソース：IDD_DLG_PRM_KASHI

ハンドラ：CKashiWnd (kashiwnd.cpp)
メンバ変数：CDialogBar::m_Prm_Bar
④　下側ダイアログ：リソース：IDD_DLG_CTL
ハンドラ：CKashiWnd(　.cpp)
メンバ変数：CDialogBar::m_Ctl_Bar
⑤　ヘルプ：kashiwnd.txt

（１７）平面図表示画面における表示の上下範囲の設定

平面図表示（オルソ画面）を中心とするダイアログ（配置/コピー、平面生成など）のメニューから起動し、平面図表示における上下の範囲を設定する。建築物のデータ等の場合には、ある階の床と天井の間の高さを上限とすると、その階の平面図を表示することができる。地面よりも高い高さを下限とすると、地面の表示を消すことができる。

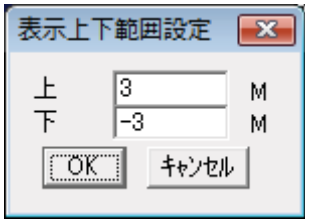


図４－２２：表示上下範囲設定画面

リソース：IDD_DIALOG_KELIHATAN ハンドラ：CKelihatan (kelihatan.cpp)

（１８）平面図表示画面におけるシフトと拡大・縮小の移動レートの設定

平面図表示（オルソ表示）を用いた編集ダイアログにおいて、表示範囲の選択を詳細に行いたい場合は、ステップレートを小さく設定する。また、広域的に探索する場合にはステップレートを大きく設定する。シフトの値は画面サイズに対する比率であるため、１を超えた値に設定すると、離れた領域をサンプルすることとなり、中間が抜けるため、見落とす場合がありうる。

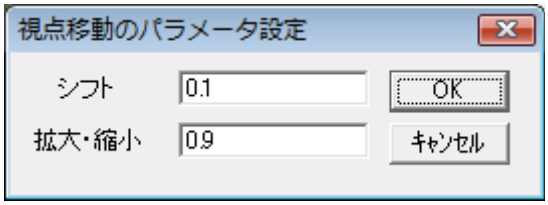


図４－２３：視点移動パラメータ設定画面

リソース：IDD_VIEWPARAM ハンドラ：CVwParam (vwparam.cpp)

（１９）視点設定

平面図上で位置をクリック指定することにより、その地点から眺めた景観をメイン画面で表示する。注視点はオブジェクトの選択または画面クリックで設定する。

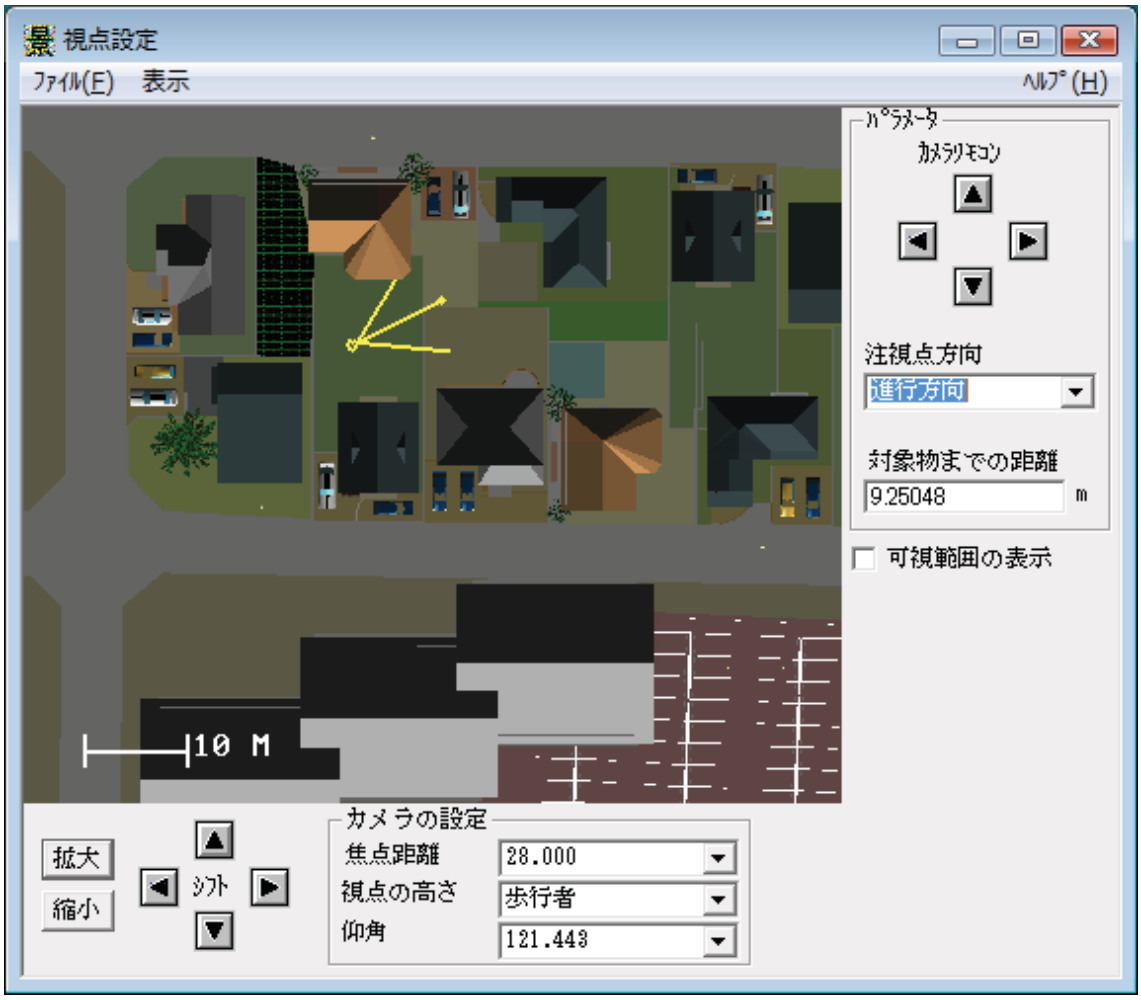


図４－２４：視点設定画面

① メニュー：IDR_MENU_KEIRO　（１６．①と同じ）

② ＧＬ画面：COrthoVW m_orthoView (orthovw.cpp)

③ 右側ダイアログ：リソース：IDD_DLG_PRM_SHITEN

ハンドラ：CShitenWnd (shitenwn.cpp)

メンバ変数：CDialogBar::m_Prm_Bar

③ 下側ダイアログ：リソース：IDD_DLG_CTL

ハンドラ：CShitenWnd(shitenwn.cpp)

メンバ変数：CDialogBar::m_Ctl_Bar

⑤ ヘルプ：shitenwn.txt

（２０）移動経路設定画面

画面上で移動経路（軌道）を設定し、メイン画面に移動途中の風景の変化をアニメーションで表示する。移動経路は、点列で指定するが、スプライン曲線で滑らかにした上で、一定の間隔で移動することにより、スムーズなカメラワークとすることができる。設定し

た経路は、LSS-G 形式の線としてファイル保存し、後日再利用することができる。

なお、アニメーションとして保存される AVI 形式のファイルは、通常の場合、三次元データよりもはるかに大きくなる。

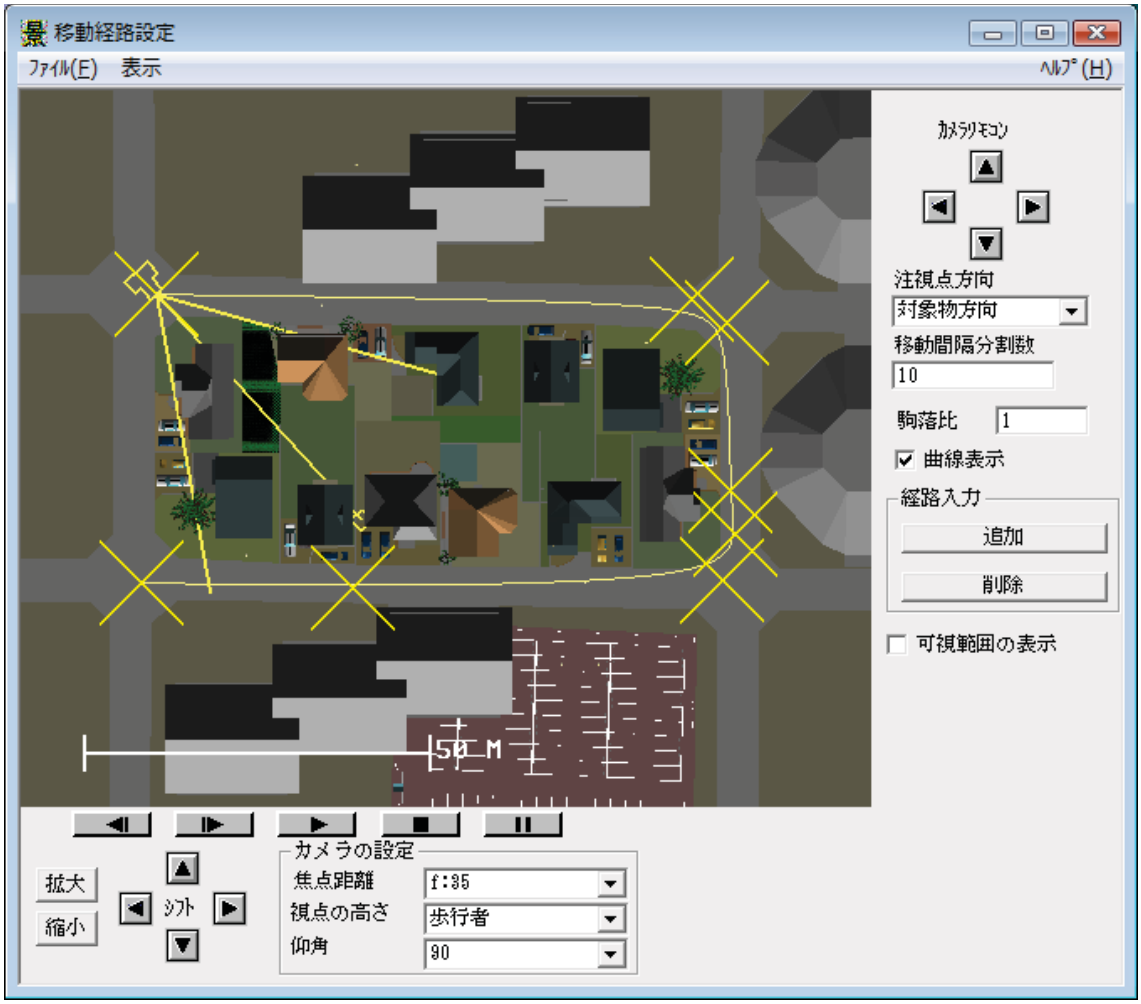


図４－２５：移動経路設定画面

① メニュー：IDM_MENU_KEIRO

メニュー構成：
［ファイル］
＋［経路読込］ファイル名を選択して、保存してある経路を読み込む
＋［経路保存］ファイル名を指定して、経路を保存する
＋［上書保存］読み込んだ名前、または以前保存した名前で保存する
＋［動画保存］上演と同時にアニメーションを avi ファイルとして保存する
＋［終了］終了する
［表示］
＋［全体視界］存在するオブジェクト全てを表示する平面図表示範囲とする
＋［上下範囲］平面図表示する高さ範囲を設定する

＋［縦横範囲］平面表示する水平範囲を設定する
＋［表示モード］
＋＋［テクスチャ表示］平面図をテクスチャ表示する
＋＋［シェーディング表示］平面図を面で表示する
＋＋［ワイヤーフレーム表示］平面図を辺・稜で表示する
＋＋［オプション設定］
＋＋＋［地面のみ表示］地面の属性があるオブジェクトだけを表示する
＋＋＋［地面テクスチャ表示］地面の属性があるオブジェクトだけテクスチャ表示する
＋＋［オプション解除］オプションの設定を解除する

② OpenGL 画面：COrthoVW m_orthoView(orthovw.cpp)
③ 右側ダイアログ：
リソース：IDD_DLG_PRM_IDO　ハンドラ：CKeiroWnd (keirownd.cpp)
CDialogBar:: m_Prm_Bar
④ 下側中段ダイアログ：
IDD_DLG_PLAY　ハンドラ：CKeiroWnd(keirownd.cpp)
⑤ 下側下段ダイアログ
IDD_DLG_CTL　ハンドラ：CKeiroWnd (keirownd.cpp)
CdialogBar::m_Ctl_Bar
⑥ ヘルプ：keirownd.txt

（２１）マテリアル、カラー、テクスチャの編集（古典的スタイル）

メイン画面で選択したオブジェクトに関して、カラーやマテリアルの編集を行う。

カラー編集は、三原色とアルファ値をスライドバーで指定する他、左側の数値表示欄の数値をキーボード入力し、フォーカスを別の数値表示欄などに移す方法でも指定することができ、スライドバーの位置も自動的に追随する。

上方の見出し用の OpenGL 画面に球面（左）と平面（右）があり、設定したカラーを表示する。

このダイアログを開いたまま、メイン画面で次々と選択対象を変更し、能率的に作業することができる。メイン画面のメニュー[編集][選択モード]のサブメニューから、選択方法を、グループ、面、同色面のいずれかに設定する。初期値はグループとしており、その場合、選択したグループの全ての面に、同じカラーが設定される。

特殊な機能として、[地面の属性]のチェックボックスにより、選択したオブジェクトに対して地面の属性を設定／解除することができる。

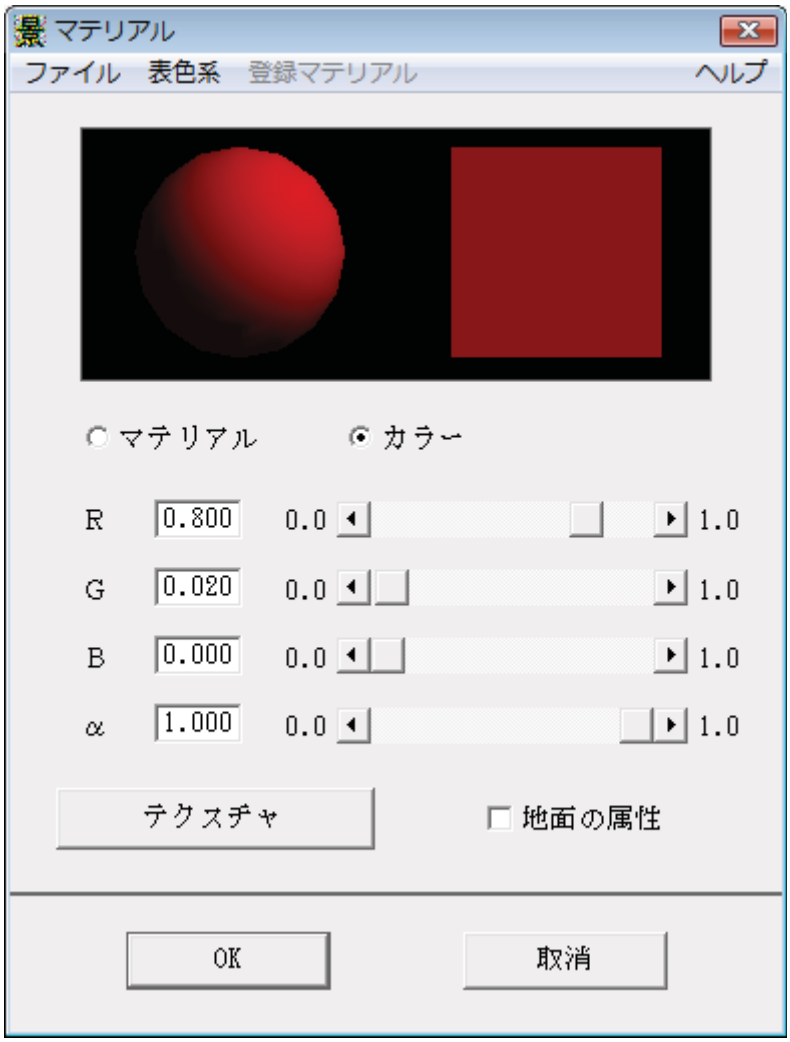


図４－２６：カラー・マテリアル編集画面

① メニュー：IDR_MENU_MATERIAL

メニュー構成
［ファイル］
＋［閉じる］ダイアログを終了する
＋［カラーのファイル保存］カラーを LSS-G 形式のファイルに保存する
＋［カラーの上書保存］読み込んだファイル名、以前保存したファイル名にて保存する
＋［カラーのファイル読込］保存してあるカラーを読み込む
［表色系］３種類の表色系で相互変換するダイアログ（２８）を開く
［登録マテリアル］
＋［マテリアル選択画面］グラフィックなマテリアルの選択画面(25)を開く
＋［マテリアルファイル選択］マテリアルファイル選択画面(24)を開く
（この画面でマテリアルファイルが選択された場合、同じダイアログを用いて
マテリアル選択に進む）

＋［選択済マテリアルファイル］直ちにファイル中のマテリアルを選択する(24)
＋［指定マテリアル内容表示］選択対象物に設定されているマテリアルを表示

② OpenGL 画面：CMate2Dlg m_dlg (mate2.cpp)
③ ダイアログ　リソース:IDD_DLG_MATERIAL　ハンドラ:CEditMate (editmate.cpp)
④　ヘルプ：edimate.txt

（２２）テクスチャ編集（クラシック）

テクスチャをファイル名で選択し、選択したオブジェクトに貼り付ける。

図４－２７：テクスチャ編集画面

① メニュー：IDR_MENU_TEXTURE
② ダイアログ　リソース：IDD_DLG_TEXTURE ハンドラ：CEditText (edittext.cpp)
③　ヘルプ：edittext.txt

（２３）テクスチャ・リスト（古典的スタイル）

・概要：テクスチャ編集ダイアログのテクスチャ選択ボタンで起動する。SGI 形式のテクスチャの一覧を表示する。何も選択せずに OK ボタンを押すと、一般的なファイル選択ダイアログを開き、JPG など、様々な形式の画像をテクスチャとして選択する。

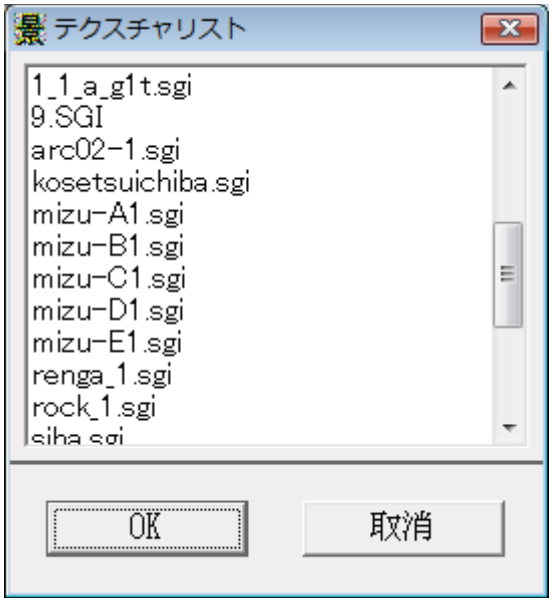


図４－２８：テクスチャファイル選択画面

リソース：IDD_DLG_TEXLIST ハンドラ：CEditTexl (edittexl.cpp)

（２４）マテリアル選択（古典的スタイル）

マテリアル編集画面で、ラジオボタンをマテリアルに変更し、メニューのマテリアルを選択すると開く。同じダイアログを二段構成で使用し、第一段階では、マテリアルファイルの選択を、また第二段階ではマテリアルの選択を行う。

第一段階：

選択して OK ボタンを押すと、そのマテリアルファイルの中で定義されたマテリアルの一覧を表示して第 2 段階に進む。また、何も選択されない状態で OK ボタンが押された場合には、現在編集中の地物のいずれかの部分に適用されているマテリアルの一覧を表示して第２段階に進む。

第二段階：

一覧表示されているマテリアルからユーザーが選択した時点で、マテリアル設定画面(21)の上方の色玉に、そのマテリアルを反映させる。

OK ボタンが押された時点で、メイン画面で選択されている編集対象物にマテリアルを設定する。この時、画面で何も選択されておらず、選択したマテリアルを適用する対象がない場合には、選択したマテリアルファイルの内容をメモ帳で表示する。

また、一覧表示されているマテリアルからユーザーが何も選択していない状態で、OK ボタンが押された場合には、設定されているマテリアルを解除する。このとき、メイン画面で何も選択されていない場合には、確認の上でマテリアル解消のための特殊な処理を実行する。この処理においては、全ての地物に適用されているマテリアルを解除し、近い色のカラーに置き換える。この時、マテリアルに含まれていたカラー以外の、鏡面反射率や輝

度などに関する情報は失われる。

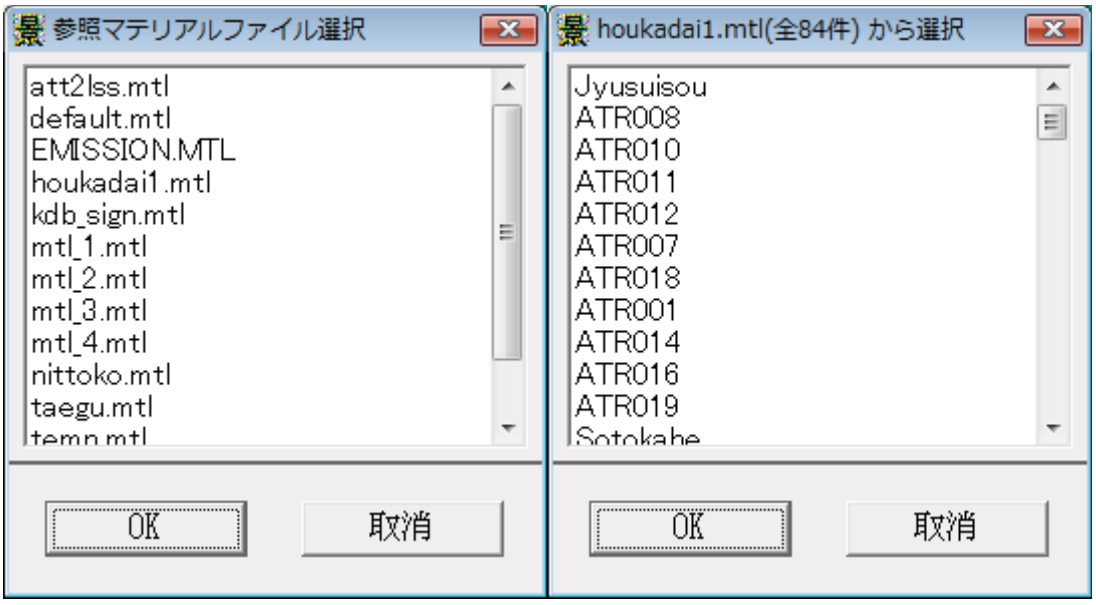


図４－２９：マテリアル選択画面（古典的）

リソース：IDD_DLG_MAT_LIST ハンドラ：CMatList (matlist.cpp)

（２５）グラフィックなマテリアルの選択

選択しているマテリアル・ファイルの中で定義されたマテリアルを、参照表示しながら選択できる編集画面である。

中央にマテリアル名称のリストを表示する。左側に縦に並んだ７の小さなカラーボックスは、各々OpenGL の画面である。右側のスクロールバーは一つの OpenGL 画面である。右側の OpenGL 画面は、マテリアル・ファイルに含まれる全てのマテリアルを総覧しており、スクロールバーのつまみは、マテリアルが７を超える場合に表示され、その内７個分の縦幅となっている。スクロールバーに対応する位置にある 7 種類のマテリアルが、左側の７個のカラーボックスに表示される。

上にある色球の OpenGL 画面は左右二つあり、設計検討の途上で、右側に一時保存し、比較しながら決定することができる。

現在一覧表示しているマテリアル・ファイル名称を、タイトルに補足表示している。

図４－３０：グラフィックなマテリアル選択画面

① メニュー：IDR_MENU_MATERIAL2

［ファイル］
＋［閉じる］ダイアログを終了する。
［色見本］所定のディレクトリにある全てマテリアルファイル名をサブメニューに表示

② 上左の OpenGL 画面 CSphereDlg m_KyuDlg (spheredlg.cpp)
③ 上右の OpenGL 画面 CSphereDlg m_saveKyuDlg (spheredlg.cpp)
④ 左側の OpenGL 画面 CMatFrm m_colFrm[7] (matfrm.cpp)
⑤ 右側の OpenGL 画面 CColorSashDlg m_obiDlg (colsashdlg.cpp)
⑥ ダイアログ
　リソース：IDD_DIALOG_MATERIAL　ハンドラ：CEditMaterial (editmat2.cpp)
⑦ ヘルプ：editmate2.txt

（２６）グラフィックなテクスチャ編集

ディレクトリにあるテクスチャを画像で一覧表示する。

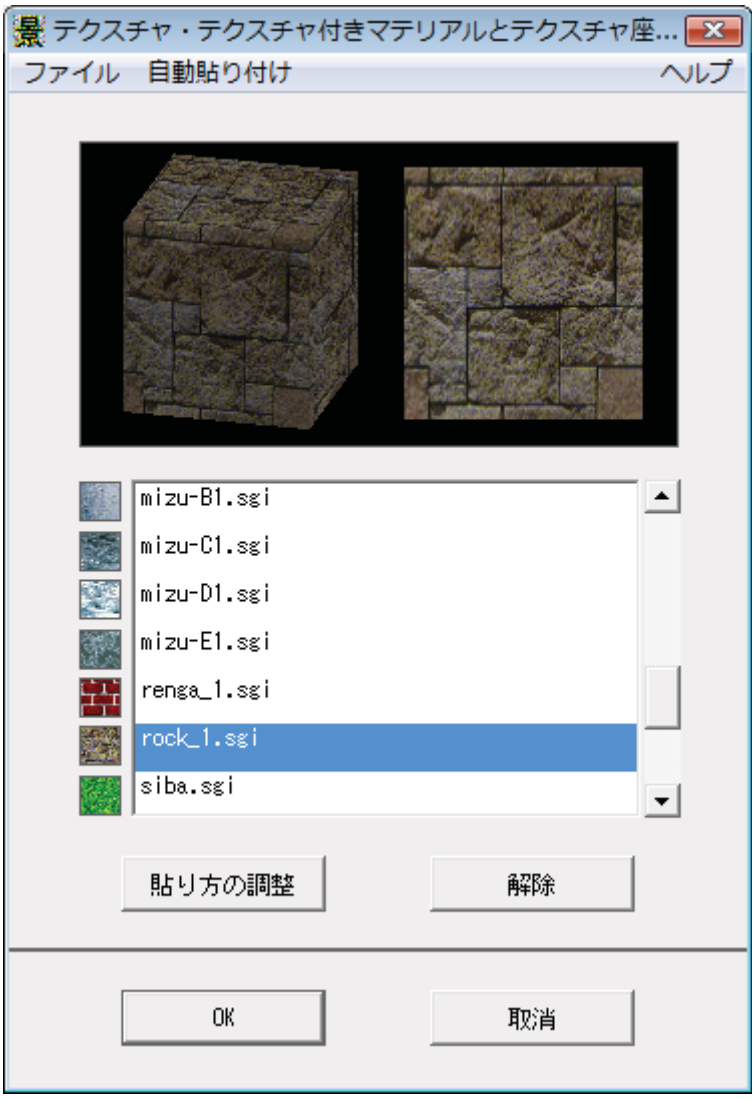


図４－３１：グラフィックなテクスチャ選択画面

① メニュー：IDR_MENU_TEXTURE2

［ファイル］ ＋［閉じる］ダイアログを終了する。 ［自動貼り付け］autotex.set ファイルによる定義内容をサブメニューとして構成する （autotex.set は、選択されている言語のディレクトリにあるものを使用する）

② 上の OpenGL 画面 CCubeDlg m_boxDlg (cubedlg.cpp)

③ 左の 7 個の OpenGL 画面 CTexFrm m_texFrm[7] (texfrm.cpp)

④ ダイアログ

リソース：IDD_DIALOG_TEXTURE ハンドラ：CEditTexture(edittex2.cpp)

⑤ ヘルプ：edittex2.txt

（２７）テクスチャの貼り方の調整

多くの面から成る近似的曲面に対して、平行投影で、連続的に見えるようにテクスチャを貼るために、選択したグループの全ての面テクスチャ座標（２－９参照）を一括設定する。このための個々の面のテクスチャ座標を計算のために必要となる、集合的な座標軸とスケールを設定する。u 軸と v 軸をベクトルとして指定すると、uv 原点として指定したポイントから、u 軸 v 軸に直角な向きに、u スケール・v スケールで倍率を掛けたしたテクスチャを投影され、この時選択したグループを構成する各面の頂点の uv 座標を計算し、これをテクスチャ座標として各頂点に設定する。

図４－３２：テクスチャの貼り方の調整画面

リソース：IDD_DIALOG_TEXTURE_MAP ハンドラ：CTextureMap(textmap.cpp)

（２８）様々な表色方式によるカラー編集

選択した表色方式のスライドバーを移動すると、他の表色方式による換算値に対応して、スライドバーを自動的に動かす。

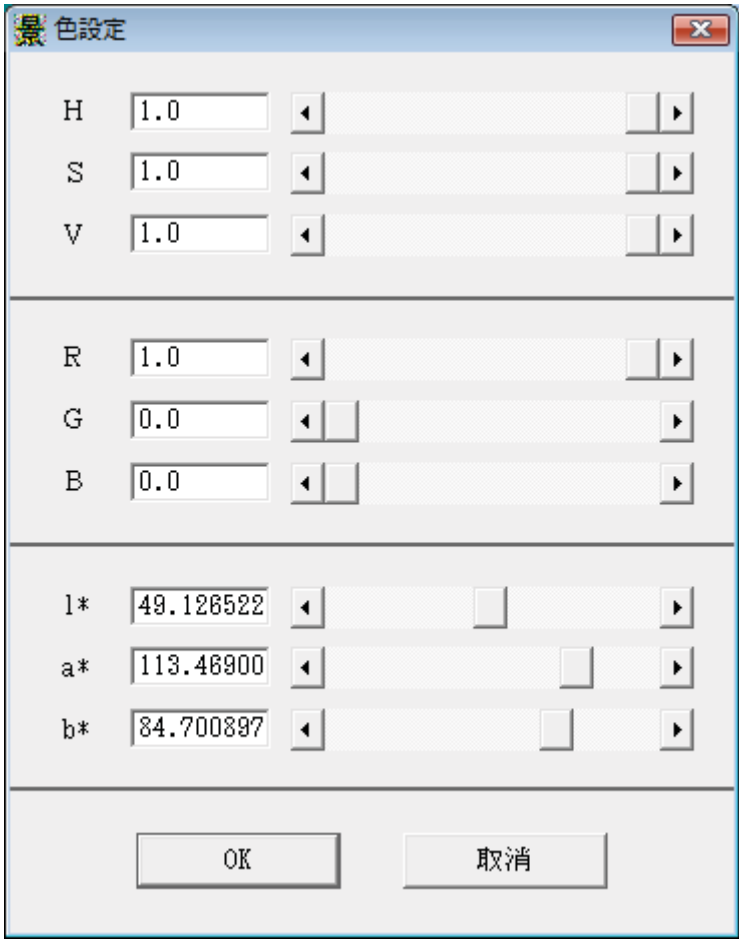


図４－３３：様々な表色系による色設定画面

リソース：IDD_DIALOG_COLOR_SET ハンドラ：CColorSet (colorset.cpp)

（２９）光源の自動設定

月日・時分と、検討地域の緯度経度から、太陽方位を自動的に計算する。同時に、選択した天候に応じて、副光源を設定する。更に、効果（EFFECT）を用いて、霧も設定することができる。

詳細設定ボタンで、クラシックな光源設定ダイアログを開く（一つづつ色と位置を設定する）。

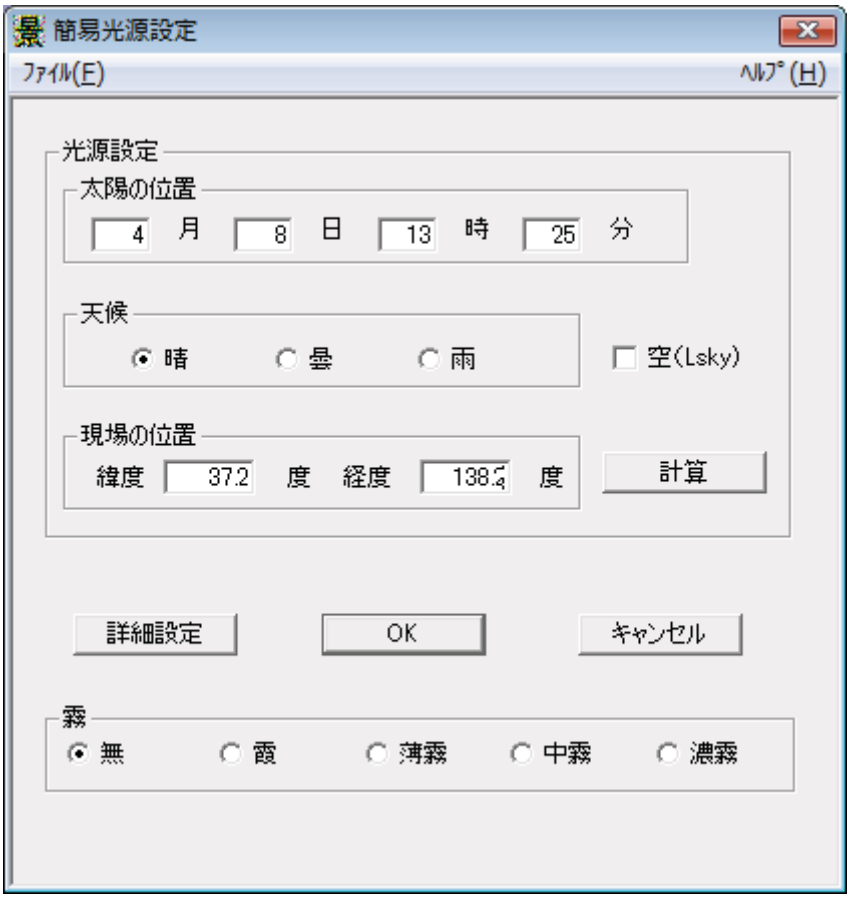


図４－３４：簡易光源設定画面

① メニュー：IDR_MENU_GY

［光源読込］LSS-S 形式のサブセットを用いたファイル（拡張子.s）を読み込む ［光源保存］光源をファイルに名前を付けて保存する ［光源上書］読み込んだファイル名または以前保存したファイル名で保存する ［終了］ダイアログを閉じる

② リソース：IDD_DLG_NEW_LIGHT ハンドラ：Cgydlg (gydlg.cpp)

③ ヘルプ：gydlg.txt

（３０）光源グループ設定

光源グループを構成する光源を編集する。シーン・ファイルを編集する中で、同じ光源条件を複数のシーンに対して共通で使用するような編集を行うことができる。

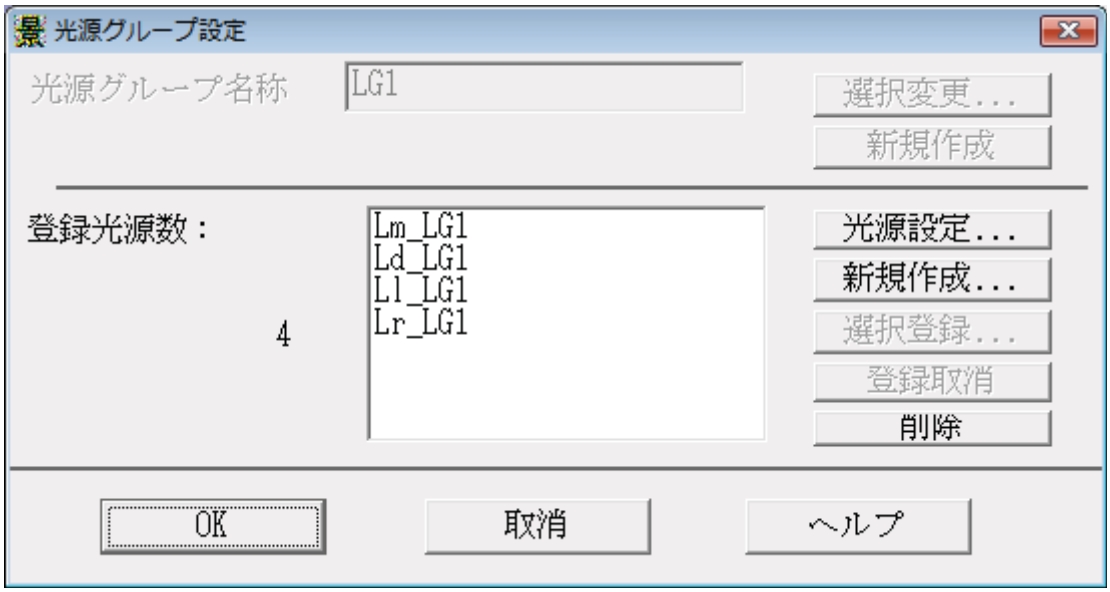


図４－３５：光源グループ設定画面

①リソース：IDD_DLG_LIGHT　ハンドラ：CEditLigh (editligh.cpp)

②ヘルプ：editligh.txt

（３１）古典的な光源編集

光源グループ編集ダイアログで選択した一つの光源のカラーと位置を編集する。OpenGLの表示画面で、球と正方形に試し適用する。

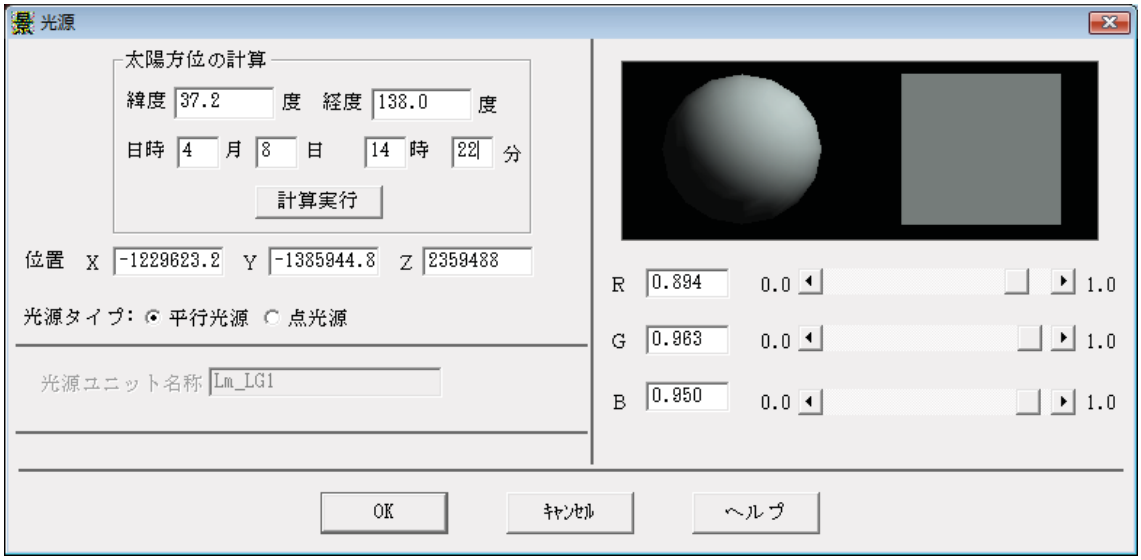


図４－３６：光源ユニット設定画面

① OpenGL 画面：CeditLight2Dlg　m_dlg (editlig2.cpp)

② ダイアログ　リソース：IDD_DLG_KOUGEN　ハンドラ：Ckougen (kougen.cpp)

③ ヘルプ：kougen.txt

（３２）経年変化

システムの時間を変更する。時間依存するマテリアルを貼り替えると共に、時間依存するモデルも更新する。

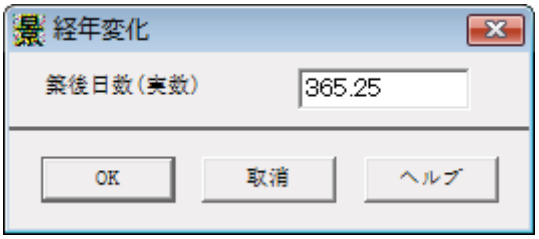


図４－３７：経年変化設定画面

①ダイアログ

リソース：IDD_DLG_KEINEN_HENKA　ハンドラ：CDispKeinen (dispkein.cpp)

②ヘルプ：dispkein.txt

（３３）グリッド

オルソ系画面（平面図、立面図）表示モードにおいて、指定した格子間隔のグリッドを表示する。このほかに、縮尺、可視範囲解析結果の表示の ON-OFF をこのダイアログで指定する。

図４－３８：グリッド設定画面

①ダイアログ　　リソース：IDD_ DLG_GRID ハンドラ：CGrid (grid.cpp)

②ヘルプ　　grid.txt

（３４）アンチエリアシング(Anti Aliasing)

鋸状の境界線の表示を解消する。このために、表示画面よりも細かい解像度の画面に対する表示処理を行い、表示画面の解像度に下げる際に、集約するメッシュ格子点の色彩を平均することにより、中間色を挿入し、ギザギザ感を減じる。

OK ボタンを待たずに、設定が変更された時点で、直ちに表示に反映する。

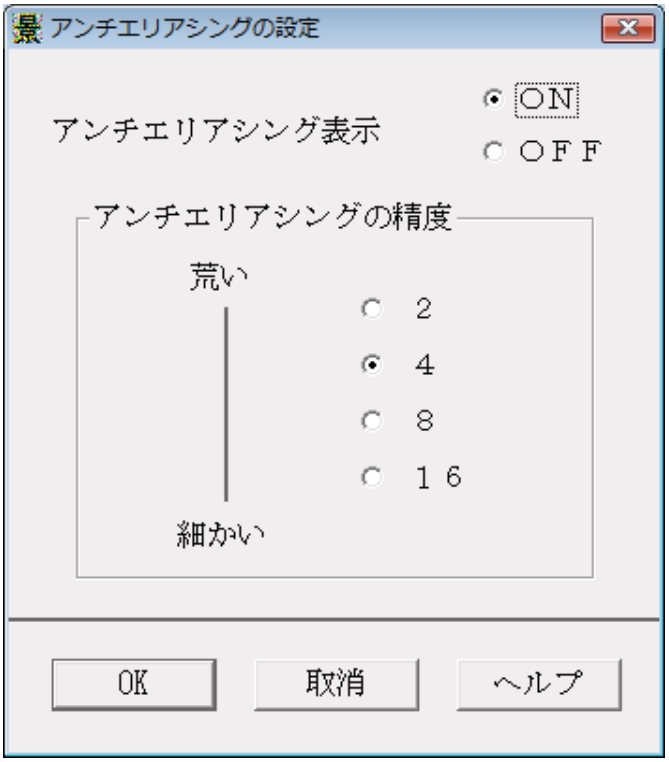


図４－３９：アンチエリアシング設定画面

①ダイアログ　リソース：IDD_DLG_ANTI_AREA ハンドラ：CantiAlias (antialia.cpp)

②ヘルプ　antiarea.txt

（３５）影のパラメータ設定

影のタイプ、副次的光源を影の部分に適用するか否かを設定するチェック・ボックスを有する。

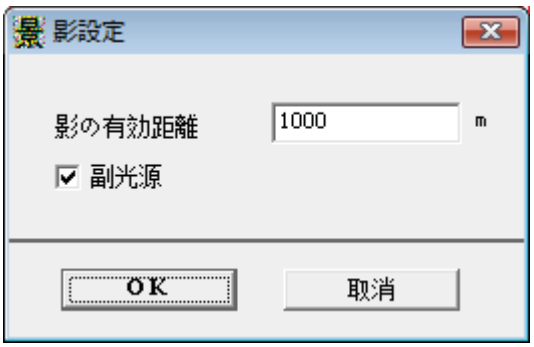


図４－４０：影設定画面

リソース：IDD_DLG_SHADOW　ハンドラ：CShadowDlg(shadowdlg.cpp)

（３６）ステレオ表示設定

ステレオ・モード、影の長さ、ステレオ・スワップを指定する。

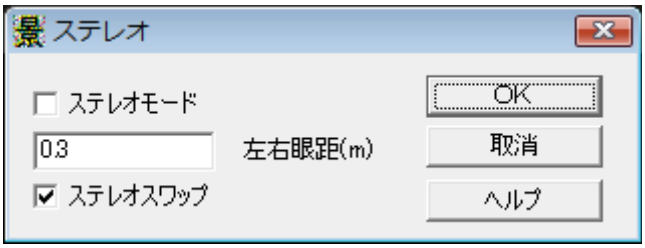


図４－４１：ステレオ表示設定画面

リソース： IDD_DIALOG_STEREO　ハンドラ：Cstereo (stereo.cpp)

（３７）平面の編集

メイン画面でセレクト行為が無い場合、頂点列から新しい平面を生成する。

メイン画面でセレクトが行われていた場合には、既存平面の再編集を行う。

このダイアログが開いてからメイン画面で面が選択された場合には、既存平面に対する穴あけ加工を行う。

その他、面を定義するのと同様な手順で指定した頂点列を用いて道路や河川等の掃引体のための断面を定義したり、病的な面を治療するなど、様々な機能がある。

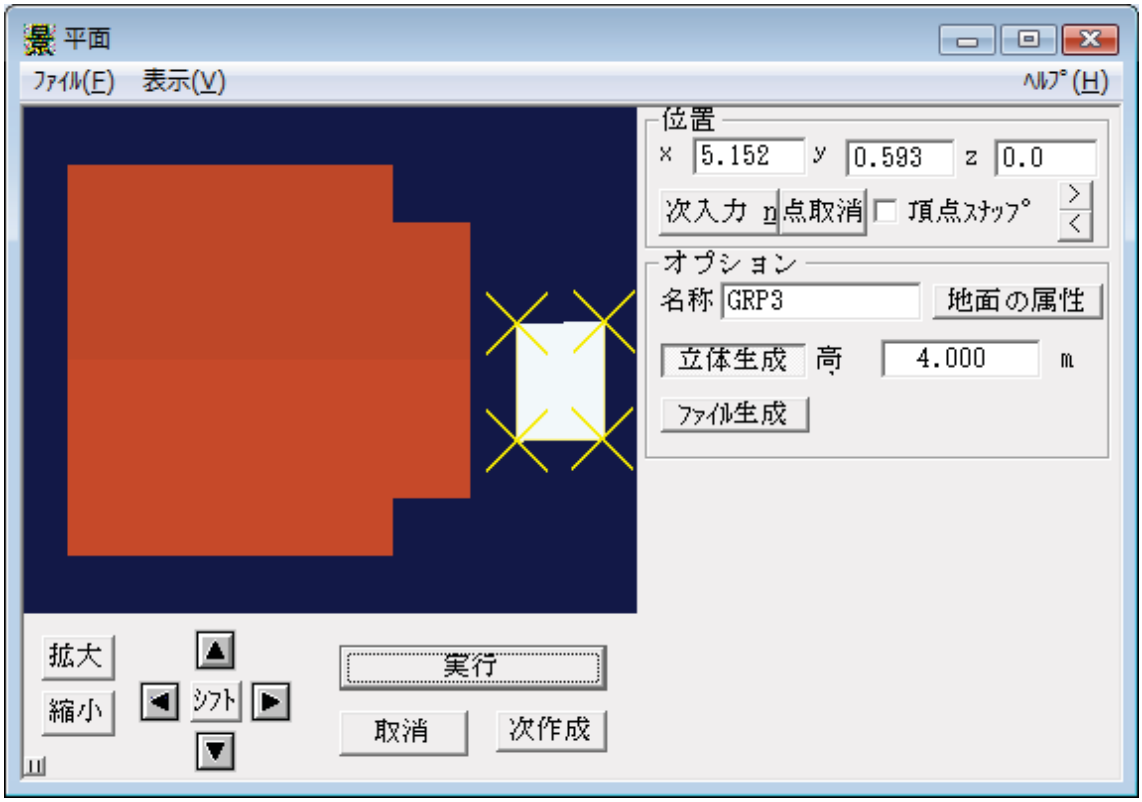


図４－４２：平面入力画面

① メニュー：IDR_MENU_PLANE

メニュー構成：
［ファイル］
＋［選択された面の編集］
＋＋［面の削除］選択した面を削除する
＋＋［面を別グループにする］新たなグループを作成し、選択した面を移管する
＋＋［面をファイル保存する］ファイル名を指定して、選択した面を保存する
＋＋［面を裏返す］面を構成する頂点列を逆回りに変換する
＋＋［面の情報を見る］頂点数、法線、カラー等の情報を表示する
＋＋［面の病気を治療する］病的な面を正常な面に変換する
＋［点列を線として保存］
＋＋［道路断面として保存］図形を XZ 平面上の線として保存、道路断面リストに追加
＋＋［河川断面として保存］図形を XZ 平面上の線として保存、河川断面リストに追加
＋＋［線として保存］頂点列を線として保存する
＋［終了］終了する
［表示］
＋［グリッド］1 m 固定のグリッドの表示の ON／OFF
＋［メジャー］スケール自動設定による縮尺の表示の ON／OFF
＋［全体視界］平面図表示範囲を調整し、存在するオブジェクト全てを表示する
＋［上下範囲］平面図表示する高さ範囲を設定する
＋［縦横範囲］平面図表示する水平範囲の移動の刻みを設定する
＋［表示モード］
＋＋［テクスチャ表示］平面図をテクスチャ表示する
＋＋［シェーディング表示］平面図を面で表示する
＋＋［ワイヤーフレーム表示］平面図を辺・稜で表示する
＋＋［オプション設定］
＋＋＋［地面のみ表示］地面の属性があるオブジェクトだけを表示する
＋＋＋［地面テクスチャ表示］地面の属性があるオブジェクトだけテクスチャ表示する
＋＋［オプション解除］オプションの設定を解除する

② OpenGL 画面：COrthoVW　m_orthoView (orthovw.cpp)

③ 右側ダイアログ
リソース：IDD_DLG_PLANE_PRM　ハンドラ：CPlaneWnd (planewnd.cpp)
メンバ変数：CDialogBar　m_Prm_Bar

④ 下ダイアログ
リソース：IDD_DLG_CTL　ハンドラ：CPlaneWnd (planewnd.cpp)
メンバ変数：CDialogBar m_Ctl_Bar

（３８）クラシックな線の編集

以下のような操作を行う。

① 頂点数を指定した上で、各頂点の座標値を入力し、新たな線を生成する。
このとき、線分名称、適用するマテリアルを指定することができる。

② 探索ボタン：線の選択ダイアログを開き、編集対象とする既存の線を選択する

③ 接続ボタン：線の選択ダイアログで対象とする線分を選択した上で、この線分と接続している線分を検索し、存在する場合には一つの折れ線に統合する。

④ 軌跡保存：現在選択している線を、道路生成・河川生成で使用する中心線軌跡データとして保存する。

⑤ 軌跡読み込み：中心線軌跡データを読み込み、線として編集する

⑥ >X>Y>Z：線の XYZ 座標を、巡回的に交換する

⑦ 逆順：線の頂点列を逆準に変換する

⑧ 一括変換：ダイアログ(62)を開き、平行移動、倍率の一括変換を行う

図４－４３：線の編集画面

ダイアログ：IDD_DLG_PRIM_LINE　ハンドラ：CPrimLineDlg (primline.cpp)

ヘルプ：primline.txt

（３９）線分の一括変換

選択した線に対する一括処理内容とパラメータを設定する。

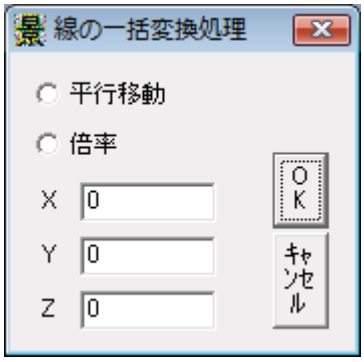


図４－４４：線分の一括変換画面

リソース：IDD_LINE_PROCESS　ハンドラ：CLineProcess (lineproces.cpp)

実際のデータ変換は、ダイアログ呼び出し元の CPrimLineDlg の中で実行している。

（４０）橋

開発初期の原始的機能。選択した種類の橋（固定的外部ファイルデータ）を読み込んで、シーンの中に表示する。現在では配置コマンドによりデータベースから検索したり、直接ファイルとして読み込むことが可能。博物館的な意味合いで残している。

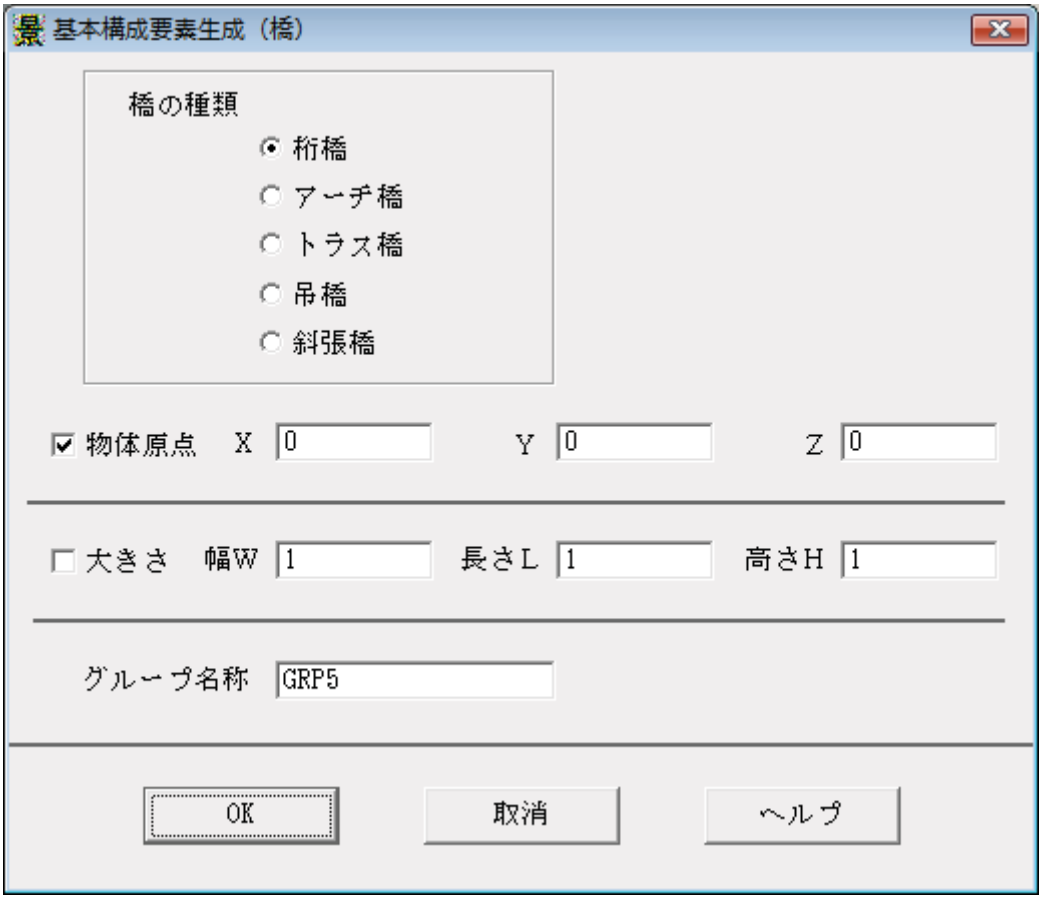


図４－４５：橋の画面

リソース：IDD_DLG_ELEM_BRIDGE ハンドラ：CElemBridgeDlg(elembrid.cpp)
ヘルプ：elembrid.txt

（４１）基本構成要素（草）

初期の機能である。その後、配置機能のエリア配置で代替可能。博物館的な意味しかない。

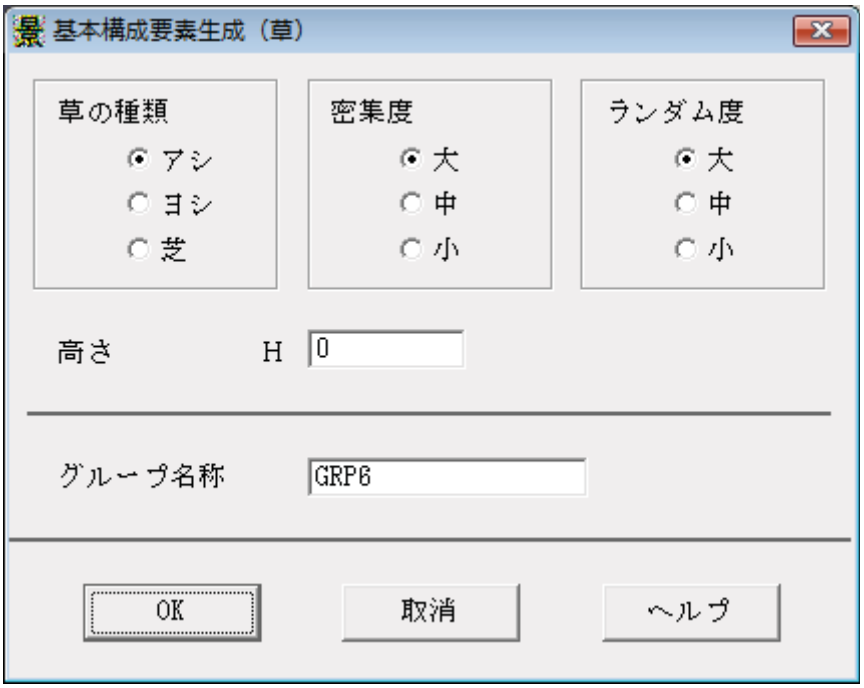


図４－４６：草の画面

リソース：IDD_DLG_ELEM_GRASS ハンドラ：CElemGrassDlg(elemgras.cpp)
ヘルプ：elemgras.txt

（４２）基本構成要素：道路

選択した断面を、平面図表示の OpenGL 画面でユーザーが指定した中心線軌跡に従って掃引して、立体形状を生成する。

中心線の軌跡は、LSS-G 形式としてファイル保存・読み込みができる。スプライン曲線により、入力された折れ線を滑らかに補間する機能がある。

オプションとして地面を識別し、地面から相対的な高さとして、軌跡上の点のｚ座標を指定することができる。

形状生成のロジックは、掃引体１面と同様である。既存の地面や地形に対する加工機能が無いため、生成した道路が地面よりも低い場合には、下に隠れてしまう。

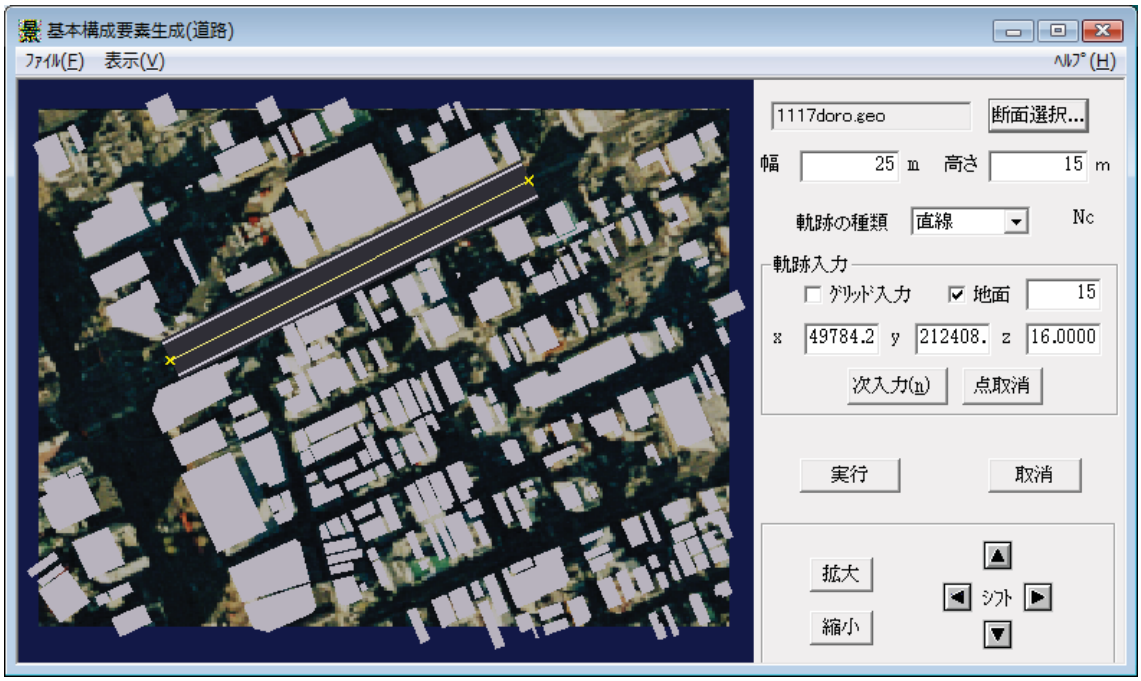


図４－４７：道路生成画面

①┌　　メニュー：IDR_MENU_HAICHI

メニュー構成：
［ファイル］
＋［経路読込］ファイル名を選択して、保存してある経路を読み込む
＋［経路保存］ファイル名を指定して、経路を保存する
＋［上書保存］読み込んだ名前、または以前保存した名前で保存する
＋［終了］終了する
［表示］
＋［グリッド］１m 固定のグリッドの表示の ON／OFF
＋［メジャー］スケール自動設定による縮尺の表示の ON／OFF
＋［全体視界］平面図表示範囲を調整し、存在するオブジェクト全てを表示する
＋［上下範囲］平面図表示する高さ範囲を設定する
＋［縦横範囲］平面図表示する水平範囲を設定する
＋［表示モード］
＋＋［テクスチャ表示］平面図をテクスチャ表示する
＋＋［シェーディング表示］平面図を面で表示する
＋＋［ワイヤーフレーム表示］平面図を辺・稜で表示する
＋＋［オプション設定］
＋＋＋［地面のみ表示］地面の属性があるオブジェクトだけを表示する
＋＋＋［地面テクスチャ表示］地面の属性があるオブジェクトだけテクスチャ表示する
＋＋［オプション解除］オプションの設定を解除する

② OpenGL 画面：COrthoVW m_orthoView (orthovw.cpp)
③ 右側ダイアログ
リソース：IDD_DLG_ROAD　ハンドラ：CRoadWnd (roadwnd.cpp)
メンバ変数：CDialogBar::m_Prm_Bar
④ ヘルプファイル roadwnd.txt

その他：リソース：IDD_DLG_ELEM_ROAD と関連ハンドラ
C:ElemRoadwayDlg(elemroad.cpp)は Ver.2.09 では使用されていない。

（４３）断面ファイルリスト

コントロール･ファイル　ROADSEC.SET に登録されている断面ファイル（LSS-G 形式）の一覧を表示し、選択・OK で、ファイルを開き、呼び出し元の道路生成ダイアログの断面データに登録を行う。

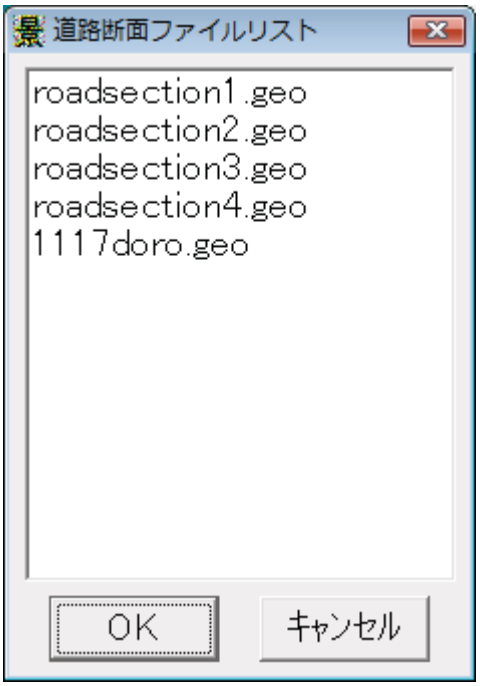


図４－４８：道路断面ファイル選択画面

リソース：IDD_DLG_DANMEN　ハンドラ：CRoadDanmen (roaddan.cpp)

（４４）基本構成要素：河川

前項と殆ど同じロジックにより、河川（堤防＋川原）を生成する。

図４－４９：河川生成画面

① メニュー：IDR_MENU_HAICHI

メニュー構成：
［ファイル］
＋［経路読込］ファイル名を選択して、保存してある経路を読み込む
＋［経路保存］ファイル名を指定して、経路を保存する
＋［上書保存］読み込んだ名前、または以前保存した名前で保存する
＋［終了］終了する
［表示］
＋［グリッド］１m 固定のグリッドの表示の ON／OFF
＋［メジャー］スケール自動設定による縮尺の表示の ON／OFF
＋［全体視界］平面図表示範囲を調整し、存在するオブジェクト全てを表示する
＋［上下範囲］平面図表示する高さ範囲を設定する
＋［縦横範囲］平面図表示する水平範囲を設定する
＋［表示モード］
＋＋［テクスチャ表示］平面図をテクスチャ表示する
＋＋［シェーディング表示］平面図を面で表示する
＋＋［ワイヤーフレーム表示］平面図を辺・稜で表示する
＋＋［オプション設定］
＋＋＋［地面のみ表示］地面の属性があるオブジェクトだけを表示する
＋＋＋［地面テクスチャ表示］地面の属性があるオブジェクトだけテクスチャ表示する
＋＋［オプション解除］オプションの設定を解除する

② OpenGL の編集画面：COrthoVW　m_orthoView (orthovw.cpp)

③ 右側のダイアログ

リソース：IDD_DLG_ROAD　ハンドラ：CRoadWnd (roadwnd.cpp)

メンバ変数：CDialogBar::m_Prm_Bar

ヘルプ：roadwnd.txt

（４５）河川断面の選択ダイアログ

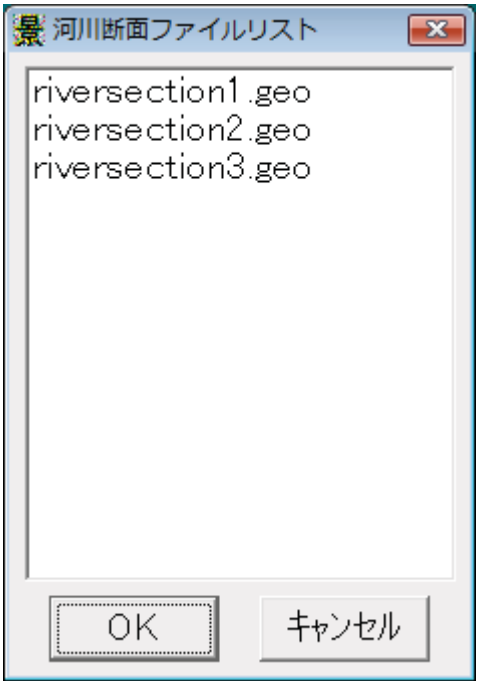


図４－５０：河川断面ファイル選択画面

リソース：IDD_DLG_ROAD　ハンドラ：CRoadDanmen (roaddan.cpp)

（４６）シャッター

シーン編集中に、ユーザーが操作すると、その時点の視点情報を記録する新たなシーンを生成し、末尾に追加する。

初期は、視点情報を記録するのみであったが、Ver.2.09 においては、機能を大幅に増補し、その他の全ての編集操作（モデルの変更、時刻の変更、背景・前景の変更）を反映させた新たなシーンを生成する仕様とした。また、既存シーンの修正も可能とするために、現在のシーンを更新するボタンを追加した。更に、ステレオ表示（７－４）、影の表示（７－５）、高速表示処理（７－６）など、表示方法が多様化したことに対応して、このようなオプショナルな表示環境に関する情報も、従来活用されていなかった EFFECT コマンドの形でシーンデータ(s3Scene 構造体及び LSS-S ファイル)に記録できるようにした。これらにより、

外部ファイル形式（３－２）を変更することなく、異なる実行環境におけるプレゼンテーションに際しての表示の再現性を高めている。

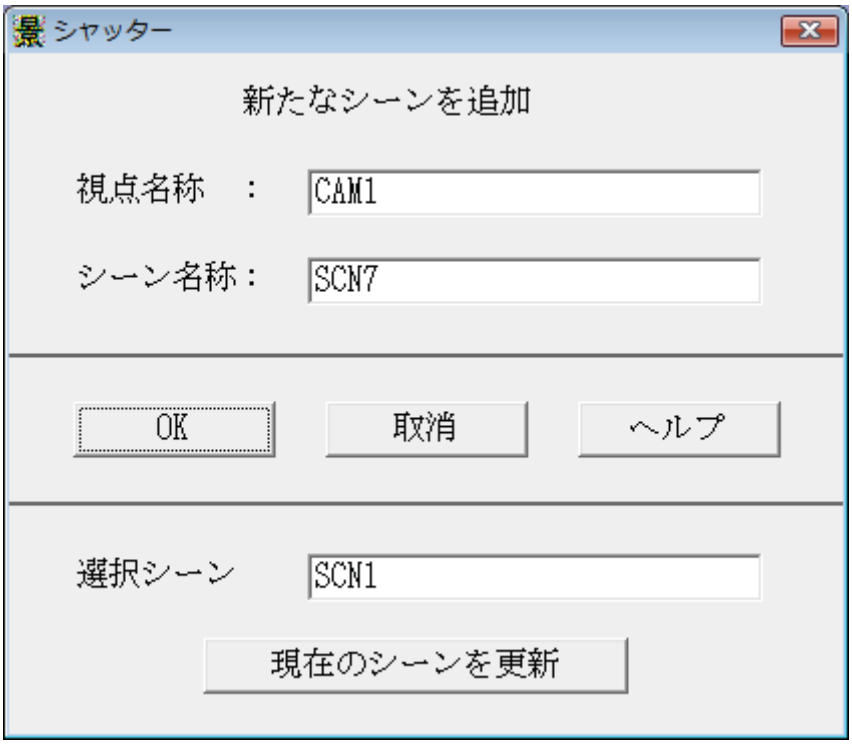


図４－５１：シャッター画面

リソース：IDD_DLG_SUTTER　ハンドラ：CShutterDlg(shutterd.cpp)

ヘルプ：shutterd.txt

（４７）LSS-G ファイルの最適化保存

地形データなど、規則的なパターンを繰り返すデータは、コンパクトに保存することが可能であり、コンバータの中で様々な工夫が可能であるが、そのようなファイルを読み込んだ上で加工を行うと、データの規則性は失われてしまう。しかし、加工されなかった周辺の大部分の形状は、元の地形の性質を保っている。そこで最適化保存により、可能な限りコンパクトな形でデータを保存する。

共通のカラー、テクスチャ、マテリアルに関しては、それぞれのグループ、面の出力に際して新たな記述を起こすのではなく、ファイル書き出しに先立って、保存しようとする全体に関してリストアップ、ソーティングを行い、共通の定義を冒頭で行った上で、個々のグループや面の書き出しでこれらを参照する形式とすることで、重複を避け、コンパクトに記述することができる。

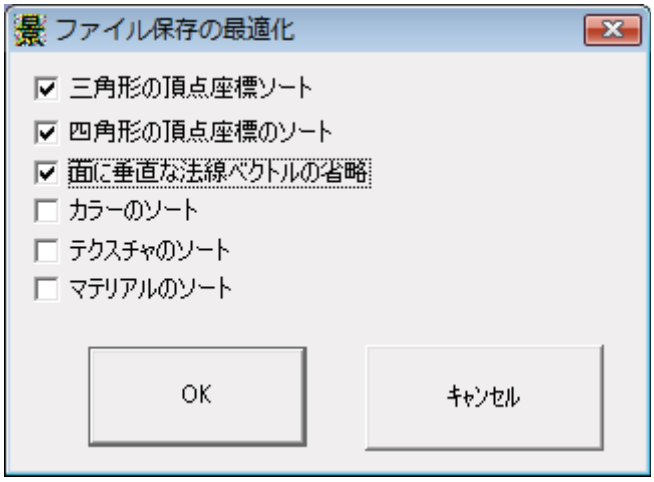


図４－５２：最適化保存設定画面

リソース：IDD_DLG_SAVE_OPTIM　ハンドラ：COptim (optim.cpp)

（４８）画像視点抽出

初期のバージョンから提供している、背景写真と三次元モデルの位置合わせを支援する機能である。対象区域に座標系を定義した上で、映っている対象物から座標のわかるポイントを指定した上で、それぞれのポイントを画面クリック選択し、対応する三次元座標を入力する。作成された画像ピクセル座標と実空間の三次元座標のセットに関して、誤差を最小とする視点位置を反復計算し、画像を撮影したカメラ情報（視点・注視点・焦点距離・傾きなど）を計算する。

計算の結果得られたカメラ情報を、モデルの表示に適用することにより、正しい位置に表示が行われる。

図４－５３：画像視点抽出画面

① メニュー：IDR_MENU_EXTRACT

メニュー構成： ［ファイル］ ＋［閉じる］終了する ［ヘルプ(H)］ヘルプを表示する

② OpenGL 画面：COrthoVW m_orthoView (orthovw.cpp)

③ 右側ダイアログ

リソース：IDD_DLG_PRM_EXTRACT ハンドラ：CShitenWnd (shitenwn.cpp)

④ ヘルプ：shitenwn.txt

（４９）シーン選択

シーン編集中に、現在存在するシーンの一覧を表示し、選択したシーンに移動する。

なお、画面下の矢印ボタンで、一つ前のシーン、次のシーンに移動することができるため、主に離れたシーンにジャンプする場合に使用する他、編集のために現在までに作成されているシーンを確認するために有用である。

図４－５４：シーン選択画面

リソース：IDD_DLD_SCENELIST　ハンドラ：CSceneList (scenelst.cpp)

（５０）ユーザー定義によるパラメトリック部品の選択

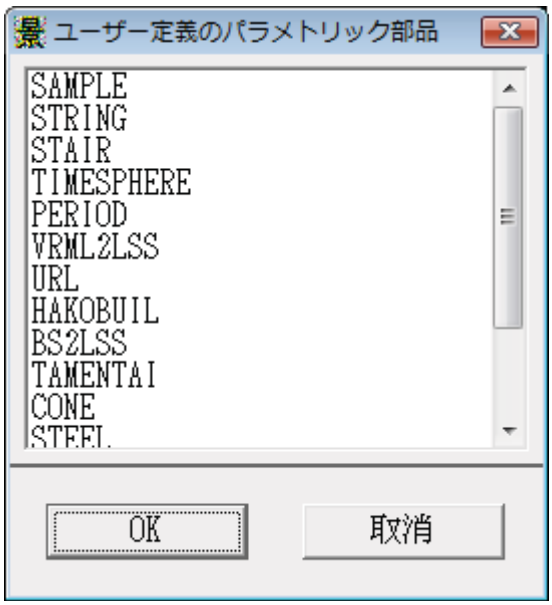


図４－５５：パラメトリック部品選択画面

リソース：IDD_DLD_SCENELIST　ハンドラ：CSceneList (scenelst.cpp)

（５１）面情報

選択したグループの配下にある面に関する情報を表示する。

４－５６：面情報表示画面

リソース：IDD_DLG_FACE　ハンドラ：CFaceinfo (faceinfo.cpp)

（５２）ソリッド分析

一つのグループに属する面群が、閉多面体を形成しているかどうかを検査し、該当する場合には、それによって規定されるソリッドの諸元を計算して表示する。

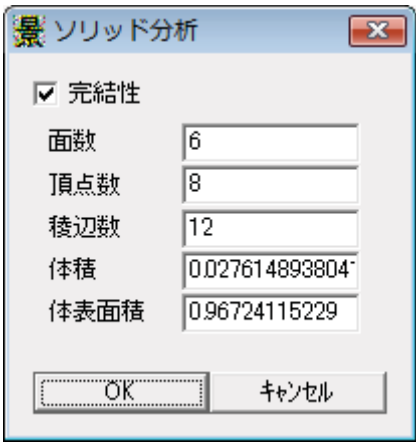


図４－５７：ソリッド分析画面

リソース：IDD_DLG_SOLIDANAL　ハンドラ：CSolidanal (solidanal.cpp)

（５３）単面分析

一つの面を対象として、正常な面であるかどうかを判定する。穴あきポリゴンとなっている場合や、ブリッジで離れた島（飛び地）が連結している場合、全ての頂点が（誤差の範囲で）１平面上にあるかどうか、自己交差しながら、多重ループを形成していないか等についての検査を行う。

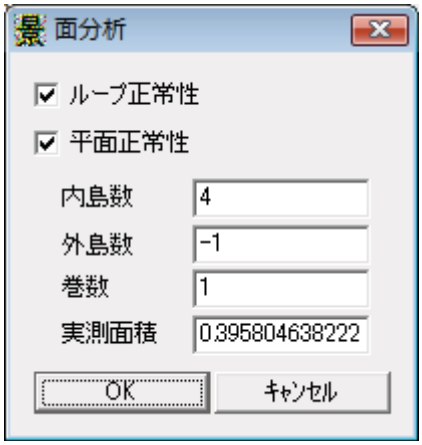


図４－５８：単面分析画面

リソース：IDD_FACEANAL　ハンドラ：CFaceanal (faceanal.cpp)

（５４）頂点検査

一つの面を構成する頂点について逐次的に表示・編集する。頂点の座標のみならず、VERTEX として定義されているテクスチャ座標、法線ベクトル、カラーについて表示し、必要であれば変更する。

図４－５９：頂点検査画面

リソース：IDD_DLG_VANAL　ハンドラ：CVanal (vanal.cpp)

（５５）住宅情報

オブジェクトの属性として、住宅情報が登録されている場合に、表示を行う。通常は、住戸･住宅や敷地の形状を記述する外部ファイルとして配置された FILE コマンドにより記述されたグループに対して属性を定義し、情報そのものは、各戸を記述する外部ファイルではなく、これらを集合的に束ねる上位のグループ（例えば街区）の側に記録する。

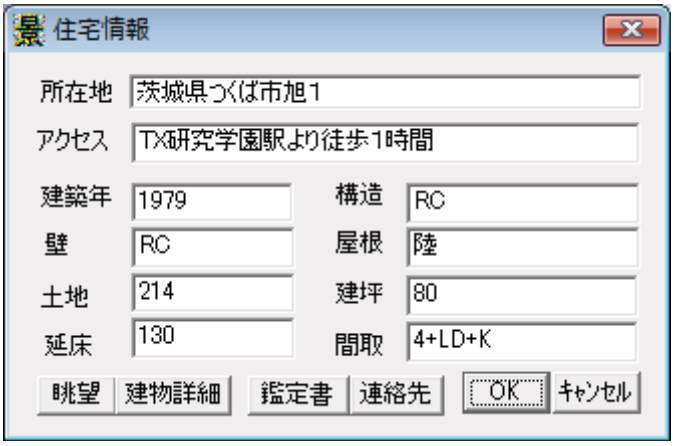


図４－６０：住宅情報画面

リソース：IDD_DIALOG1　ハンドラ：CHouse (house.cpp)

（５６）効果グループ

各シーンに対して一つ定義される効果グループを表示・編集する。効果グループは、複数の効果ユニットの組み合わせとして定義する。

このダイアログにおいては、編集対象のシーンに適用する、効果グループの「新規作成」や、別のシーンのために既に定義されている別の効果グループの再利用を指定するための「選択変更」を行う。更に、現在指定されている効果グループを構成する効果ユニットに関して、その内部構成を編集するダイアログを起動したり、定義済みの効果ユニットを効果グループに追加したり、効果グループを構成する効果ユニットを登録解除・削除する操作を行うことができる。

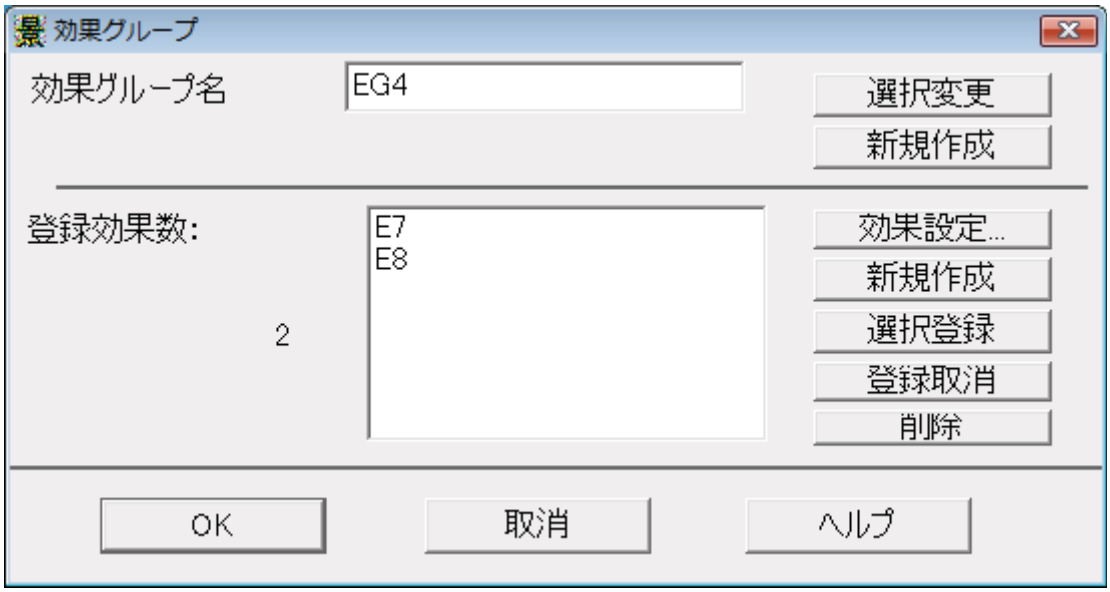


図４－６１：効果グループ編集画面

リソース：IDD_DLG_EFFECT　ハンドラ：CEditEffect (effect.cpp)

（５７）効果ユニット

効果グループを構成する効果ユニットを編集する。多くの場合、効果は表示に関する各種の設定を記録・再現するために用いており、各ダイアログにおける設定を行うことで自動的に設定されるため、ユーザーはその存在を意識する必要はない。このダイアログで表示することにより、具体的にどのようなキーワードとパラメータにより効果ユニットが定義されているかを確認することができる。

システム基幹部分で定義・実装されていない新たなキーワードを定義した場合、表示には何も影響しないが、LSS-S 形式のファイルには保存される。プラグイン DLL などの開発にあたり、シーン表示機能の拡張などを行う際に活用することが可能である。

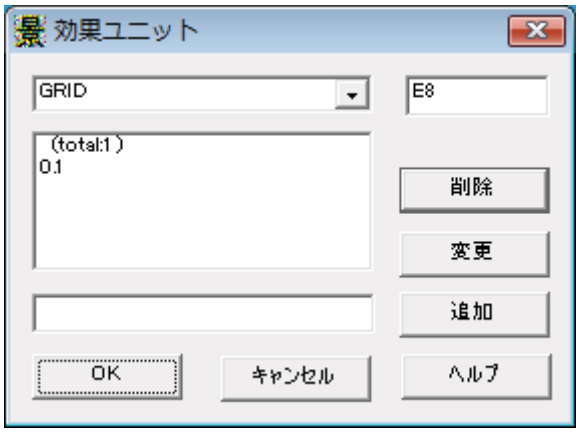


図４－６２：効果ユニット編集画面

リソース：IDD_DLG_EFFECTUNIT ハンドラ：CEffectUnit (effectunit.cpp)

（５８）環境設定

システム起動時に、環境編集 KSIM_ENV で指定した環境設定ファイル（デフォルト名：kdbms.set）で定義された各種の環境設定項目を表示・編集する。編集結果を保存することにより、環境設定ファイルの内容を更新する。このダイアログでファイル保存操作を実行しなかった場合には、元の環境設定ファイルはそのまま維持されるため、次に起動した時点では編集結果は反映されない。

環境設定ファイルの読み込みに際しては、Ver.2.05 以前のバージョンでの定義方法を読み替える処理が行われているため、何も編集せずに上書き保存すると、現在のバージョンにおける定義方法により出力される。従って、本ダイアログから保存した環境設定ファイルを、2.05 以前のバージョンの景観シミュレータ、景観データベースで使用する場合（殆ど無いケースと思われる）、動作は保障されない。

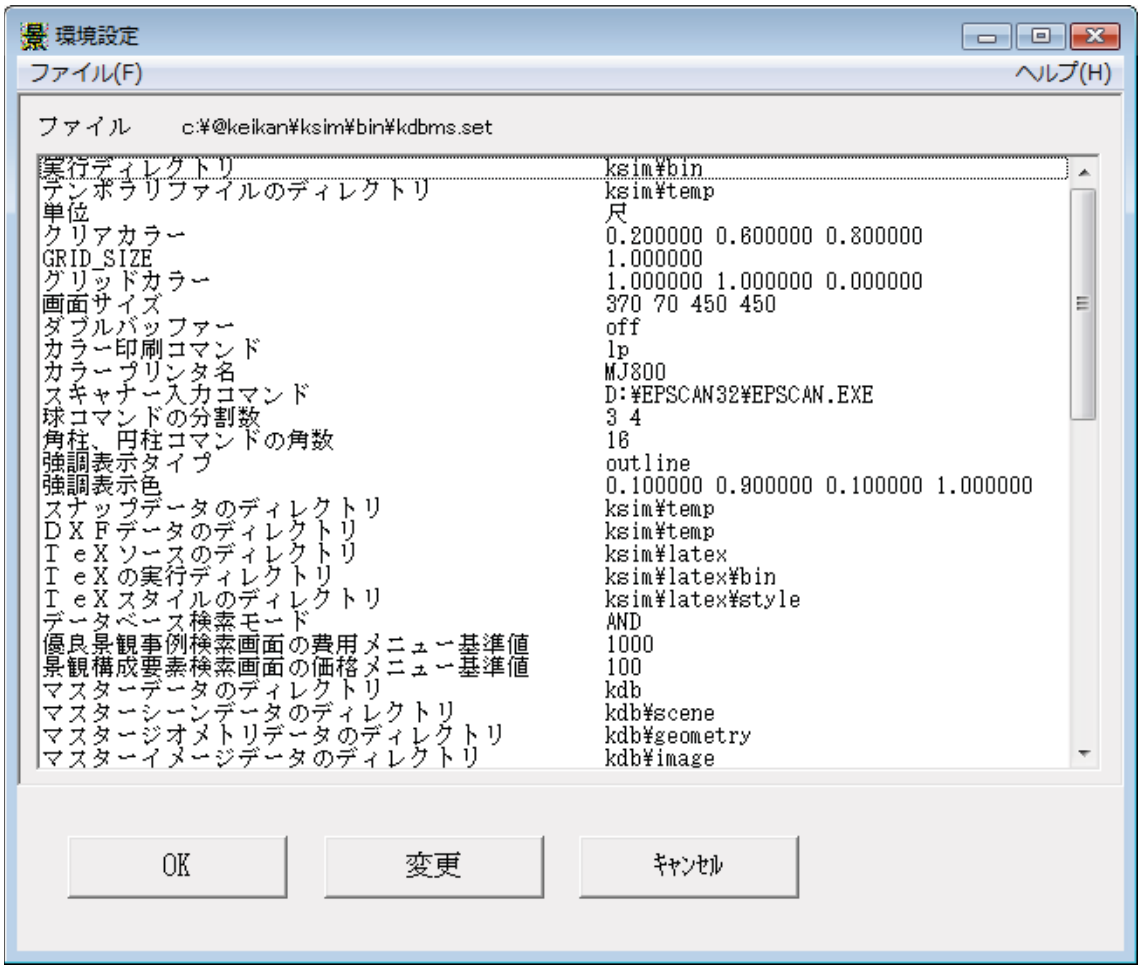


図４－６３：環境設定画面

①メニュー

メニュー構成：IDR_MENU_ENV

［ファイル］

＋［環境設定ファイルを開く］初期化に用いた現在のファイルとは別の環境設定ファイルをロードする。

＋［環境設定ファイルに名前を付けて保存］ファイル名を指定して、保存する

＋［環境設定ファイルの上書保存］読み込んだ名前、または以前保存した名前で保存する

＋［環境編集終了］このダイアログを終了する

②ダイアログ

上部(リスト)　リソース：IDD_MAIN_FORM ハンドラ：CEnvView (envview.cpp)

下部(３ボタン) リソース：IDD_BTN_FORM ハンドラ：CEnvEdit(envedit.cpp)

（５９）環境設定：文字列型

環境設定ダイアログのリストボックスから、一つの設定項目を選択して変更ボタンを押すと、設定内容が文字列型であった場合に起動する。

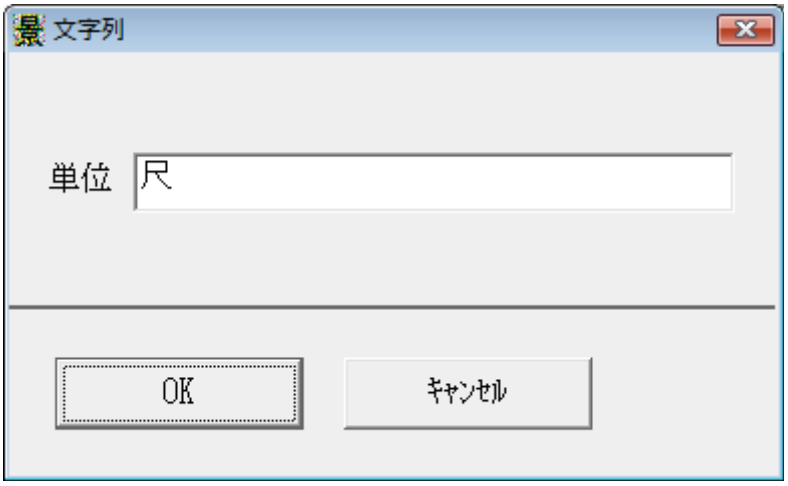


図４－６４：文字列型環境設定項目入力画面

リソース：IDD_DLG_STRING ハンドラ：CStrDlg (strdlg.cpp)

（６０）環境設定：選択型

環境設定ダイアログのリストボックスから、一つの設定項目を選択して変更ボタンを押すと、設定内容が選択型であった場合に起動する。選択肢は、コンボボックスのプルダウンとして表示する。

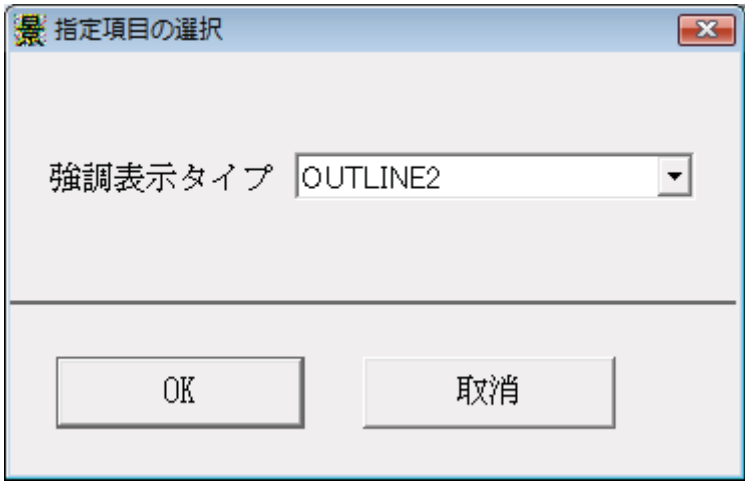


図４－６５：選択型環境設定項目入力画面

リソース：IDD_DLG_SELECT ハンドラ：CSelDlg (seldlg.cpp)

（６１）環境設定：数値・範囲型

環境設定ダイアログのリストボックスから、一つの設定項目を選択して変更ボタンを押すと、設定内容が数値・範囲型であった場合に起動する。数値や範囲の指定方法（項目数）が環境変数によって異なるため、その差異に対応して、必要な数だけ、数値入力のためのエディットボックスを表示する。

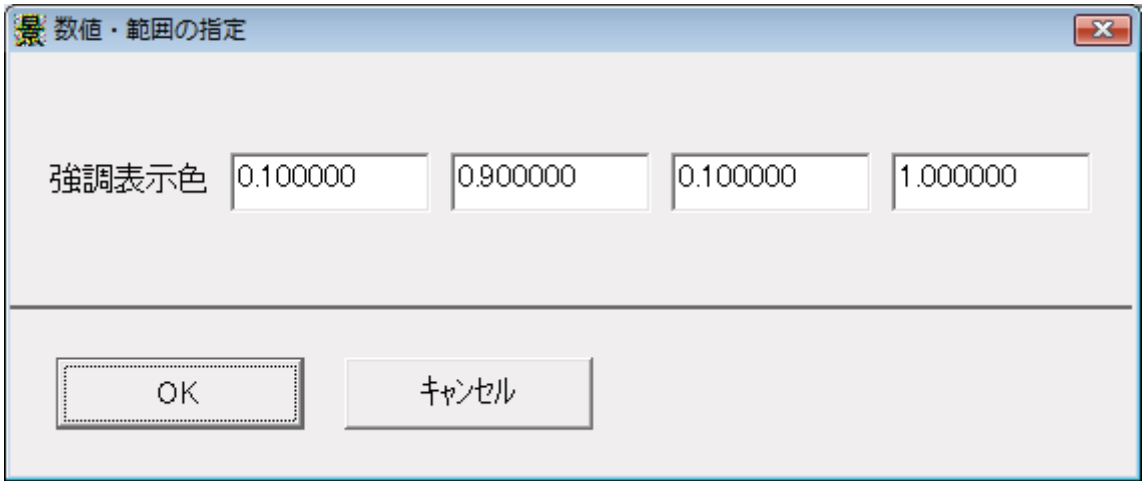


図４－６６：数値・範囲型環境設定項目入力画面

リソース：IDD_DLG_RANG ハンドラ：CRangDlg (rangdlg.cpp)

（６２）線分の選択

現在編集中の地物の中に存在する全ての線データを一覧表示し、その中から特定の線を選択する。また、このダイアログの中で線を削除したり、選択した線と同色の線を全て削除する。線の編集ダイアログ(38)から起動する。

図４－６７：線分の選択画面

リソース：IDD_DLG_LINELIST　ハンドラ：CLineList (linelist.cpp)

（６２）プリンターの選択

開発環境が提供している CView クラスに標準で装備されているメニュー項目を、そのまま利用している。

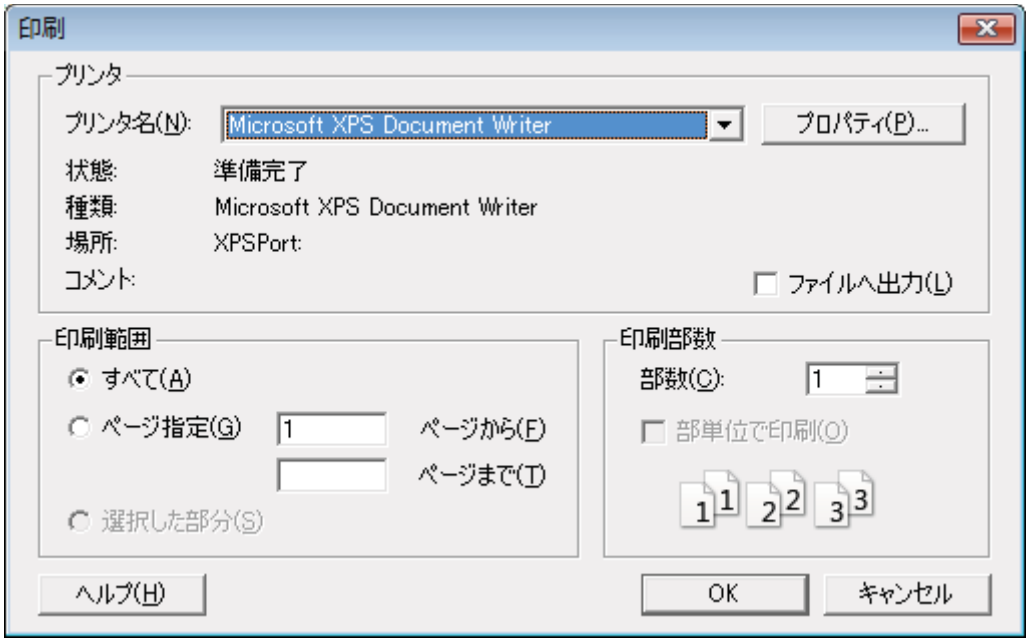


図４－６８：プリンターの選択画面

ハンドラ：CPrintDialog　(MFC の組み込みクラス, dlgprnt.cpp)

（６４）質量

オブジェクトの物理属性の一実装例として用意している。オブジェクトを選択した上で、ポップアップメニューからこのダイアログを開くと、現在選択しているグループに属性として定義されている質量、及び子グループとしてリンクしている配下のグループの合計質量を表示する。下のコンボボックスで、選択階層をルートから最末端の最初に選択したグループまで変更することができる。ルートを選択すると、世界全体の質量となる。

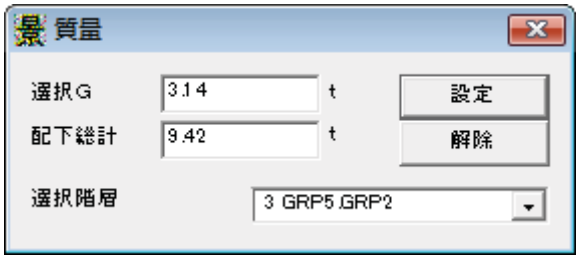


図４－６９：質量画面

リソース：IDD_PHISI　ハンドラ：CPhisi (phisi.cpp)

（６５）炭素含量

オブジェクトの化学属性の一実装例として用意している。オブジェクトを選択した上で、ポップアップメニューからこのダイアログを開くと、現在選択しているグループに属性として定義されている炭素含量、及び子グループとしてリンクしている配下のグループの合計炭素含量を表示する。下のコンボボックスで、選択階層をルートから最末端の最初に選

択したグループまで変更することができる。ルートを選択すると、世界全体の炭素含量となる。

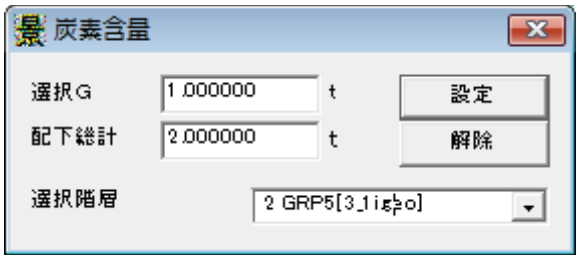


図４－７０：炭素含量画面

リソース：IDD_CHEMI　ハンドラ：CChemi(chemi.cpp)

（６６）データベース情報

景観データベースから選択され、配置されたオブジェクトに関して、データベースに関する情報にアクセスする。

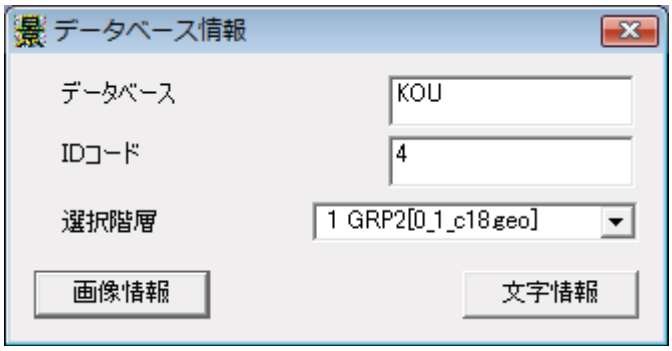


図４－７１：データベース情報画面

リソース：IDD_DBINFO　ハンドラ：CInfoDB (infodb.cpp)

（６７）高速表示設定

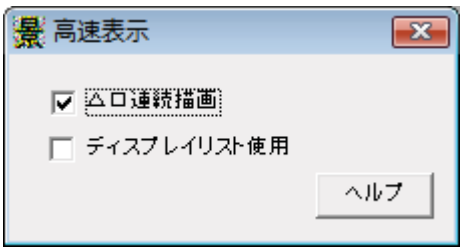


図４－７２：高速表示設定

リソース：IDD_DLG_HIGHSPEED ハンドラ：CHiSpe(HiSpe.cpp)

５．外部関数

５－１．概要

1996 年に公開開始した Ver.2.03 から提供している、ユーザーにより追加可能なモデリング機能の仕組みである。主に、パラメトリックな部品を作成・修正する場合が多い。例えば、中心位置と半径を指定することにより球を生成する。

パラメトリックな部品の意義は、第一に定型化された幾何学図形のデータをコンパクトに記述することにある。例えば、上記の球は、表示段階では多面体として近似する必要があり、しかも不自然でない表示を得るためには十分に多い数の面に分割する必要がある。このような展開されたデータは、保存する際に、大きなファイルを形成する。しかし、パラメトリックな部品として保存すれば、１行の記述で済む。

第二に、上記の球の例のように、球である、という性質が属性として残るため、図形を検索して処理するようなことが可能になる。

第三に、一度作成した部品に対して、パラメータだけを修正するような再編集が可能となる。

このような利点のほかに、景観構成要素の中には、単に座標軸毎の拡大縮小だけでは、必要なサイズが得られないようなものが多い例えば、型鋼であれば、ウェブとフランジの各部の幅と厚さを変える必要がある。階段であれば、蹴上・踏面・幅を歩きやすいサイズに設定したうえで、段数で階高に合わせて調整する。

（１）原始図形

以下のような機能を標準で用意してある。

①cube.exe 直方体
②cylinder.exe 円柱
③flatcyli.exe 角柱
④cone.exe 円錐・円錐台
⑤flatcone.exe 角錐・角錐台
⑥sphere.exe 球
⑦sweep1.exe 掃引体１面（断面と中心線軌跡から立体を生成）
⑧sweep2.exe 掃引体２面（連続する断面をつないで立体を生成）

（２）基本構成要素

①hsteel.exe H 形鋼
②tsteel.exe T 形鋼
③csteel.exe C 形鋼
④lsteel.exe L 形鋼

（３）ファイル形式変換

パラメータにファイル名を指定することにより、ファイル・コンバータにも使用される。
①vrml2lss.exe VRML ファイル
②basic2lss.exe 建築確認申請形式
③fire2lss.exe 延焼シミュレーション形式
④scadec2lss.exe 電子納品形式
出力に関しては、現在は基幹部分のビルドに組み込んでいる（dxf, vrml, fire)。

（4）ネットワーク・アクセス

特殊な用途として、URL を引数として、インターネット上のファイルをダウンロードする関数がある。
①geoload.exe　URL からのファイルのダウンロード

（5）これまでに実装されているユーザー定義によるパラメトリックな図形

①sample.exe 切妻屋根
②stair.exe 階段
③tamentai.exe 5 種類の正多面体
④segitiga.exe 3 辺の長さから三角形を生成する
等

これらの関数は、環境設定ファイル kdbms.set の BIN_PATH エントリーで定義されたディレクトリに置かれ、かつ利用可能な外部関数のリストと引数構成を登録するファイル ext.tab(これも同じディレクトリに置かれる)に登録することにより利用可能になる。DOS コマンドとして、パラメータを引数として起動され、LSS-G 形式のファイルを出力する。

外部関数は､LSS-G 形式の外部ファイルにおける FILE コマンドにより参照された場合、インタープリタ（IP ライブラリ）から起動される（3－3 参照)。

また、通常の場合、外部関数は、パラメータを設定するためのダイアログのための別の実行形式（関数名_D.exe）を随伴している。これにより、モデリング作業の中で、外部関数を用いて形状生成を行うことができる。また編集結果を LSS-G 形式に保存する際には、FILE コマンドの形で記録される。

ユーザーが、メニューの[形状生成][オプション]を起動した場合、または、配置ダイアログの中での部品選択において外部関数が選択された場合に、ext.tab に登録された関数の一覧がまず表示される。次に、この一覧の中からいずれかの関数が選択されると、それぞれの関数に付随した、パラメータ設定のためのダイアログが起動される。

また、ユーザーが、一度生成したパラメトリック部品を選択した上で、[形状生成][オプション]を選択すると､現在設定されているパラメータを表示した状態でダイアログが開く。ユーザーがパラメータを変更して実行すると、これに対応して図形が修正される。例えば球の半径を修正する、といった具合である。

パラメトリックな部品は、メモリ上では、具体的な形状に展開された形でデータを形成しているが、ファイル保存に際しては、これらは捨象され、関数名とパラメータだけがLSS-G 形式のファイルに保存されるため、たとえば球のように多くの面に分割されて表示される部品は、コンパクトな形で保存される。但し、パラメトリックな部品に対して切り欠きなどの処理が行われた後は、パラメトリックな部品として記述し切れない情報を含むため、一般の図形と同様の扱いとなり、全ての展開された構成要素を保存する。このことは、一般のファイル型の部品と同様である。また、一つのファイルから参照されたパラメトリックな部品がその属性を失った場合には、上位のファイルもファイル属性を失うこととなる。

５－２．ダイアログ部の起動

起動時の処理は、メイン画面において既存のオブジェクトが選択されているか否かにより、異なる処理を行っている。

（１）既存オブジェクトが選択されておらず、一覧を表示する場合：

パラメータ設定ダイアログ表示に先立って、まずシーン選択ダイアログ CSceneLst のメンバ関数の機能を共用して外部関数の一覧を表示する。

```
m_dispScene->SetListType(K_OPTION);
m_dispScene->ShowWindow(SW_SHOW);
```

このダイアログで、ユーザーがいずれかの外部関数を選択しOKを押した場合には、ダイアログの表示を行う。

パラメータ設定ダイアログ表示に先立って、CMainFrame::OnMenuitemOption()関数の中で、前回に同じパラメータ設定ダイアログ作成した、下記の一時的なファイルを調べ、もし存在していれば、この形状を生成した際のコマンドを復原し、また存在していなければデフォルト値でコマンドを新たに作成する。これを kata 文字列として buOption 関数に渡す。これにより、ダイアログ部では、前回使用した際のパラメータを参照しながら、新たなパラメータを設定することができる。パラメータ部の起動は以下による。

```
m_dispScene->buOption(mode,kata);
```

mode には、一覧表（配列）の中での関数の順位 ID 整数値が入る。

kata 文字列には、現在のパラメータを示す１行が入る。

例：GRP2=FILE(SPHERE,0,0,0,1);　　（原点に中心がある半径１の球）

これらを初期表示値として、ダイアログ部を CreateProcess 関数により起動すると同時にタイマをセットし、OnTimer() 関数の中で、ダイアログ部の終了を監視する。

ダイアログ部が OK で終了した場合には、temp ディレクトリの中に、パラメータを格納した小さなファイルを作成する。

例：c:¥@keikan¥ksim¥temp¥sample.geo

このファイルは、以下のような１行のみを含む小さなファイルである。

GRP4=FILE(SAMPLE,1.000,2.000,3.000);

この１行のコマンドを文字列 dialog に取得し、これを用いて、

IP_interpret(dialog)　　(i3 ライブラリの関数)

を実行することにより、インタープリタのライブラリ関数の処理の中で、実際の形状を生成する関数を起動し、形態を生成し表示を更新する。

形状生成関数の名称を XXX(.exe)とすると、ダイアログ部を記述する関数は、XXX_D.exe という名称を有し、共に bin ディレクトリに置かれる。

（２）既存部品の再編集

画面上であるパラメトリックなオブジェクトを選択し強調表示されている状態で、プルダウンメニューの[形状生成][オプション]、またはポップアップメニューの[オプション]が選択されると、それぞれのオブジェクトに対応するパラメータ設定ダイアログを起動し、現在のパラメータを表示する。この処理も CSceneLst->buOption 関数が行う。新規作成の場合との違いは、外部関数の側では、引数として渡されたグループ名称と、ファイルとして渡されたコマンド（temp/XXX.geo）に記述されたグループ名称を比較することで認識する。即ち、グループ名称が同一であれば、以前に生成したオブジェクトの再編集であると認識する。この場合には、グループ名称の変更は禁止する（エディットボックスを編集不可とし、その結果グレー表示となる)。

意図的に、あるいは何らかの錯誤により形状生成の関数のみが存在し、ダイアログ部の関数が存在しない場合について、特例的に（３）または（４）のような処理を行っている。

（３）別のダイアログ関数へのリダイレクト

例えば型鋼のように、一つのダイアログで４種類の型鋼についてパラメータを設定するような場合、EXT_TAB には、HSTEEL、LSTEEL、CSTEEL、TSTEEL の４種類が登録され、それぞれに対応する形状生成関数 hsteel.exe、lsteel.exe、csteel.exe、tsteel.exe がセットアップされている。これに対して、パラメータ設定のダイアログは、steel_D.exe の一つしかない。このような場合、bin ディレクトリに、hsteel_D.exe の代わりに、hsteel_D.ijk という小さなテキスト・ファイルを置き、この中に Steel_D.exe という、処理を付託したい実行形式の名称を１行で記述しておく。基幹部分が関数の起動（メニューからの選択、あるいは既存パラメトリック部品の再編集）を行おうとする時に、hsteel_D.exe が存在しないことを検知した後、hsteel_D.ijk の有無を確かめ、もし存在していれば、その内容から、同じ bin ディレクトリにある、付託先の Steel_D.exe を起動する。この起動は通常と同じように行われるため、多言語処理も同様に行うことができる。またシステム終了時にこのダイアログが開いていた場合には、強制終了することができる。

別の方法として、たとえば上記の例で Steel_D.exe を起動するだけの機能を有する小さな Hsteel_D.exe を作成し bin ディレクトリに置いておくことにより、Steel_D.exe を起動しパラメータを表示することができる。但し、この場合、ダイアログが開いたままの状態で基幹部分 sim.exe が終了した場合に、Hsteel_D.exe だけが終了し、Steel_D.exe は開いた

ままの状態となる。また、多言語処理機能において、Steel_D.exe が新たな言語に翻訳されたり、コントロールの追加が行われたりしたような場合において、外部関数用の xml ファイルの鮮度管理における適切な処理が行われない。

（4）ダイアログ部の実行形式が存在しない場合

ダイアログの実行形式 XXX_D.exe が bin ディレクトリに存在しない場合には、環境設定ファイル kdbms.set の EXTERNAL_PATH エントリーで定義されたディレクトリの名称(文字列)を調べ、これが http://で始まる URL となっている場合には、このサイトにダウンロードしに行く。ダウンロードに成功すれば、実行する。

もしこのディレクトリ名称がローカルなディレクトリであった場合、または URL からのダウンロードに失敗した場合には、最後の手段として、default_D.exe を起動し、形状生成コマンドを 1 行のテキストとして表示する。ユーザーは手入力により、コマンドラインを作成し、OK を押すことにより、新規作成または既存部品のパラメータ変更を行うことができる。この機能は、ダイアログをまだ作成していない外部関数の機能テストなどを行う際に有用である。

（5）ファイル選択ダイアログだけを有する会話部

VRML 形式など、外部ファイルを変換するだけの機能を有する外部関数のダイアログは、固有の画面構成を必要とせず、OS と開発環境で提供されているファイル選択ダイアログを開くだけで十分である。このような簡単なダイアログは、コピーし名称変更するだけで、他の外部関数に対しても使用することができる。

（6）ダイアログ部における、メイン画面との協調動作

ダイアログ部でのパラメータ入力に際して、メイン側の座標を、画面クリック操作により取り込み、パラメータ値として使用することができる。クリックされたポイントの座標値は、一時的なファイルにより会話部に渡される。このファイルの名称は、会話部の起動に際して、引数として渡しているので、会話部では環境設定ファイル等を取得・解読する必要はない。

具体的にはダイアログにおいては、メイン画面がオルソ系の表示（平面図、立面図）となっている場合に、ユーザーがクリックした地点の座標値を利用することができる。

メイン側では、CDrawFrm::OnLButtonUp 関数の中で、外部関数のダイアログが起動されている時に、オルソ系の画面がクリックされると、tmp ディレクトリに Option.dat というファイルが作成され、その中に、XYZ 座標値が格納され、更にこれが新鮮であることを示すために、Option.flg という小さなファイルが作成され、1 の値が書き込まれる。

外部関数のダイアログにおいては、タイマ割り込みで、Option.flg の中身を調べ、1 が立っていた場合には、Option.dat のデータを取り込み、選択されている値のエディットボックスに書き込むことができる。データを取り込んだ後、Option.flg に 0 を書き込むことで賞味期限切れを明示することにより、同じデータは 1 回だけしか利用しない。

（7）外部関数の無いダイアログ部

やや、トリッキーな方法ではあるが、ダイアログ部に豊富な編集機能を持たせ、形状などを生成する外部関数を使用しない編集環境の増設方法も可能である。例えば、ダイアログ部で、部品の検索や選択のみを行い、結果として生成する一時的なファイルにおいては、ユーザーが選択した固定的なファイルを参照するのみとするような方法である。

このような処理により指定された固定的なファイルは、パラメトリックではないため、パラメータ再編集の対象とはならない。

（8）外部関数ダイアログ部の多言語処理

本体基幹部分(sim.exe)と同様に外部関数も多言語表示となっていることが望ましい。

外部関数は、bin ディレクトリにある Lang.ctl を参照することにより、使用する言語を使い分けることができる。外部関数の多言語処理のための xml ファイルは、最初に外部関数ダイアログ部が起動された時に、基幹部分が、リソースファイルとリソースのヘッダーから作成する。これに各言語の翻訳を入力することにより、各言語での表示を行うことができる。実際のメニューやコントロールの言語処理は、ダイアログの OnInitDialog 関数の中で実行している。また、2バイト系と1バイト系言語でのレイアウトの変更に関しては、リソース中にレイアウトの異なるダイアログ・テンプレートを用意しておき、ダイアログの構築部 Cxxxx_DDlg::Cxxxx_DDlg() 関数の中で、m_lpszTemplateName を設定する方法で選択している。外部関数の側で XML から各コントロールを設定する処理などは、LocalLang.lib ライブラリに用意してあるので、外部関数の作成にあたって、これをリンクする。

外部関数の多言語表示の詳細については、１１．多言語処理で解説した。

５－３．関数部の起動による実際の形状生成

上記の IP_interpret 関数(i3ip.c)を起点とする一連の形状生成処理の中で、i3_file 関数(I3iplssg.c)が、実際の形状を生成している。i3_file 関数は、CreateProcess システム関数を用いて、外部関数（この例では sample.exe）を起動し、起動された外部関数は、temp ディレクトリの中に、sample.g という名称で、オブジェクトを実際の形状を LSS-G 形式に従い、グループや面として表現したファイルを作成する。その後、制御が sim.exe の側に戻り、この成果を、通常の固定的な部品と同様にして取り込み、景観の中に配置する。

上記の CreateProcess による外部関数の起動に際して、各関数に渡される引数は、以下の通りである：

①各パラメータ（関数の種類に応じた数と形式。ext.tab の定義に従う）

②形状生成結果を格納するファイル名（フルパスで指定する）

③球、円柱、円錐の場合における分割数（環境設定ファイル kdbms.set の設定内容を転送）

５－４．外部関数の引数の種類と意味

外部関数の引数の型については、ext.tab の中で定義され、システムに通知される。

①INT 整数型

外部関数に渡される引数（文字列）が、整数として解釈される

②DOUBLE 倍精度実数

外部関数に渡される引数（文字列）が、倍精度実数として解釈される

③STRING 文字列

外部関数に渡される引数（文字列）が、文字列として解釈される

④FILE ファイル参照

外部関数に渡される引数（文字列）が、ファイル参照として解釈される。

ファイル名がフルパスで指定されていなかった場合、環境設定ファイル kdbms.set の定義に従い、以下のディレクトリを調べに行く。

1) 作業環境が設定されている場合、作業用ディレクトリ
2) FILE_PATH_JIREI_GEOMETRY （景観検討事例データベースの格納先）
3) FILE_PATH_YOUSO_GEOMETRY（景観構成要素データベースの格納先）
4) FILE_PATH_ZAIRYO_GEOMETRY （景観材料データベースの格納先）
5) FILE_PATH_MASTER_GEOMETRY（通常のユーザーがデータを保存する場所）
6) URL 上のデータ

この内、２～４を調べに行く順序は、配置ダイアログにおける配置部品のデータベースからの検索等の際に設定される。URL アクセスは、ウェブ・ブラウザから起動した状態で、LSS-S ファイルをダウンロードした際に、その中に定義されているアドレスを基に、以下の LSS-G ファイルを取得する。

⑤FACE 面

外部関数に渡される引数（文字列）が、直前に定義された面を参照する。具体的には、FILE 文の中に定義されたラベルを有する面が一時的なファイルに出力され、そのファイル名が外部関数に引数として渡される。

⑥LINE 線

外部関数に渡される引数（文字列）が、直前に定義された線を参照する。具体的には、LINE 文の中に定義されたラベルを有する面が一時的なファイルに出力され、そのファイル名が外部関数に引数として渡される。

⑦GROUP グループ

外部関数に渡される引数（文字列）が、直前に定義されたグループを参照する。具体的には、GROUP 文の中に定義されたラベルを有する面が一時的なファイルに出力され、そのファイル名が外部関数に引数として渡される。

⑧TIME 時間

システムの現在時刻が、引数として外部関数に渡される。LSS-G コマンド中に記述された引数は無視される。例えば、外部関数「球」は、 ext.tab の中で、

FILE(SPHERE, DOUBLE, DOUBLE, DOUBLE, DOUBLE);

と定義されている。

この状態で、原点を中心に、半径１の球を生成することは、

GRP1 = FILE(SPHERE, 0, 0, 0, 1);

というコマンドを発行することに等しい。

ここで、関数はこのままとして、定義だけを

FILE(SPHERE, DOUBLE, DOUBLE, DOUBLE, TIME);

に修正すると、上記と同じコマンドにより、半径が時間に比例する球が生成する。

５－５．エラー処理

外部関数による形状生成は、多くの工程から成るため、多くの段階でのエラー処理が必要である。

①選択した外部関数に関するダイアログ部が存在しない場合

ダイアログ部（xxxx_D.exe）を起動するに先立って、その存在を access 関数で確認している。この段階で、存在しないことが確認された場合には、環境設定ファイル KDBMS.SET で指定したサーバーからのダウンロードを試みる。それに失敗すると、Default_D.exe という、コマンドラインでパラメータを設定するダイアログを表示する。Default_D.exe が存在しない場合にはサーバーからのダウンロードを試みる。これに失敗した場合、メッセージ（番号 1398）を表示してアイドリングに戻る。ダウンロード関数が正常に終了したにもかかわらず、結果のファイルがキャッシュに存在しない場合、メッセージ(番号 1399)を出す。ダウンロードに成功した場合には、メッセージ（番号 5369）を出して、Default_D.exe を開く。それと同時にタイマをセットし、定期的にダイアログ部の終了を確認する。ダイアログ終了確認関数がエラーを返した場合には、メッセージ（番号 1331）を出してアイドリングに戻る。プロセスの終了を検出した場合には、タイマ割り込みを終了させ、次の工程に進む。

②ダイアログがキャンセル終了した場合

終了コードが２であることにより認識される。アイドリング状態に戻る。

③ダイアログに入力されたパラメータの値が不適切である場合

ダイアログの OnOK でチェックされた場合には、ダイアログの中でメッセージを出して処理を続行する（ダイアログの中での入力待ち）。

④ダイアログが OK 終了したにもかかわらず、結果を格納した

TEMP/XXX.geo

ファイルが存在しない場合、メイン側でエラーメッセージ（番号 3370）を出し、アイドリング状態になる。

⑤外部関数による形状生成が異常終了した場合

外部関数は、インタープリタのコマンド（１行のみ）として実行される。保存された LSS-G ファイルの中に含まれている外部関数の参照の処理と、同様の扱いとなる。

リターン値を認識した上で、エラーメッセージをエラーログに出し、処理を続行する。
⑥外部関数が正常に終了した後､生成されている筈の TEMP/xxxx.g ファイルが存在しない場合：メッセージを出して、インタープリタの処理を続行する。なお、インタープリタのエラーは、多数存在する場合にその都度エラーメッセージを出すと、終了するまでに多数回 OK ボタンを押さなければならなくなり、事実上終了不能となることから、エラーログの形でテキスト・ファイルを作成し、１以上のエラーが発生した場合に、ロード終了後にこのテキスト・ファイルを表示する方法を採用している。

５－６．ヘルプ

パラメータ設定ダイアログで表示すべきヘルプ・ファイルは、現在選択されている言語に依存する。そこで、多言語機能を外部関数ダイアログ部の多言語対応機能を提供している LocalLang.lib ライブラリの中に、Bantu()関数を用意し、この関数をユーザーのメニュー選択またはヘルプボタンの操作に際して起動することにより、対応する言語でのヘルプ・ファイルを表示する。

具体的には、Language ディレクトリの下の言語別のディレクトリにある、

関数名.言語コード.txt

という名称のファイルを表示する。このディレクトリの所在は、本体基幹部分で言語選択・変更操作が行われた時点で、bin ディレクトリに作成される Lang.ctl というファイルを参照することにより取得することができる。

５－７．外部関数のプログラム例

形状生成を行う外部関数は、ダイアログ等の画面表示を必要としないため、コマンドライン・ベースのプログラムで十分である。

リスト５－１：外部関数の例（切妻屋根の形状生成）

```
#include <stdio.h>
#include <math.h>
#include <stdlib.h>

int main(int argc, char *argv[]){
          int i;
          double X,Y,D,H;
          FILE *wp;
          if(argc != 4+1) return 2;//戻り値２：引数の数が不一致
          for(i=0;i<argc;i++){
                    printf("%d:[%s]¥n",i,argv[i]);
          }
          X = atof(argv[1]);
          Y = atof(argv[2]);
          D = atof(argv[3]);
          H = 0.5*Y*D;
          wp = fopen(argv[4],"wt");
          if(!wp){
                    printf("出力ファイル[%s]開かず",argv[4]);
                    exit( 3 ); //戻り値３：出力ファイルが開かない
```

```
	}else{
		fprintf(wp,"P0 = COORD(0.0,0.0,0.0);¥n");
		fprintf(wp,"P1 = COORD(%0.3lf,0.0,0.0);¥n",X);
		fprintf(wp,"P2 = COORD(%0.3lf,%0.3lf,0.0);¥n",X,Y);
		fprintf(wp,"P3 = COORD(0.0,%0.3lf,0.0);¥n",Y);
		fprintf(wp,"P4 = COORD(0.0,%0.3lf,%0.3lf);¥n",0.5*Y,H);
		fprintf(wp,"P5 = COORD(%0.3lf,%0.3lf,%0.3lf);¥n",X,0.5*Y,H);
		for(i=0;i<6;i++){
			fprintf(wp,"V%d=VERTEX(P%d);¥n",i,i);
		}
		fprintf(wp,"F0 = FACE(V0,V1,V5,V4);¥n");
		fprintf(wp,"F1 = FACE(V4,V5,V2,V3);¥n");
		fprintf(wp,"ROOT = GROUP();¥n");
		fprintf(wp,"GROUP_FACE(ROOT,F0,F1);¥n");
		fclose(wp);
	}
	return 1;//正常終了の場合、１を返す。
}

/*------------------------------*
戻り値return またはexit
0:未定義のエラー
1:正常終了
2:引数の数が合わない
3:出力ファイルが開かない
#endif
*------------------------------*/
```

main 関数に渡される argv で引数を受ける。argv[0] は、関数名自身であるため、argv[1] 以下で引数（パラメータ）を受け取る。最後の引数の次に、変換した結果を格納するファイル名を受ける。この名称のファイルを書き込みモードで作成し、その中に、パラメータに従って生成した形状を、LSS-G 形式のコマンドを用いて記述する。

エラーコードは、上記例に示したように、１が正常終了、２以下が、定形化されたエラー、０が未定義のエラーである。

５－８．ダイアログ部のプログラム例

ダイアログ部は、パラメータを入力するための画面が必要とするため、WinMain から始まるプログラムの中に、ウィンドウの定義と、コールバック関数を記述する。下記の例では、sample_D.cpp と sample_DDlg.cpp の二つのソースコードから構成し、前者の中で、引数として渡された初期値の解析を行い、ダイアログの初期化を行う。

ダイアログが OK で終了した場合には、exit(1)、Cancel で終了した場合には、exit(2)で終了している。メイン側では、ダイアログの終了をタイマ割り込みで監視しており、OK で正常終了したことを検出した場合には、次の処理を続行する。

リスト５－２：外部関数ダイアログ起動部のプログラム例（切妻屋根形状生成）

```
// sample_D.cpp : アプリケーション用クラスの定義を行います。
//#ifdef MULTI_LANG ～ #endif の間は、多言語対応のための追加部分です
#include "stdafx.h"
#include "sample_D.h"
#include "sample_DDlg.h"
```

```
#ifdef MULTI_LANG
#include "LocalLang.h"
#endif

#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif

/////////////////////////////////////////////////////////////////////////////
// CSample_DApp
BEGIN_MESSAGE_MAP(CSample_DApp, CWinApp)
          //{{AFX_MSG_MAP(CSample_DApp)
                    // メモ- ClassWizard はこの位置にマッピング用のマクロを追加または削除します。
                    //        この位置に生成されるコードを編集しないでください。
          //}}AFX_MSG
          ON_COMMAND(ID_HELP, CWinApp::OnHelp)
END_MESSAGE_MAP()

/////////////////////////////////////////////////////////////////////////////
// CSample_DApp クラスの構築
CSample_DApp::CSample_DApp()
{
          // TODO: この位置に構築用のコードを追加してください。
          // ここにInitInstance 中の重要な初期化処理をすべて記述してください。
}

/////////////////////////////////////////////////////////////////////////////
// 唯一のCSample_DApp オブジェクト
CSample_DApp theApp;

/////////////////////////////////////////////////////////////////////////////
// CSample_DApp クラスの初期化
char filename[80];
char groupname[80];
char flagfilename[80];
char datfilename[80];

BOOL CSample_DApp::InitInstance()
{
int nparam; //990818 DR.H.K.
          nparam = 0;
          if(m_lpCmdLine[0] != '"')
                    nparam = sscanf(m_lpCmdLine,"%s %s %s %s",
                              filename, groupname, flagfilename, datfilename);
          else{
                    unsigned int i,j;
                    for(j=0,i=1;m_lpCmdLine[i]!='"';){
                              filename[j++] = m_lpCmdLine[i++];
                    }
                    filename[j] = '¥0';
                    nparam++;
                    if(strlen(m_lpCmdLine)<i) goto SELESAI;
                    i++;
                    for(;m_lpCmdLine[i]!='"';i++);
                    i++;
                    for(j=0;m_lpCmdLine[i]!='"';){
                              groupname[j++]=m_lpCmdLine[i++];
                    }
                    groupname[j] = '¥0';
                    nparam++;
                    if(strlen(m_lpCmdLine)<i) goto SELESAI;
```

```
                        i++;
                        for(;m_lpCmdLine[i]!='"';i++);
                        i++;
                        for(j=0;m_lpCmdLine[i]!='"';){
                                flagfilename[j++]=m_lpCmdLine[i++];
                        }
                        flagfilename[j] = '¥0';
                        nparam++;
                        i++;
                        if(strlen(m_lpCmdLine)<i) goto SELESAI;
                        for(;m_lpCmdLine[i]!='"';i++);
                        i++;
                        for(j=0;m_lpCmdLine[i]!='"';){
                                datfilename[j++]=m_lpCmdLine[i++];
                        }
                        datfilename[j] = '¥0';
                        nparam++;
                }
SELESAI:
                if(nparam < 2){
                        char s[80];
#ifdef MULTI_LANG
                        lsKanjiDlg("sample");//これによりXMLデータの初期化を行う
                        sprintf_s(s,70,Kanji("K02").GetBuffer(),nparam);
                        MessageBox(NULL,s,Kanji("K01").GetBuffer(),MB_OK);
#else
                        sprintf_s(s,70,"引数の数[%d]は最低限あるべき数[2]より少ない",nparam);
                        MessageBox(NULL,s,"切妻屋根",MB_OK);
#endif
                        return FALSE;
                }
                if( nparam < 4){
                        lsKanjiDlg("sample");//これによりXMLデータの初期化を行う
                        *flagfilename = *datfilename = 0;
#ifdef MULTI_LANG
                        MessageBox(NULL,Kanji("K03"),Kanji("K01"),MB_OK);
#else
                        MessageBox(NULL,"情報交換ファイルが未定義","切妻屋根",MB_OK);
                        //MessageBox(NULL,filename, groupname,MB_OK);
#endif
                }

CSample_DDlg* dlg = new CSample_DDlg;
                m_pMainWnd = dlg;
                int nResponse = dlg->DoModal();
                delete dlg;
                if (nResponse == IDOK)
                {
                        // TODO: ダイアログが<OK> で消された時に、exit(1)で終了
                        //       記述してください。
                        exit(1); //980613 DR.H.K. 下記参照之事
                }
                else if (nResponse == IDCANCEL)
                {
                        // TODO:980613 DR.HK. ダイアログが<キャンセル> で消された時に、exit(2) で終了
                        exit(2);
                }
                return FALSE; //default
}
```

後者の中では、ユーザーがダイアログに対して行った各種操作に対応した処理を記述している。

リスト５－３：外部関数ダイアログのコールバック部プログラム例（切妻屋根形状生成）

```
// ユーザー定義によるパラメトリック部品のダイアログ部雛形(SAMPLE)
// #ifdef MULTI_LANG ～#endif の間が、多言語対応のための追加部分です。

// sample_DDlg.cpp : インプリメンテーションファイル
#include "stdafx.h"
#include "sample_D.h"
#include "sample_DDlg.h"
#include <io.h>

#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif

extern char groupname[]; //何故か、*groupname では落ちる
extern char filename[];

#ifdef MULTI_LANG
#include "LocalLang.h"
//#include "tinyxml.h"
extern CSample_DApp theApp;
CWinApp* thisApp = &theApp;
#endif
/////////////////////////////////////////////////////////////////////////////
// アプリケーションのバージョン情報で使われているCAboutDlg ダイアログ

class CAboutDlg : public CDialog
{
public:
          CAboutDlg();

// ダイアログデータ
          //{{AFX_DATA(CAboutDlg)
          enum { IDD = IDD_ABOUTBOX };
          //}}AFX_DATA

          // ClassWizard は仮想関数を生成しオーバーライドします
          //{{AFX_VIRTUAL(CAboutDlg)
protected:
          virtual void DoDataExchange(CDataExchange* pDX);     // DDX/DDV のサポート
          virtual BOOL OnInitDialog();
          //}}AFX_VIRTUAL

// インプリメンテーション
protected:
          //{{AFX_MSG(CAboutDlg)
          //}}AFX_MSG
          DECLARE_MESSAGE_MAP()
};

CAboutDlg::CAboutDlg() : CDialog(CAboutDlg::IDD)
{
          //{{AFX_DATA_INIT(CAboutDlg)
          //}}AFX_DATA_INIT
#ifdef MULTI_LANG
          if(IsKanjiDlg("sample"))
                    m_lpszTemplateName = MAKEINTRESOURCE( IDD_ABOUTBOX );
          else
                    m_lpszTemplateName = MAKEINTRESOURCE( IDD_ABOUTBOX_0 );
#endif
```

```
}

void CAboutDlg::DoDataExchange(CDataExchange* pDX)
{
	CDialog::DoDataExchange(pDX);
	//{{AFX_DATA_MAP(CAboutDlg)
	//}}AFX_DATA_MAP
}

BOOL CAboutDlg::OnInitDialog()
{
	CDialog::OnInitDialog();
	// "バージョン情報..." メニュー項目をシステムメニューへ追加します。
#ifdef MULTI_LANG
	if(IsKanjiDlg("sample")){
		MultiLangDialogConv("IDD_ABOUTBOX", (CDialog*)this);
	}else{
		MultiLangDialogConv("IDD_ABOUTBOX_0", (CDialog*)this);
	}
#endif

	return TRUE;  // TRUE を返すとコントロールに設定したフォーカスは失われません。
}

BEGIN_MESSAGE_MAP(CAboutDlg, CDialog)
	//{{AFX_MSG_MAP(CAboutDlg)
		// メッセージハンドラがありません。
	//}}AFX_MSG_MAP
END_MESSAGE_MAP()

/////////////////////////////////////////////////////////////////////////////
// CSample_DDlg ダイアログ

CSample_DDlg::CSample_DDlg(CWnd* pParent /*=NULL*/)
	: CDialog(CSample_DDlg::IDD, pParent)
{
	//{{AFX_DATA_INIT(CSample_DDlg)
	m_GroupName = _T("");
	m_Param1 = 0.0;
	m_Param2 = 0.0;
	m_Param3 = 0.0;
	//}}AFX_DATA_INIT
	// メモ: LoadIcon はWin32 のDestroyIcon のサブシーケンスを要求しません。
	m_hIcon = AfxGetApp()->LoadIcon(IDR_MAINFRAME);
	//sample_D¥rc にsample_D.ico が登録されている
	//リソースsample_D.rc の中で、IDR_MAINFRAMEが定義され、上記ファイルが引用されている
#ifdef MULTI_LANG
	if(IsKanjiDlg("sample"))
		m_lpszTemplateName = MAKEINTRESOURCE( IDD_SAMPLE_D_DIALOG );
	else
		m_lpszTemplateName = MAKEINTRESOURCE( IDD_SAMPLE_D_DIALOG_0 );
#else
//	BOOL rc = Create(IDD,NULL);
	m_hIcon = AfxGetApp()->LoadIcon(IDR_MAINFRAME);
#endif
}

CSample_DDlg::~CSample_DDlg(){
#ifdef MULTI_LANG
	FontManager(0,"");
#endif
}
```

```
void CSample_DDlg::DoDataExchange(CDataExchange* pDX)
{
	CDialog::DoDataExchange(pDX);
	//{{AFX_DATA_MAP(CSample_DDlg)
	DDX_Text(pDX, IDC_ED_GROUPNAME, m_GroupName);
	DDV_MaxChars(pDX, m_GroupName, 80);
	DDX_Text(pDX, IDC_ED_PARAM1, m_Param1);
	DDX_Text(pDX, IDC_ED_PARAM2, m_Param2);
	DDX_Text(pDX, IDC_ED_PARAM3, m_Param3);
	//}}AFX_DATA_MAP
}

BEGIN_MESSAGE_MAP(CSample_DDlg, CDialog)
	//{{AFX_MSG_MAP(CSample_DDlg)
	ON_WM_SYSCOMMAND()
	ON_WM_PAINT()
	ON_WM_QUERYDRAGICON()
	ON_BN_CLICKED(IDHELP, OnHelp)
	ON_WM_CREATE()
	//}}AFX_MSG_MAP
END_MESSAGE_MAP()
/////////////////////////////////////////////////////////////////////////////

// CSample_DDlg メッセージハンドラ
BOOL CSample_DDlg::OnInitDialog()
{
	CDialog::OnInitDialog();
	// "バージョン情報..." メニュー項目をシステムメニューへ追加します。
	// IDM_ABOUTBOX はコマンドメニューの範囲でなければなりません。
	ASSERT((IDM_ABOUTBOX & 0xFFF0) == IDM_ABOUTBOX);
	ASSERT(IDM_ABOUTBOX < 0xF000);

	CMenu* pSysMenu = GetSystemMenu(FALSE);
	CString strAboutMenu;
	strAboutMenu.LoadString(IDS_ABOUTBOX);
	if (!strAboutMenu.IsEmpty()){
		pSysMenu->AppendMenu(MF_SEPARATOR);
		pSysMenu->AppendMenu(MF_STRING, IDM_ABOUTBOX, strAboutMenu);
	}

	SetIcon(m_hIcon, TRUE);			// 大きいアイコンを設定
	SetIcon(m_hIcon, FALSE);		// 小さいアイコンを設定

	// TODO: 特別な初期化を行う時はこの場所に追加してください。
#ifdef MULTI_LANG
	if(IsKanjiDlg("sample")){
		MultiLangDialogConv("IDD_SAMPLE_D_DIALOG", (CDialog*)this);
	}else{
		MultiLangDialogConv("IDD_SAMPLE_D_DIALOG_0", (CDialog*)this);
	}
#endif

	return TRUE;  // TRUE を返すとコントロールに設定したフォーカスは失われません。
}

void CSample_DDlg::OnSysCommand(UINT nID, LPARAM lParam)
{
	if ((nID & 0xFFF0) == IDM_ABOUTBOX){
		CAboutDlg dlgAbout;
		dlgAbout.DoModal();
	}else{
		CDialog::OnSysCommand(nID, lParam);
	}
```

```
}

void CSample_DDlg::OnPaint()
{
	if (IsIconic()){
		CPaintDC dc(this); // 描画用のデバイスコンテキスト
		SendMessage(WM_ICONERASEBKGND, (WPARAM) dc.GetSafeHdc(), 0);
		// クライアントの矩形領域内の中央
		int cxIcon = GetSystemMetrics(SM_CXICON);
		int cyIcon = GetSystemMetrics(SM_CYICON);
		CRect rect;
		GetClientRect(&rect);
		int x = (rect.Width() - cxIcon + 1) / 2;
		int y = (rect.Height() - cyIcon + 1) / 2;
		// アイコンを描画します。
		dc.DrawIcon(x, y, m_hIcon);
	}else{
		CDialog::OnPaint();
	}
}

HCURSOR CSample_DDlg::OnQueryDragIcon()
{
	return (HCURSOR) m_hIcon;
}

void CSample_DDlg::OnHelp()
{
#ifdef MULTI_LANG
	Bantu();
#else
	//980612 DR.H.K.
	MessageBox("これは、ユーザー定義のパラメトリック部品を増築するための雛型です。・・・");
#endif
}

void CSample_DDlg::OnCancel()
{
	// TODO: この位置に特別な後処理を追加してください。
	CDialog::OnCancel();
#ifdef MULTI_LANG
	MultiLangEnd();
#endif
}

void CSample_DDlg::OnOK()
{
	//980612 DR.H.K.
	FILE *wp;
	fopen_s(&wp,filename,"wt");
	if(!wp){
#ifdef MULTI_LANG
		MessageBox( Kanji( "E01" ) );
#else
		MessageBox("中間ファイルが作成できませんでした");
#endif
		exit(2);
	}
	UpdateData(TRUE);
	fprintf(wp,"%s=FILE(SAMPLE,%0.3lf,%0.3lf,%0.3lf);¥n",m_GroupName,m_Param1,m_Param2,m_Param3);
	fclose(wp);
	m_GroupName.ReleaseBuffer();
	CDialog::OnOK();
```

```
#ifdef MULTI_LANG
		MultiLangEnd();
#endif
}

int CSample_DDlg::OnCreate(LPCREATESTRUCT lpCreateStruct)
{
		if (CDialog::OnCreate(lpCreateStruct) == -1)
			return -1;

		//980613 DR.H.K. 以下、サンプルコードです。
		m_Param1 = 9.0; //桁行初期値
		m_Param2 = 5.0; //梁間初期値
		m_Param3 = 0.6; //勾配初期値
		m_GroupName.Format("%s",groupname); //グループ名称
		{ //前回の会話部の結果を再利用したい場合には、次の処理を追加して下さい
			FILE *rp;
			char command[80],*p;
#if 1 //旧来の形式
			if(!(rp = fopen(filename,"rt"))) return 3;
			if(! (fscanf(rp,"%s",command) << fclose(rp)) ) return 4;
			if(!(p = strstr(command,"="))) 	return 5;
			*p = 0;
			if(1!=sscanf(command,"%s",m_GroupName)) return 6;
			if(3!=sscanf(p+1,"FILE(SAMPLE,%lf,%lf,%lf);",&m_Param1,&m_Param2,&m_Param3)) return 7;
#else //厳格化された開発環境への適応
			if((fopen_s(&rp,filename,"rt"))) return 3;//fopen_s正常終了するとを返すように修正
			if(! (fscanf_s(rp,"%s",command,79) << fclose(rp)) ) return 4;
			//fscanf_s の場合、入力配列の次に最大文字数を指定する
			if(!(p = strstr(command,"="))) 	return 5;
			*p = 0;
			if(1!=sscanf_s(command,"%s",m_GroupName,79)) return 6;//
			if(3!=sscanf_s(p+1,"FILE(SAMPLE,%lf,%lf,%lf);",&m_Param1,&m_Param2,&m_Param3))
				return 7;
#endif
		}
		return 0;
}

BOOL CSample_DDlg::Create(LPCTSTR lpszTemplateName, CWnd* pParentWnd)
{
#ifdef MULTI_LANG
		if(IsKanjiDlg("sample"))
			return CDialog::Create("IDD_SAMPLE_D_DIALOG", pParentWnd);
		else
			return CDialog::Create("IDD_SAMPLE_D_DIALOG_0", pParentWnd);
#else
		return CDialog::Create(lpszTemplateName, pParentWnd);
#endif
}
```

５－９．初期の外部関数の内部処理

以前のバージョンにおける原始図形、基本構成要素において、外部関数が使用される場合、ダイアログは基幹部分の一部として組み込まれ、特別な処理が行われていた。過去に関する参考情報であるが、概要は以下の通りである。

（１）原始図形(cube, sphere, cylinder, flatcylinder, cone, flatcone)

それぞれに、ダイアログ部が基幹部分のクラスとしてコーディングされ、CMainFrame

の中で、メニュー選択に対するコールバック関数が定義されていた。

各コールバック関数においては、原始図形の各パラメータ（例えば球 Sphere であれば、半径と中心の xyz 座標）がセットされ、OK ボタンが押されると、共通の genOkCB 関数（genshi.c）を呼び出す。ここから、CallExtCommand 関数(dataope.c)を呼び出し、更にそこから、原始図形の種類で場合分けして、i3CallXXX 関数(I3 ライブラリ、i3ipcall.c)を呼び出す。それぞれの関数は、共通 FORMAT のパラメータ・リストから、図形毎のパラメータを復元し、LSS-G 形式のコマンドを組み立てて、IP_interpret(コマンド文字列)を呼び出す。

以上を要約すると、関数の種類とパラメータ・リストを一度共通 FORMAT にコーディングした上で、再度デコードする、という手間をかけた、４階層の処理が行われている。中間階層にある上から二番目の genOkCB は、関数名とパラメータをコーディングしている。CallExtCommand 関数は、関数名とパラメータを再びデコードしている。この間に、意味のある付加価値はない。即ち、各ダイアログの OK 終了から、直接 IP_Interpret を起動しても結果は同様であり、時間とスタック・メモリを浪費しているだけである。この構成に意味があるとすれば、外部関数の呼び出し数等に関する統計を取るとか、頻度の高い関数に関して形状生成部を外部関数とせずに直接メモリ上に形状を生成するように作り直して、処理速度を向上させるような場合であろう。そのような場合であっても、その作り込みの場所は、IP_Interpret よりも下の階層になる（さもないと、LSS-G ファイルの中で参照された外部関数に関しては、直接生成することができない)。

以上のような理由によって、これらのオーバーヘッドとなっていた以前のバージョンにおける処理はカットした。

（２）基本図形：型鋼

ダイアログは、４種類の型鋼で共通となっており、選択したタイプ（H,T,C,L）を引数として、CallSteelCommand(dataope.c)を呼び出す。ここから、タイプ別に、i3CallXSteel 関数を起動する。タイプ別の i3CallXSteel 関数は、LSS-G コマンドの文字列

Gn=FILE(xSTEEL,a,b,c,d,h);　　(但し x は、H,T,C または L)

を構成して、IP_Interpret 関数を呼び出す。その際に、パラメータの妥当性チェックを行い、不適切であれば、メッセージを出すことなくリターンしている。このようなチェックは、ダイアログの中で実行されるべきである。なお、個々の関数におけるパラメータの組み立ては同様であるが、L 鋼だけは、x、y という二つの追加パラメータがある。

（３）掃引体

掃引体１面、掃引体２面のそれぞれのダイアログから、共通の genSweep(Genshi.c)を呼び出す。ここから、場合分けして i3CallSweepN 関数を呼び出す。そこで LSS-G 形式のコマンド文字列を作成し、IP_Interpret 関数を呼び出す。

その他の一般化された外部関数（ユーザー定義のパラメトリック部品）においては、ダイアログ部も外部関数化されている。従って、これらの古い原始図形や基本図形のパラメ

ータ部を、独立した実行形式として分離独立することにより、新しい一般的な外部関数と同様に扱うことができ、処理は簡潔となった。

なお、これらの原始図形を生成する関数は、ダイアログ部を基幹部分に内部化していたのに対して、形状生成部は当初から外部関数として分離していた。これは、前者が C++で書かれプラットフォームへの依存度が高いのに対して、後者が C で書かれ、プラットフォームに対する独立性が高い、というソースコード管理上の理由があった。しかし、処理速度の観点からは、寧ろ、ファイル読み込み等に際して頻繁に参照される原始図形について形状生成部を内部化する方が、システムの性能の観点からは効果があると考えられ、今後検討する価値がある。

５－１０．インタープリタにおける外部関数の起動とパラメータ・リスト

FILE 形式が LSS-G コマンドの中に検出された場合、これを解析するために、i3File 関数(I3iplssg.c)を起動する。ここから、スタティックな関数 l_callProc 関数を呼び出す。この関数は、引数がある場合に、そのタイプ（EXT_TAB において定義された型）に応じた処理を行う。

このタイプには、以下のものがある：

① 整数 (I3_EXT_INT)
② 倍精度実数 (I3_EXT_DOUBLE)
③ 文字列 (I3_EXT_STRING)
④ ファイル (I3_EXT_FILE)
⑤ 面 (I3_EXT_FACE)

面(d3Face 構造体)のアドレスを直接渡すことはできないので、一時的なファイルに出力した上で、ファイル名を渡す関数が用意されている（i3_outputGroupFace）が、実装されていない（TRUE を返すのみ）

⑥ 線 (I3_EXT_LINE)

線(d3Face 構造体)のアドレスを直接渡すことはできないので、一時的なファイルに出力した上で、ファイル名を渡す関数が用意されている（i3_outputGroupFace）が、実装されていない（TRUE を返すのみ）。

ファイル・ロードが終了し、インタープリタ空間がリセットされた後の、メモリ上のデータにおける面や線の構造体は、名称を持たないので、将来実装することがあるならば、生成されたグループから、参照する面をポインタで把握しておき、ファイル保存に際しては、明示的に関連づけられるような名称を新たに付すといった処理を工夫する必要がある。

⑦ グループ (I3_EXT_GROUP)

メモリ上にあるグループ(d3Group 構造体)のアドレスを外部関数に直接渡すことはできないので、一時的なファイルに出力した上で、ファイル名を渡す関数が用意されている（i3_outputGroup）。これを用いた関数の例はまだない。

⑧　時刻（I3_EXT_TIME）

システムの現在時刻を取得し、引数として渡す。システムの時刻が変更された場合には、時刻の引数をもつパラメトリック部品が一度削除され、再生成される。

５－１１．既存外部関数のダイアログ部各説

以前のバージョンにおいては、原始図形と、ユーザーが定義した部品の間で、上記のように形状生成部において共通の処理は行われていたものの、パラメータ設定部は後者のみが別関数として処理されていた。Ver.2.09 においては、これらを全て同一の処理で行う方法によりシステムの単純化を行った。これにより、旧来の原始図形に関しても、一度形状を生成した後に、パラメータを再編集することができるようになった。但し、操作環境の互換性を保つために、旧来の原始図形のダイアログ部は、メニューの[形状生成][原始図形]以下の各メニューから起動できるようにすると共に、[形状生成][オプション]で一覧表示するユーザー定義のパラメトリック部品に関しては、原始図形を除いたもののみを表示している。

なお、以下の解説中、旧版のリソース ID の名称に一部混乱が存在していたが、新版においては修正を行った。このような旧版のリソース名称に関しては、誤記ではないことを明示するために、(ママ)と付記した。例えば、（３）円柱のパラメータ設定ダイアログは、旧版では、IDD_DLG_CONE(ママ)という、円錐を想起させるリソース ID の名称であった。

（１）直方体　CUBE

南西下隅の座標と、幅・奥行き・高さをパラメータとして立方体を生成する。大きさの各項目に負の値が入力された場合でも、外側が表となるように面を構築する。

メイン画面をオルソ系（平面、立面等）とし、画面クリックを行うことで、座標を取り込む。配置座標は、クリックした位置、また幅、奥行き、高さに関しては配置座標との差分が代入される。

幅、奥行き、高さのいずれかにゼロを指定した場合には、平面図形を出力する。また、この内２項目がゼロであった場合には、線を出力する。

全てゼロであった場合には、表示する実体（面や線）の無いグループのみを出力する。

図５－１：直方体のパラメータ設定画面

旧版：リソース：IDD_DLG_PRIM_CUBE ハンドラ：CPrimCubeDlg(primcube.cpp)
新版：リソース；IDD_CUBE　ハンドラ：CCubeDDlg(Cube_DDlg.cpp)

（2）球

配置座標を中心する、半径 R の球を生成する。

パラメータ名称の左にあるチェックボックスにチェックを入れ、メイン画面をオルソ系（平面、立面等）とし、画面クリックを行うことで、座標を取り込む。配置座標は、クリックした位置、半径に関しては配置座標との差分が代入される。

表示にあたって球面を多面体で近似表現する際の分割数は、環境設定ファイル kdbms.set の SPHERE エントリで指定したパラメータを用いる（デフォルト値緯度 16 分割、経度 20 分割）。この値は、外部ファイルには記述されない。この情報は、形状生成関数(この場合 sphere.exe)をインタープリタが起動する際に、追加の引数として渡される。

半径として負値を指定した場合には、内側が表となる球を生成する。この場合、体積を計測すると負値となる。

半径としてゼロを指定した場合には、データが生成されるが、表示不能である。そのような場合、報告書機能において確認することができる。

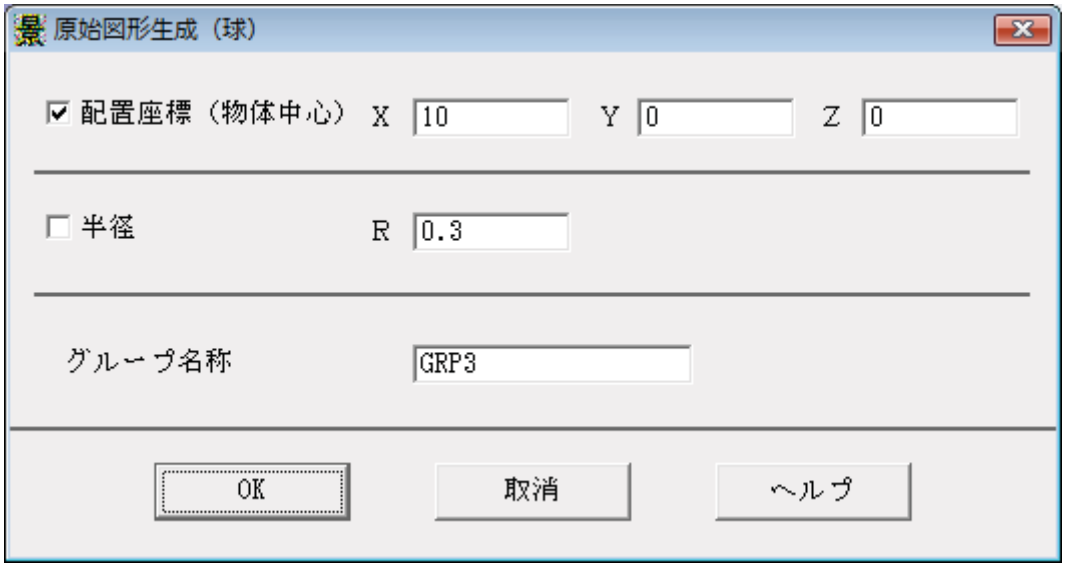


図５－２：球のパラメータ設定画面

旧版　リソース：IDD_DLG_PRIM_SPHERE ハンドラ：CPrimSphereDlg(primsphe.cpp)
新版　リソース：IDD_SPHERE　ハンドラ：CSphere_DDlg(sphere_DDlg.cpp)

（3）円柱

上中心と下中心を結ぶ線分を軸とし、半径を R とする円を断面とする円柱を出力する。円柱の形状は角錐による近似とし、頂点数は、環境設定ファイル kdbms.set の SEGS エントリーに設定された値とする。この値は、基幹部分から外部関数を起動する際の追加の引

数として渡される。円柱の側面を構成する各面には、頂点毎の法線ベクトルを定義し、陰影が連続するように表示する。

上中心と下中心が一致する場合には、XY 平面と平行（水平）な円（平面図形）を出力する。

メイン画面をオルソ系（平面、立面等）とし、画面クリックを行うことで、座標を取り込む。配置座標は、クリックした位置、半径に関しては配置座標との差分が代入される。

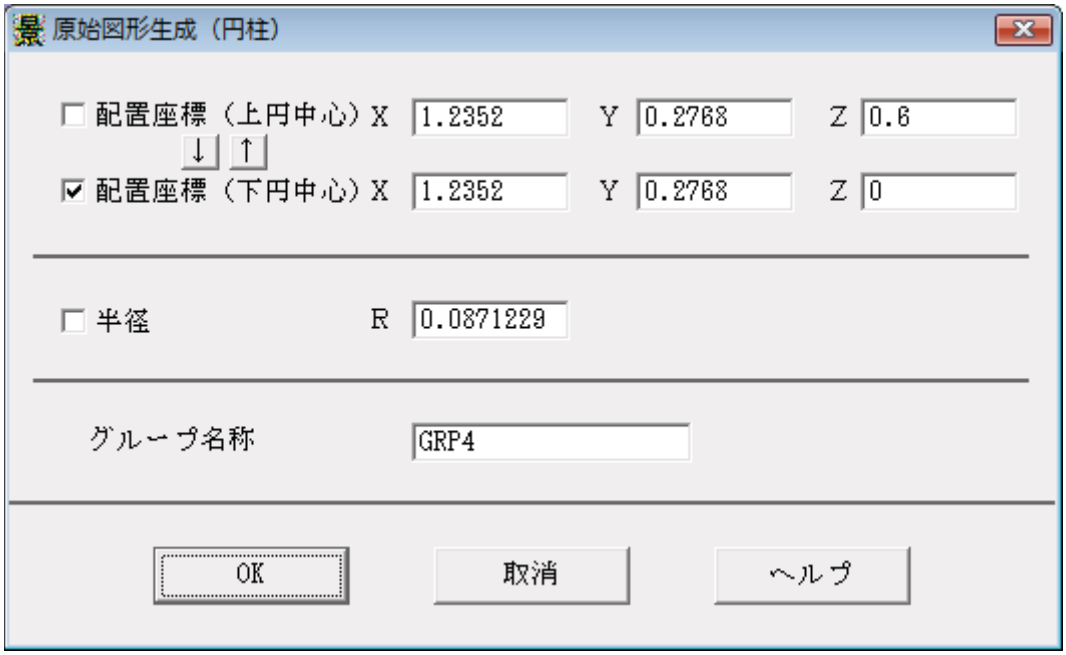


図５－３：円柱のパラメータ設定画面

旧版リソース：IDD_DLG_PRIM_CONE(ママ) ハンドラ：CPrimCylinderDlg(primcyli.cpp)
新版リソース：IDD_CYLINDER ハンドラ：CCylind_DDlg(Cylind_DDlg.cpp)

（４）円錐・円錐台

上面中心と下面中心を軸とし、それぞれの半径を R1、R2 とする、円錐または円錐台を出力する。

上面中心と下面中心が一致する場合には、XY 平面と平行（水平）な円を生成する。この時 R1 と R2 が一致する場合には、円を、また異なる場合には、ドーナツ形を生成する。

メイン画面をオルソ系（平面、立面等）とし、画面クリックを行うことで、座標を取り込む。配置座標は、クリックした位置、また半径に関しては配置座標との差分が代入される。

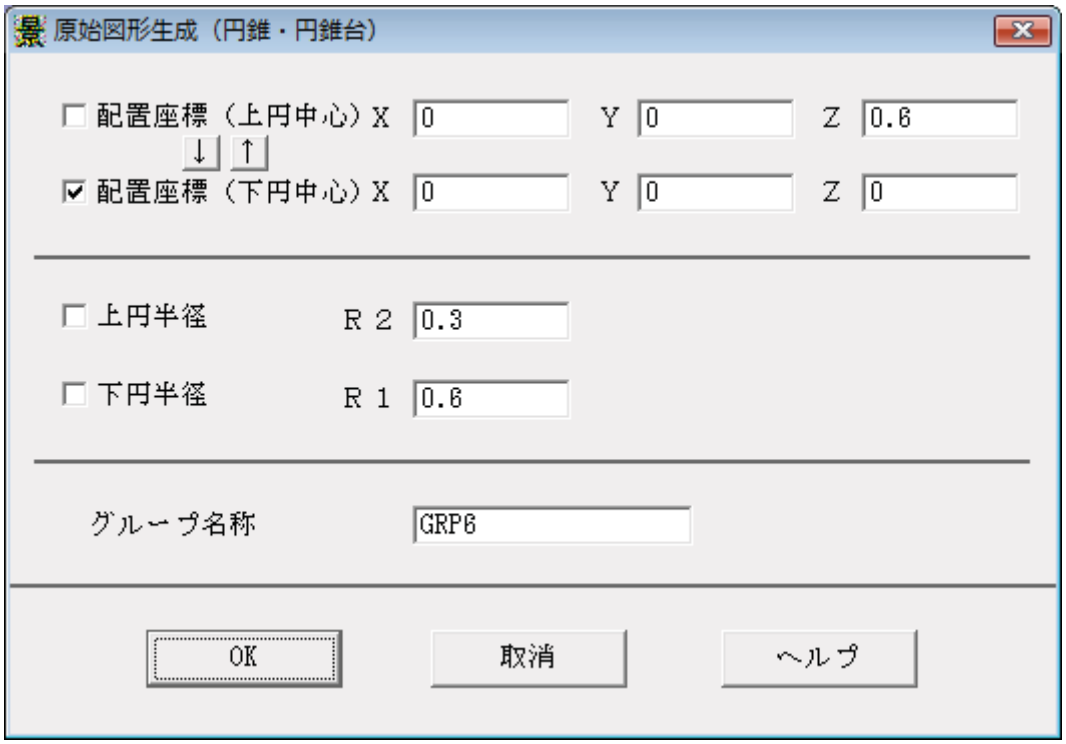


図５－４：円錐・円錐台のパラメータ設定画面

旧版　リソース：IDD_DLG_PRIM_FLATCYLI(ママ)

　　ハンドラ：CPrimFlatconeDlg(primflco.cpp)

新版　リソース：IDD_FLATCYLI　ハンドラ：CFlatCyli_DDlg(FlatCyli_DDlg.cpp)

（５）角柱

上中心と下中心を結ぶ線分を軸とし、軸から各頂点までの長さをＬとする、頂点数Ｎの正多角形を断面とする角柱を出力する。

上中心と下中心が一致する場合には、XY 平面と平行（水平）な、正多角形（平面図形）を出力する。

頂点数Ｎが２以下の場合には、エラーとする。

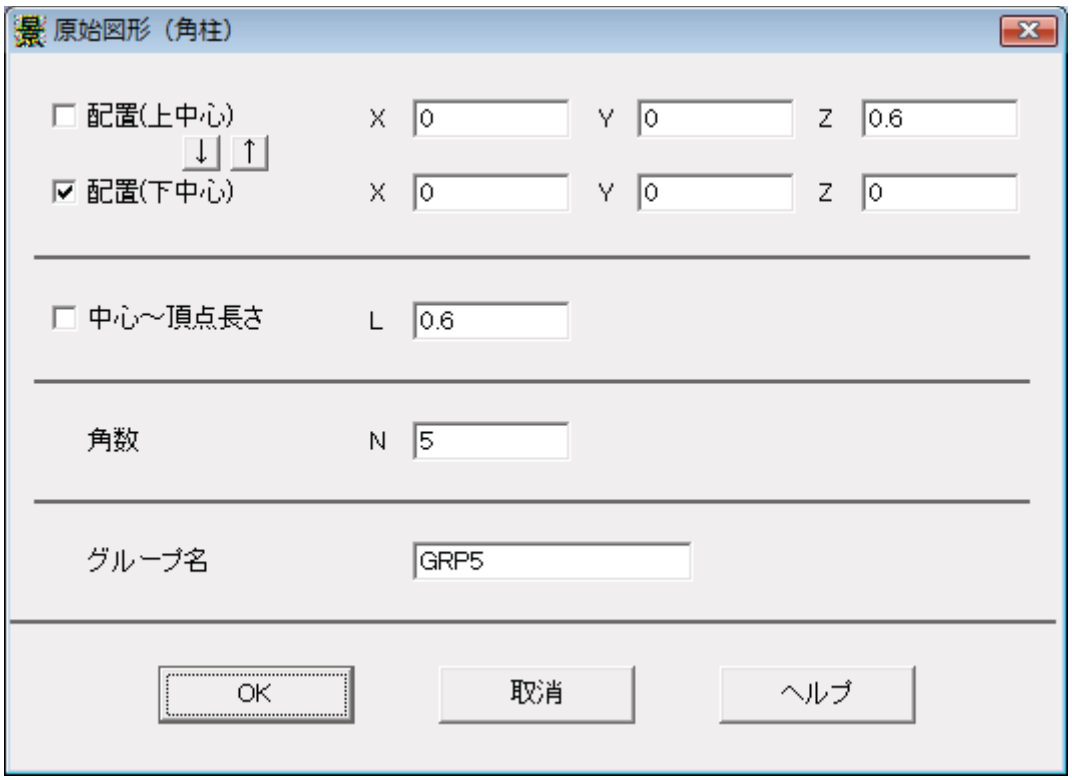


図５－５：角柱のパラメータ設定画面

旧版リソース：IDD_DLG_PRIM_CYLI(ママ)

ハンドラ：CPrimFlatCylinderDlg(primflcy.cpp)

新版リソース：IDD_FLATCYLI ハンドラ：CFlatCyli_DDlg(FlatCyli_DDlg.cpp)

（６）角錐・角錐台

上面中心と下面中心を軸とし、軸からそれぞれの頂点までの長さを L1,L2 とする、頂点数 N の角錐または角錐台を出力する。

上面中心と下面中心が一致する場合には、XY 平面と平行（水平）な平面図形を生成する。この時 L1 と L2 が一致する場合には、正多角形を、また異なる場合には、穴あき図形を生成する。

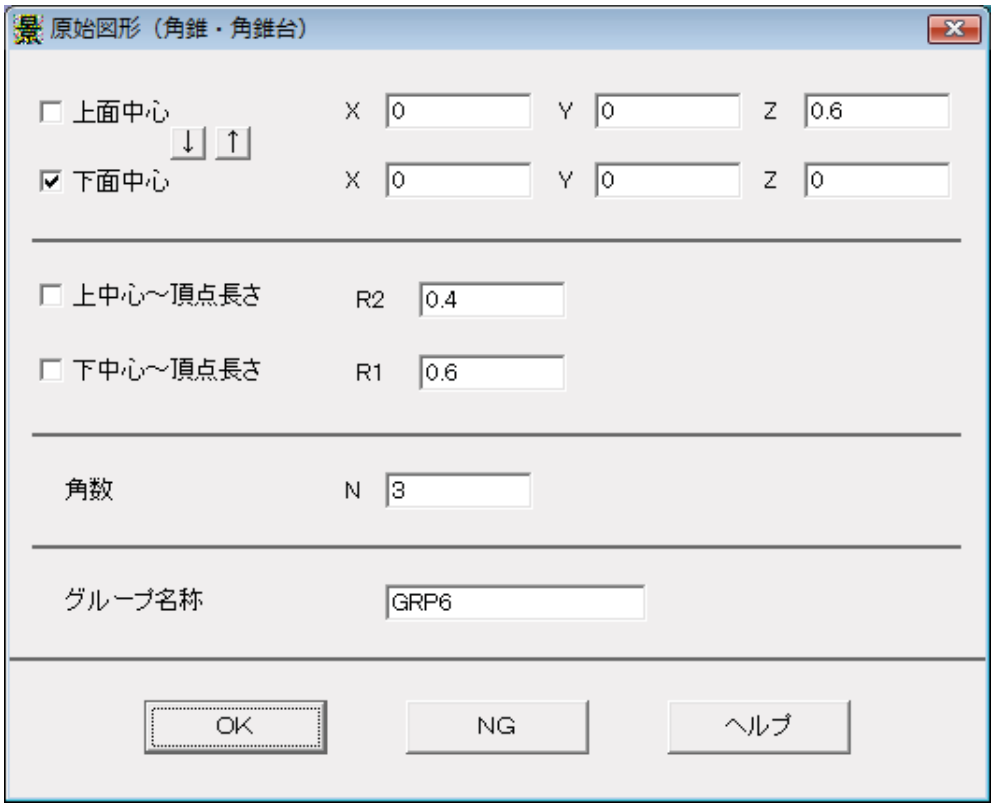


図５－６：角錐・角錐台のパラメータ設定画面

旧版　リソース：IDD_DLG_PRIM_FLATCYLI(ママ)

　　ハンドラ：CPrimFlatConeDlg(primflco.cpp)

新版　リソース：IDD_FLATCONE　ハンドラ：CFlatcone_DDlg(Flatcone_DDlg.cpp)

（７）掃引体１面

断面を定義する LSS-G 形式のファイルと、軌跡を定義する LSS-G 形式のファイルを引数とし、後者に従って、前者の図形を平行移動した時に生成する立体図形を出力する。

断面については､選択した LSS-G ファイルに含まれる最初のグループの最初の面を用い、また軌跡については、選択した LSS-G ファイルに含まれる最初のグループの最初の線の最初の線分を用いる。

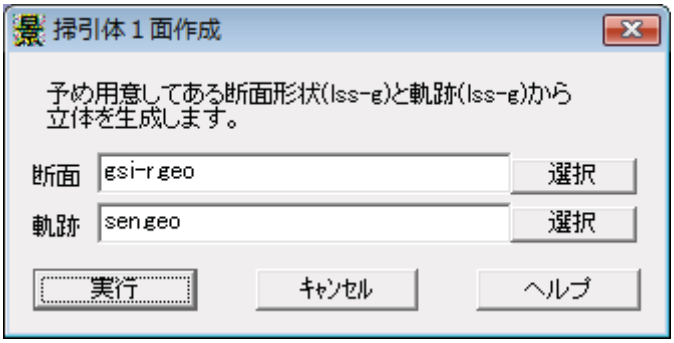


図５－７：掃引体１面のパラメータ設定画面

旧版　リソース：IDD_SWEEP1 ハンドラ：CSweep1(sweep1.cpp)
新版　リソース：IDD_SWEEP1 ハンドラ：CSweep1_DDlg(sweep1_DDlg.cpp)

（８）掃引体２面

断面を定義する二つの LSS-G ファイルを引数とし、二つの断面を端面として間を充填した立体図形を出力する。それぞれのファイルの最初のグループの最初の面を形状生成に使用する。この時、面の頂点数が異なる場合には、少ない方と同じ数の頂点を多い方の面から採り、残りを棄却する。

図５－８：掃引体２面のパラメータ設定画面

旧版　リソース：IDD_SWEEP2 ハンドラ：CSweep2(sweep2.cpp)
新版　リソース：IDD_SWEEP2 ハンドラ：CSweep2_DDlg(sweep2_DDlg.cpp)

（９）切妻屋根（パラメトリックな部品の雛型として）

桁行、梁間、勾配から、切妻屋根の形状を生成する。
パラメータ１：桁行　パラメータ２：梁間　パラメータ３：勾配

図５－９：切妻屋根のパラメータ設定画面

リソース：IDD_SAMPLE_D_DIALOG　ハンドラ：CSample_DDlg (sample_DDlg.cpp)

（１０）文字列

文字列を引数として、形状を生成する。

図５－１０：文字列のパラメータ設定画面

リソース：IDD_SAMPLE_D_DIALOG　ハンドラ：CSample_DDlg(string_DDlg.cpp)

（１１）階段

幅、蹴上、踏面、段数をパラメータとして、階段を生成する。

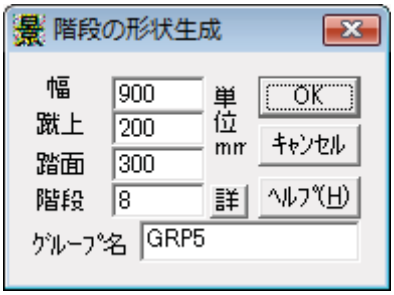


図５－１１：階段のパラメータ設定画面

リソース：IDD_STAIR_D_DIALOG　ハンドラ：CStair_DDlg (stair_DDlg.cpp)

（１２）ダイアログ部が未実装な外部関数への仮の共通ダイアログ

任意の外部関数を対象として、通常のダイアログ部が生成し、仮のファイルとして出力するコマンドを、手入力する。ユーザーが選択した外部関数に対応するダイアログ部が存在しない場合に、代替的な入力手段として起動される。その場合、ext;.tab に登録された引数の形を文字列として表示する。

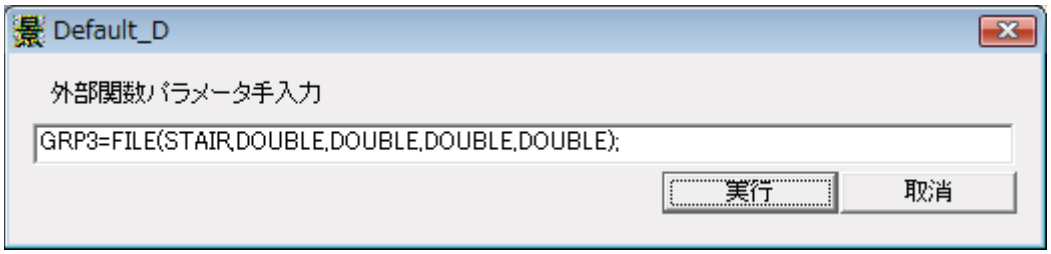


図５－１２：ダイアログ部が未実装な外部関数の共通パラメータ設定画面

リソース：IDD_DEFAULT　ハンドラ：CDefault_DDlg(default_DDlg.cpp)

（１３）URL アクセス

WEB 上にある LSS-G ファイルを取得して取り込む。

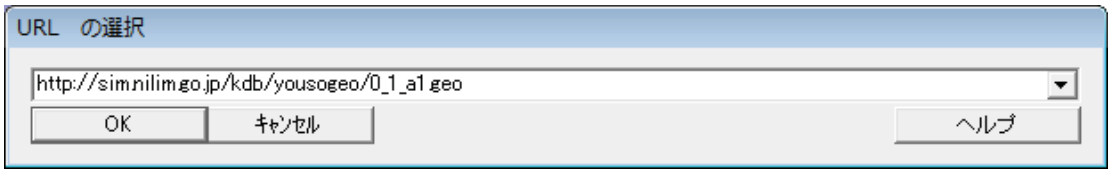


図５－１３：URL アクセスのパラメータ設定画面

リソース：IDD_URL_D_DIALOG　　ハンドラ：CURL_DDlg(URL_DDlg.cpp)

（１４）箱ビル正面テクスチャ付き

間口、奥行き、高さ、及びテクスチャファイル名を引数として、正面（ファサード）にテクスチャを適用した直方体を生成する。

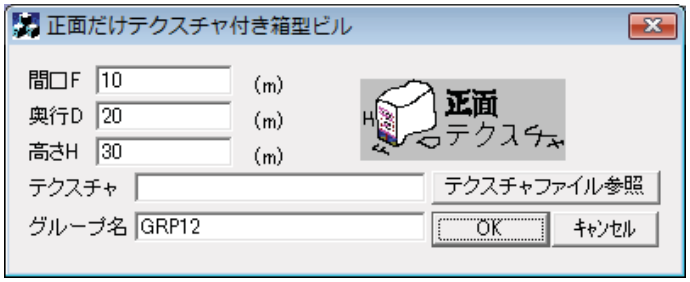


図５－１４：箱ビルのパラメータ設定画面

リソース：(紛失)　　ハンドラ：(紛失)

（１５）正多面体

多面体の種類、及び辺の長さを引数として、正多面体を生成する。

図５－１５：正多面体のパラメータ設定画面

リソース : IDD_TAMENTAI_D_DIALO ハンドラ : CTamentai_DDlg(tamentai_DDlg.cpp)

（１６）VRML2LSS

VRML ファイル(*.wrl)を変換する。パラメータは、ファイル名のみであるため、直接ファイル選択ダイアログを開く。

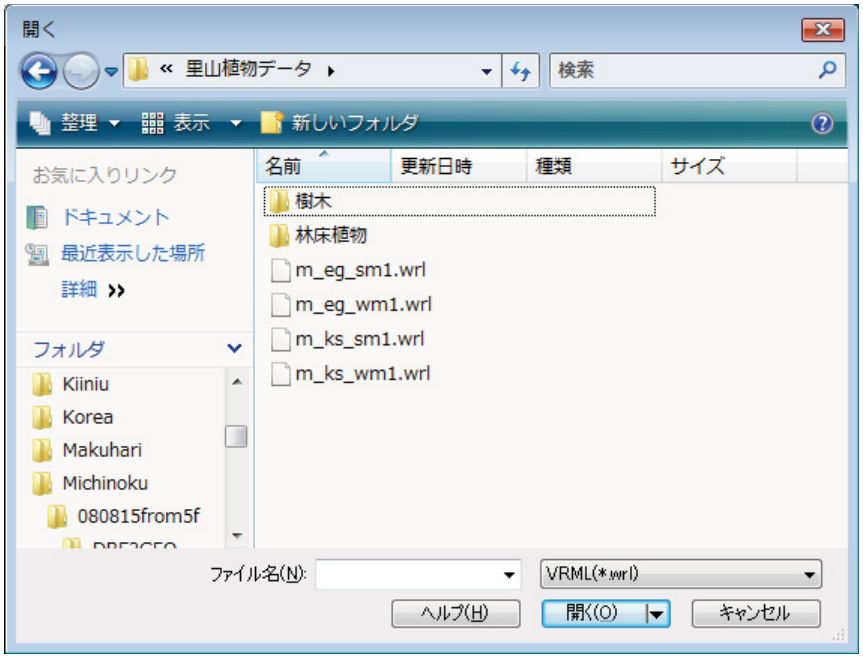


図５－１６：VRML ファイル名選択画面

リソース：なし　ハンドラ：GetOpenFileName()関数(bahasa_D.cpp)

（１７）SEGITIGA

三辺の長さから三角形（平面図形）を生成する。

図５－１７：三角形を定義する各辺の長さ設定画面

リソース：IDD_SEGITIGA_D_DLG　ハンドラ：CSegitiga_DDlg(segitiga_DDlg.cpp)

（１８）SCADEC2

電子納品ファイル(SXF, .p21 形式)を読み込む。参照ボタンによりファイルを選択すると、そのファイルに含まれるレイヤーの一覧を表示する。必要とするレイヤーを選択した上で、ＯＫボタンを押すと、そのレイヤーが変換されてメイン画面に表示される。

現段階ではＳＸＦ形式は二次元までにしか対応していないため、高さのない、ＸＹ平面

上の図形として変換している。メイン画面において、メニューの[表示][平面図]で平面図表示に設定することにより、図面として表示する。これを下図として立体を生成する。

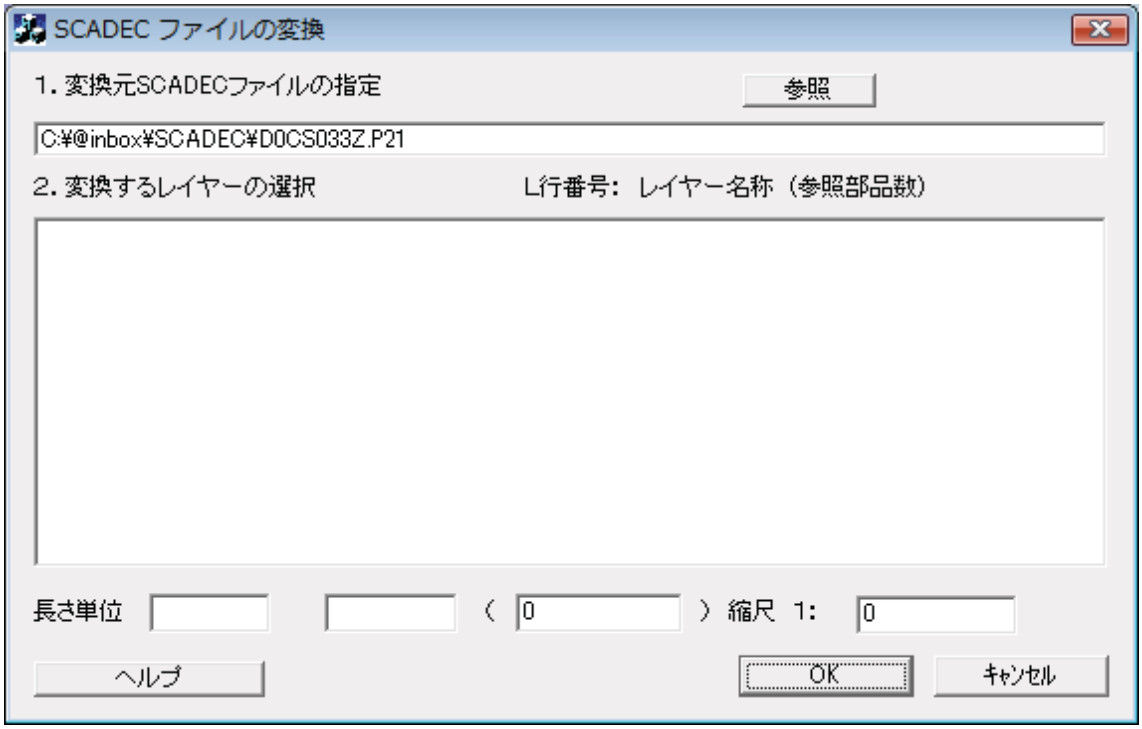


図５－１８：電子納品データ変換のパラメータ設定画面

リソース:IDD_SCADEC_D_DIALOG　ハンドラ:CSCADEC_DDlg(SCADEC_DDlg.cpp)

（１９）STEEL

型鋼５種類から一つを選択し、パラメータを設定して形状生成する。実際の形状生成は、以下の４の外部関数（①HSTEEL.exe、②CSTEEL.exe、③LSTEEL.exe、④TSTEEL.exe）により行っており、ダイアログ部は一つで共用している。

なお、既存パラメトリック部品の再編集において、通常は、部品名_D.exe というダイアログを開くが、型鋼の場合には、Hsteel_D.exe などは存在していない。そこで、このような場合には、メイン側で、Hsteel_D.ijk というテキスト・ファイルを調べ、その中に記述されているダイアログ部の実際の実行形式「steel_D.exe」名称を取得し、これを開いている。この方法により、４本の小さなテキスト・ファイルを用意することで、再編集の一貫性を確保した。

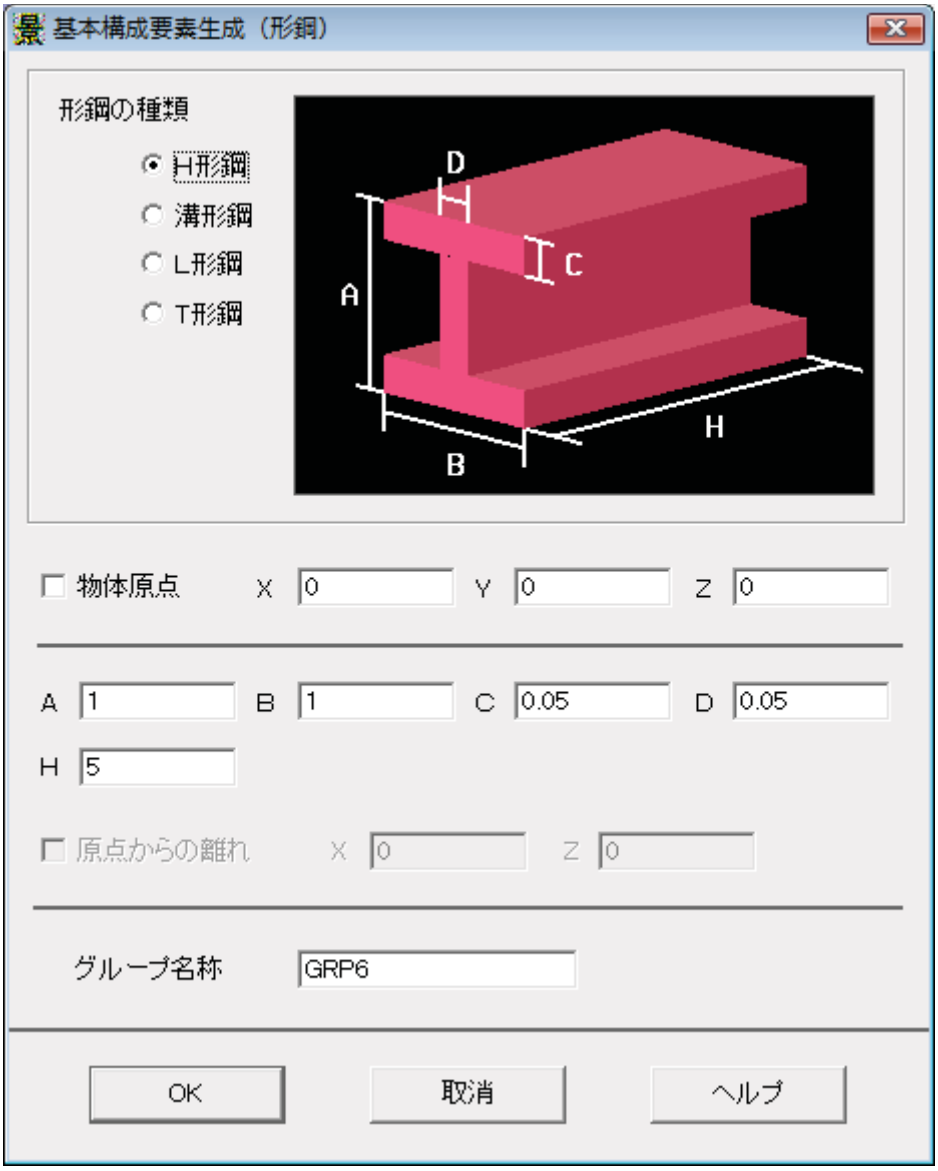


図５－１９：型鋼のパラメータ設定画面

リソース：IDD_STEEL ハンドラ：CSteel_DDlg(Steel_DDlg.cpp)

なお、研究開発の過程で試作した、CreateProcess により steel_D.exe を CreateProcess により起動する方法を Tsteel_D.exe として残している。これをコピーし、各外部関数_D.exe という名称に Rename することによっても、外見上同等の操作環境を得ることができる。

‘

（２０）PERIOD

各構成要素に対して、生成時点と消滅時点を定義する。景観シミュレーション・システムにおいては、時間は、地物固有の属性ではなく、シーンを構成するパラメータとして、光源や視点位置などと同列の、状況に応じて変化する条件として定義している。しかしながら、地物において、時間の関数として変化する属性を定義することができるように、外

部関数には、現在の時間を引数として渡すことができる。従って、これを受けて、時間に応じて変化・変形するようなオブジェクトを定義することができる。

PERIOD はその最も単純な例で、建設時点と除却時点、及び対象となるオブジェクトを引数として形状を生成する。シーンに定義された時間（日数で定義する）が建設時点と除却時点の間にある場合にのみ、ファイルで指定されたオブジェクトを実体として扱う。

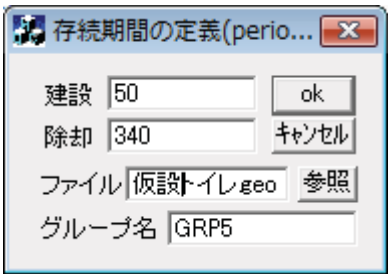


図５－２０：存続期間の設定画面

リソース：IDD_PERIOD_D_DIALOG ハンドラ：CPeriod_DDlg (period_DDlg.cpp)

（２１）トーラス

原点を中心とする水平のドーナツ形を生成する。断面半径と軌道半径を指定する。軌道半径の方が大きな値でなければならない。さもないと、自己交差する図形となる。

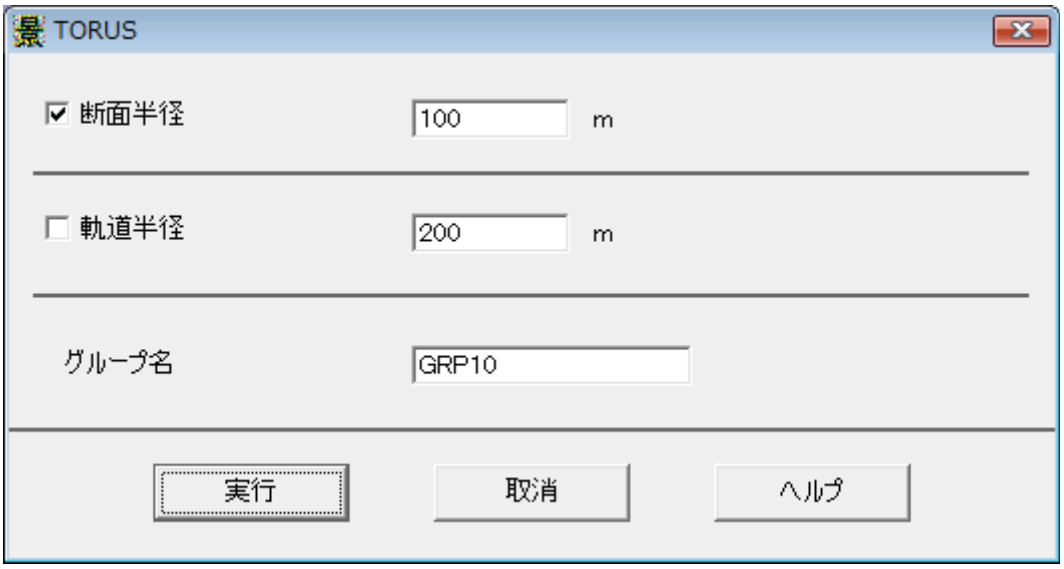


図５－２１：トーラスのパラメータ設定画面

リソース：IDD_TORUS　ハンドラ：CTorus_DDlg(torus_DDlg.cpp)

（２２）その他のファイルコンバータ

コンバータであって、ファイル選択ダイアログを開くだけの（１６）とほぼ同様のダイアログとして、BS2LSS, FIRE2LSS がある。

６．プラグインDLL

６－１．概要

景観シミュレータにはいくつかの枝分かれしたバージョンが存在する[1)]。これらの機能を統合し、全ての機能を使用することができる統合バージョンを作ると、不必要に大きなシステムとなることから、以下の方針を採った。

（１）多くの利用分野に共通する一般的な機能を有する、完成度の高い基幹部分を作成する。この基幹部分には、選択的に利用する専門的な機能を、プラグインとして必要となった時点でメニュー一覧から選択し、動的に追加できるインターフェースを用意する。

（２）枝分かれバージョンのそれぞれが有する、専門性の高い機能を、プラグインDLLとして分離し、必要となった時点で基幹部分に付加できる形とする。

（３）プラグインDLLからは、基幹部分における編集対象となっている全てのデータ（最も重要なものは、ジオメトリのルートなど）へのアクセスを可能とする。

（４）プラグインDLLのコーディングのために、基幹部分が有するスタティックなライブラリ関数の使用が可能とする。

（５）プラグインDLLは、ユーザーが終了を選択し、処理が終了した時点でアンロードする（占有していたメモリを解放する)。

（６）基幹部分の側から、開いているプラグインを閉じるためのプロトコルを用意する。

６－２．基幹部分のライブラリ関数の提供

基幹部分がビルドにおいてリンクしている、スタティック・リンク・ライブラリの諸関数を、DLL側にエクスポートされる関数として再定義した。このために、各ライブラリのヘッダー(d3dml.h, g3drl.h 等)をそれぞれ二つのヘッダー・ファイルに分割し、関数の宣言を行っている部分を、d3symbol.h, g3symbol.h 等の名称の別ファイルに分離した上で、関数宣言についてコンパイル・スイッチにより、DLL側に対してエクスポートされるように宣言している。この宣言は、スタティック・リンク・ライブラリ（dml.lib, drl.lib 等）のビルド時、およびこのライブラリへのスタティック・リンクを含む基幹部分(sim.exe)のビルド時に、二度使用する。基幹部分が、_declspec(dllexport) 宣言によりエクスポートされる関数を有する結果、sim.exe のビルドに際して、sim.lib というライブラリが生成される。この中に、エクスポートされる（即ちプラグインDLLの側から利用することができる）関数に関する情報が含まれている。従って、プラグインDLLをビルドする際に、スタティック・リンク・ライブラリをリンクする必要はない。また仮にそのようなビルドを行うと、ライブラリ中のスタティック変数やグローバル変数の動作が問題となる危険性がある。

なお、注意しなければならない点は、関数のDLLエクスポートを前提としていなかった過去のバージョンのビルドにおいては、ソースコードによっては、当該ソースコードで定

義された関数を宣言するヘッダー・ファイルを冒頭でインクルードしていないものがあった、という点である。ビルドを通すだけのためには、自分自身を宣言する必要は無いが、それらの関数を DLL エクスポートするためには、宣言が必要である。従って、エクスポート関数を含むソースには冒頭部分でエクスポートを宣言するヘッダーをインクルードしておく必要がある。この点が原因となるリンク・エラーは解決に時間を要する場合がある。

リスト６－１：DLL エクスポートするライブラリ関数のヘッダー例

[例]d3dml.h の関数宣言の部分を、d3symbol.h として分離した（末尾部分）

```
・・・・(前略)・・・・
};
struct _d3PickPath {
/*private:*/
/*public:*/
        d3TravInfo *trav;
        int num;
};

#include "d3symbol.h" /*050418従来は全てこのヘッダー内容を直接個別に宣言していた*/
#ifdef __cplusplus
}
#endif
#endif  /* !__D3DML_H__ */
```

[例]d3symbol.h における DLL エクスポートの関数宣言例

```
#ifdef SIMDLL
#define a __declspec(dllexport)
#else
#define a
#endif

/* functions in d3dml.c プリプロセッサでSIMDLLが定義されていれば、DLLエクスポートされる関数として宣言される。そうでなければ、従来通り*/
a double inpo(double A[], double B[]);
a void expo(double A[], double B[], double H[]);
a int ketemu(double A0[], double A1[], double B0[], double B1[]);
a int d3CalcBoundingBox(d3Group *g);
/* 注）別の目的でシンボル「a」を用いている箇所はコンパイル・エラーを起こすので修正する必要がある。例：
NG: void SetColor(float r, float g, float b, float a);
OK: void SetColor(float r, float g, float b, float aa);  */
```

一方、同じヘッダーを DLL 側で利用する際には、SIMDLL というプリプロセッサを定義しないでコンパイルを行い、シンボル部分を通常の関数として宣言すると共に、リンク・ライブラリに sim.lib を加えることにより、プラグイン DLL が起動した時点で sim.exe に含まれるこれらの関数がダイナミックにリンクされ、それらを使用することができる。

プラグイン DLL は、独立性の高い外部関数とは異なり、起動後は基幹部分の一部を構成

するダイアログと同等に基幹部分の側のデータにアクセスし、削除・修正を含む様々の処理を行うことができる。この処理のために基幹部分のライブラリ関数を利用することもできる。従って、様々な機能・形態のものが作成可能であると言える。

６－３．基幹部分でのプラグイン DLL の管理機能の作成

プラグイン DLL を管理する各種機能を整理するために、基幹部分側に CPlugin というクラスを新たに作成した。この中では以下の処理を行っている。

① ライブラリ関数の一覧を記載した plugin.tab（テキスト・ファイル）を読み込みメニューに追加する。

② ユーザーが、メニューからあるプラグインを選択した場合、当該プラグイン DLL をロードし、エントリー・ポイント関数を呼び出す。

③ ユーザーが、プラグイン DLL の側での操作の中で「終了」を選択した場合に、CPlugin の中の終了処理を要求する。この終了処理の中では、プラグイン DLL のアンロードのためのタイマー割り込みによるスレッドを立て、DLL の全てのダイアログが閉じたことを確認の上でアンロードを行う。

④ ユーザーが基幹部分において、DLL の終了を選択した場合に、DLL 側に対して終了処理を行うようにリクエストする。

なお、プラグイン DLL の完成度が低く、その中でアクセス違反等が生じた場合には、操作中であった基幹部分も含め、アプリケーションが異常終了することは避けられない。また、DLL の中でメモリー・リークが生じた場合には、基幹部分が終了するまで解消しない。この２点は、CreateProcess 関数により起動される外部関数とは異なっている。

この他に、CPlugin から各プラグイン DLL を起動する際には、display_dialog という共通のエントリー・ポイントを設けている。これにより、基幹部分側の重要な情報を引数として渡している。更に、終了時点での制御にもこの関数を用いている。

６－４．プラグイン DLL と基幹部分のインターフェースの詳細

プラグイン DLL は、ロードされた後に起動される display_dialog(CMainFrame*) という関数を用意していなければならない。この関数は、_declspec(dllexport)宣言により、基幹部分側に対してエクスポートされるように定義する。基幹部分は、プラグイン DLL をロードした後に、この関数に引数として、基幹部分側の pMainFrame ポインタを渡す。プラグイン DLL の側では、このポインタを手掛かりとして、基幹部分側のデータにアクセスする。

終了時点のアンロード処理は、起動時よりも条件がシビアとなる。基幹部分側からプラグイン DLL のアンロードを実行した時点で、プラグイン側が起点となるスレッドが何か動いていると、アクセス違反が生じ、エラー終了となる。そこで基幹部分側では、プラグイン DLL が既に終了していることを確認したうえで、アンロードしなければならない。アン

ロードは基幹部分側の関数をスレッドの起点として実行するため、プラグイン DLL 側で基幹部分側のアンロードを実行する関数を起動するわけにはいかない。

そこで、本システムにおいては、タイマー割り込みを使用して基幹部分側に別スレッドを発生させ、そのコールバックの中でアンロードを実行することとした。

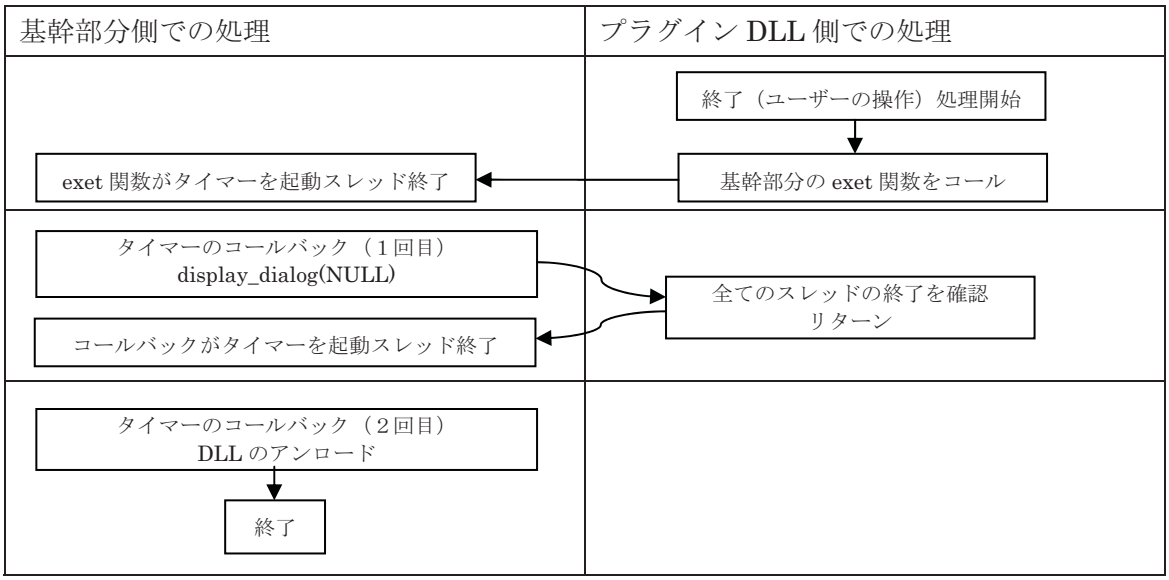


図６－１：プラグイン DLL 側からの終了要求に基づく、アンロードまでの処理

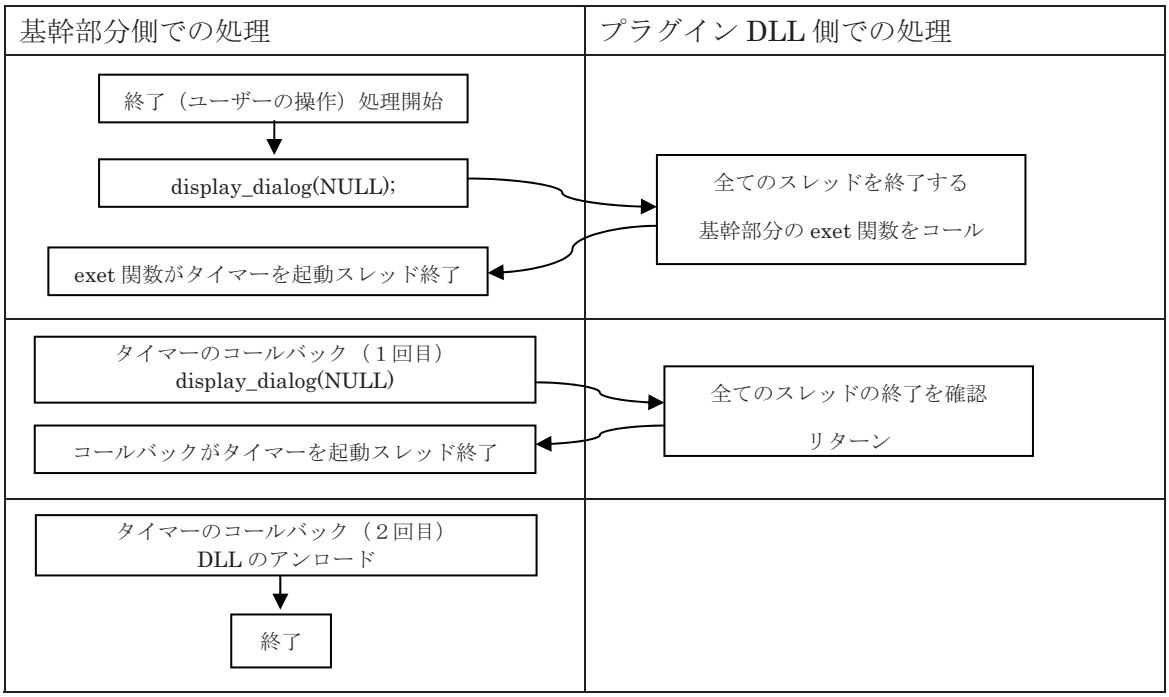


図６－２：基幹部分側からの終了要求に基づく、アンロードまでの処理

終了時点では、この引数が NULL だった場合には、基幹部分側からの終了リクエストと解釈して、終了処理に入る。その終了処理の中では、基幹部分側に対して、アンロード要求を出す。基幹部分側では、DLL からのアンロード要求を受けない限り、アンロードを実行しない。また、DLL 側から基幹部分に対してアンロード要求が出された場合においては、

基幹部分側でアンロードを行う直前に再確認のために display_dialog(NULL)が呼び出される。DLL 側では、全てのダイアログが閉じていることを確認の上、リターンする。

基幹部分側（CPlugin）では、アンロード要求を受けると、タイマーをセットする。実際のアンロードは別スレッドであるタイマーのコールバック関数の中で実行する。このコールバック関数においては、再度 DLL 側の display_dialog 関数を引数 NULL で呼び出し、全てのダイアログが閉じていることを確認した上で、アンロードを行う。

なお、CPlugin に関して、以下の２点を改良する余地がある。

①複数のプラグインが起動できるようにする

現在実装例としているプラグイン DLL はいずれも地形編集機能であるため、二つのプラグインが同時に起動して相互に干渉すると、不具合が生じる場合がありうる。そこで、プラグインを起動しようとした時点で実行されている別のプラグイン DLL が存在すれば、基幹部分の側から終了要求を出すようにしている。将来的には複数の性格が異なるプラグインが協調して動作するようなことも考えられるので、そのような場合には複数のプラグイン DLL の同時起動の仕組も検討する必要が生じるかも知れない。

②プラグインからの終了要求は、SendMessage 関数による CMainFrame への通知を用いている。

６－５．サンプル実装したプラグイン DLL

従来の枝分かれバージョンから、以下の機能を取り出し、プラグイン DLL の実装例とした。

（１）地形編集機能（land.dll）

起動した時点で、機能選択画面が開く。ユーザーはラジオボタンで、標高面作成、頂点移動、地形切断、地形の細分化と最適化、側面と底面の追加のいずれかの機能を選択する。各機能において終了することにより、プラグイン DLL 全体が終了する。

A. 機能選択画面

Land.dll は、地形編集に関するいくつかの機能をまとめた DLL である。プラグインを起動すると最初に機能選択画面を表示し、ここから個別の機能に分岐する。また、それぞれの機能を選択したうえで HELP により各機能の解説を読むことができる。

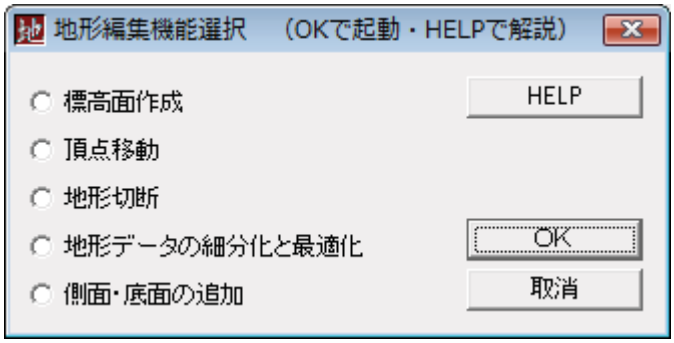


図６－３：地形編集機能選択画面

リソース：IDD_LAND_DLG_MENU ハンドラ：CLandMenu (LandMenu1.cpp)

ヘルプ：land.txt

B. 標高面作成

指定した高さの標高面を作成する。山頂の場合は、地面の下に形成され、窪地の場合は地面の上の水平面となる。処理に先立ってエリア指定を行うと、その範囲の標高のレンジを表示する。そのレンジの中での、面を作成する標高を指定し実行する。

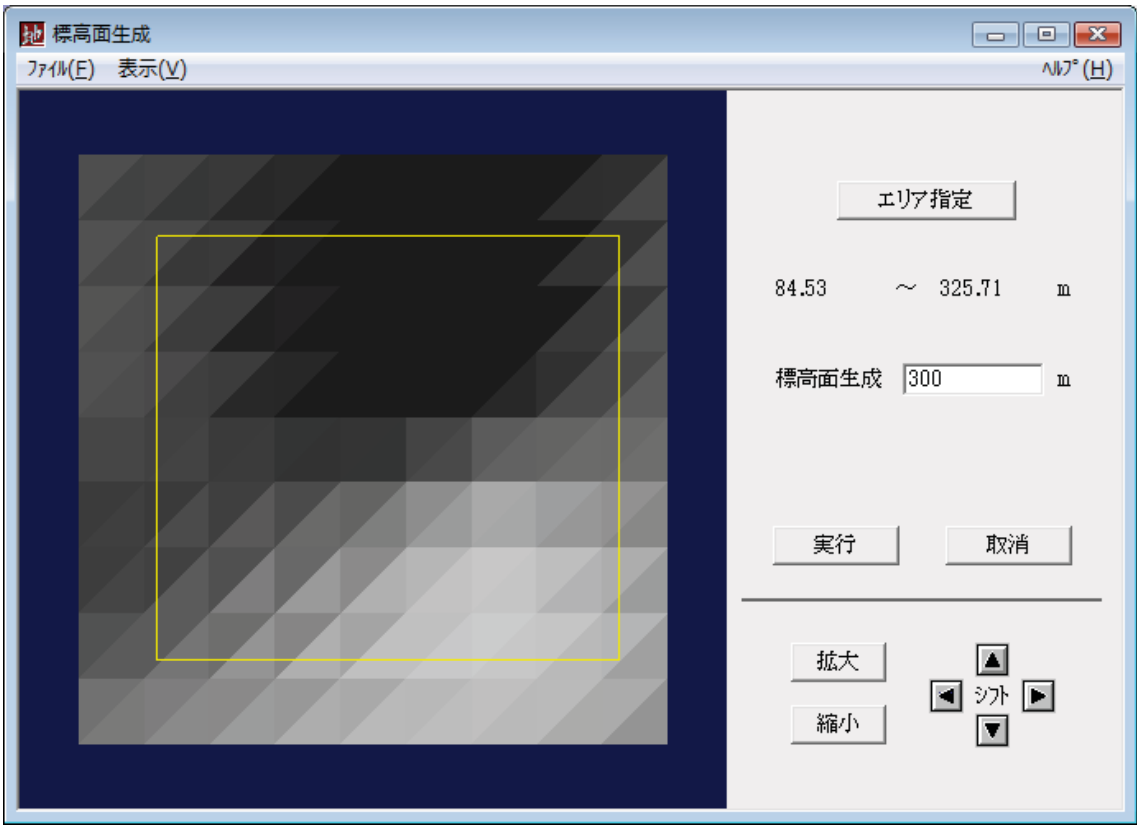


図６－４：標高面生成画面

①メニュー：IDR_MENU_LAND_HAICHI

メニュー構成 ［ファイル(F)］ ＋［閉じる］ダイアログを終了する ［表示(V)］ ＋［グリッド］５０m間隔のグリッドを表示する ＋［メジャー］縮尺を表示する ［ヘルプ］

②OpenGL 画面：CMyOrthoVW (myorthovw.cpp)

③右側ダイアログ：

リソース：IDD_LAND_DLG_HYOUKOU

ハンドラ：CHyoukouWnd (hyoukouwnd.cpp)

④ヘルプ：hyoukou.txt

C. 頂点移動

自由曲面の編集を指向した機能。変形の中心や、影響範囲を指定することにより、連続的に地形の一部を盛り上げたり、水平移動したりすることができる。

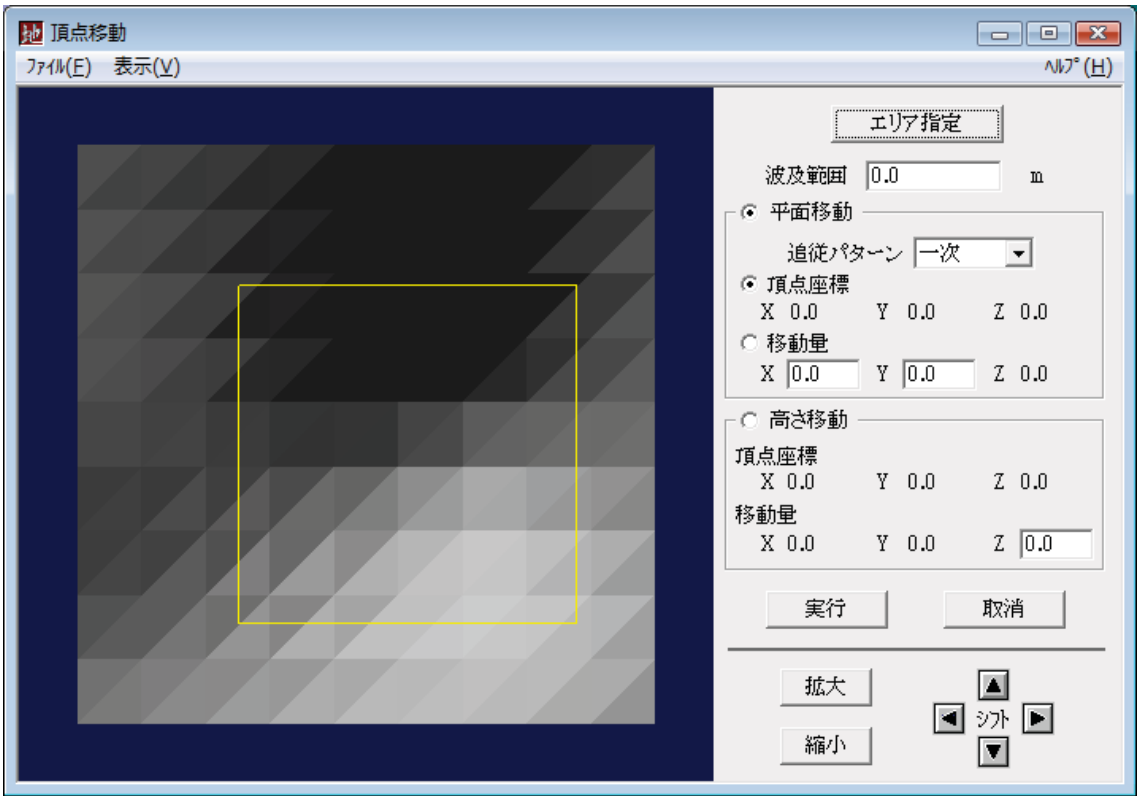


図６－５：頂点移動画面

① メニュー：IDR_MENU_LAND_HAICHI

ハンドラ：CPointMoveWnd (pmovewnd.cpp)

メニュー構成 ［ファイル(F)］ ＋［閉じる］ダイアログを終了する ［表示(V)］ ＋［グリッド］５０m間隔のグリッドを表示する ＋［メジャー］縮尺を表示する

② OpenGL 画面：CMyOrthoVW::m_orthoView (MyOrthovw.cpp)

③ 右側ダイアログ

リソース：IDD_LAND_POINT_MOVE　CDialogBar::m_Prm_Bar

ハンドラ：CPointMoveWnd (pmovewnd.cpp)

⑤　ヘルプ：pmove.txt

D. 地形切断

画面操作で多角形を指定し、その範囲内の地形を切り抜く。

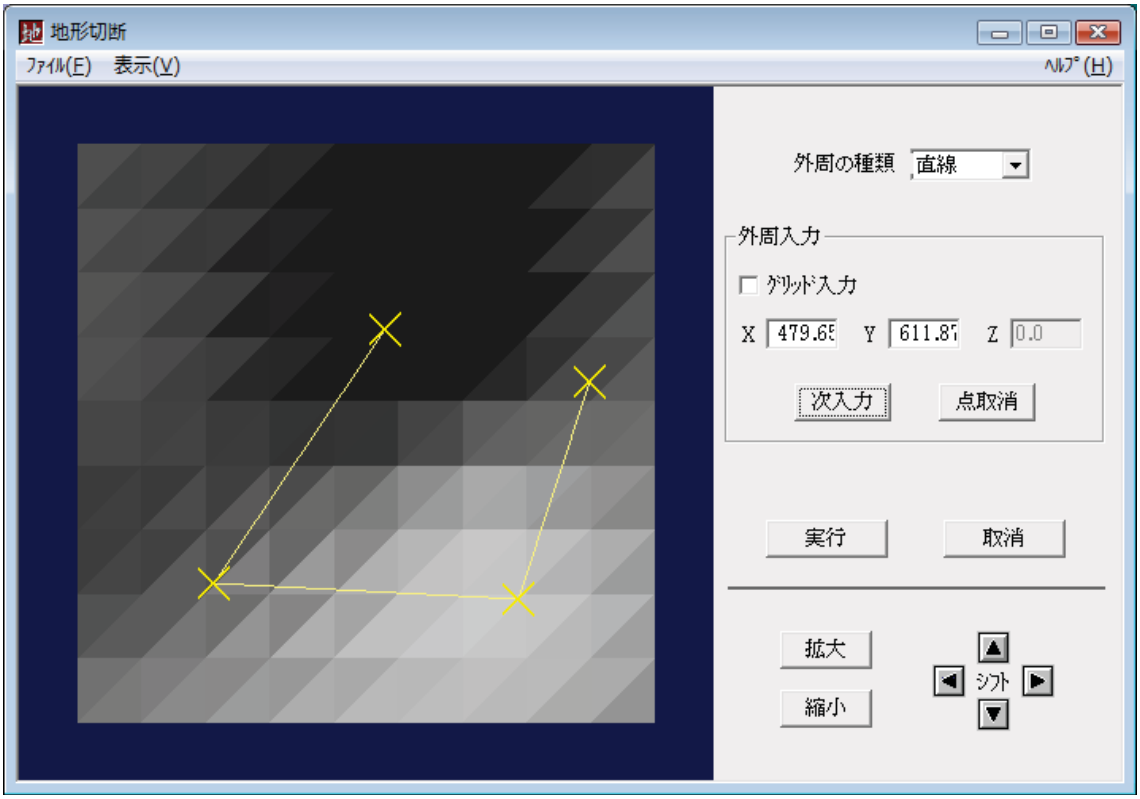


図６－６：地形切断画面

①メニュー：IDR_MENU_LAND_HAICHI

ハンドラ：CTopoCutWnd (topocutwnd.cpp)

メニュー構成 ［ファイル(F)］ ＋［閉じる］ダイアログを終了する ［表示(V)］ ＋［グリッド］５０ｍ間隔のグリッドを表示する ＋［メジャー］縮尺を表示する

②OpenGL 画面：CMyOrthoVW::m_orthoView (myorthovw.cpp)

③右側ダイアログ：

リソース：IDD_LAND_DLG_TOPO_CUT

ハンドラ：CDialogBar CPointMoveWnd::m_Prm_Bar (topocutwnd.cpp)

④ヘルプ：topocut.txt

E. 地形データの細分化と最適化

地形を構成する三角形分割に対して、統計的・大局的な加工を行い、細分割したり、平坦に近い領域で面の統合を行ったりする。

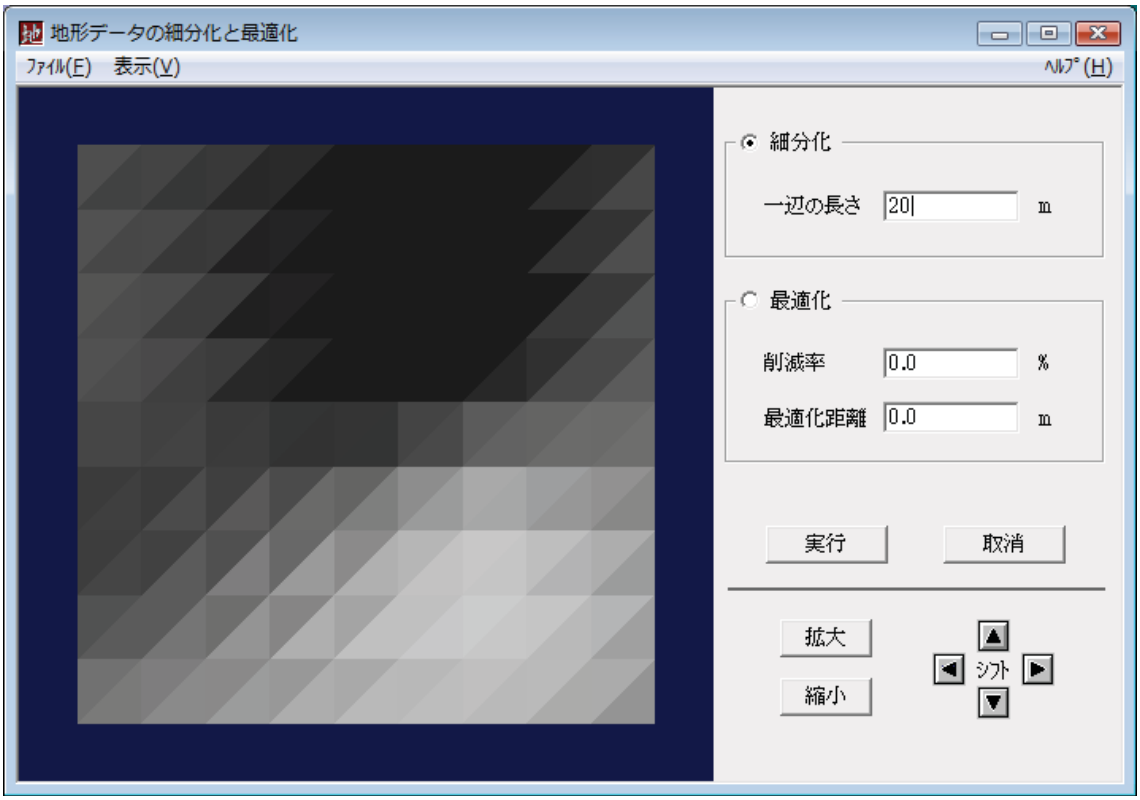


図６－７：地形データの細分化と最適化画面

①┌ メニュー：IDR_MENU_LAND_HAICHI

メニュー構成
［ファイル(F)］
＋［閉じる］ダイアログを終了する
［表示(V)］
＋［グリッド］５０m間隔のグリッドを表示する
＋［メジャー］縮尺を表示する

② OpenGL 画面：CMyOrthoVW*m_orthoView (orthovw.cpp)

③ 右側ダイアログ：リソース：IDD_LAND_TOPO_EDIT

ハンドラ：CTopoEditWnd (topoeditwnd.cpp)

メンバ変数 CDialogBar m_Prm_Bar

④ ヘルプ：topoedit.txt

F. 側面と底面の追加／削除

地表面だけから成る地形データに、側面や底面を付加して、BOOL 演算が行いやすい閉じた閉多面体を作る。合わせて、地形の体積や表面積などの諸元素を計算し表示する。

図６－８：地形諸元／側面底面追加画面

①リソース：IDD_LAND_DLG_SOLIDANAL

②ハンドラ：CSolidanal (kabe.cpp)

③ヘルプ：kabe.txt

（２）園路生成機能（parkroad.dll）

地物を平面図表示した画面表示の上で、地形の上に園路を構築する。この処理は、図形演算を用いて地形をカットする処理を含んでいる。機能的には高度であるが、ダイアログとしては一つだけのシンプルな構成である。

A. メイン画面

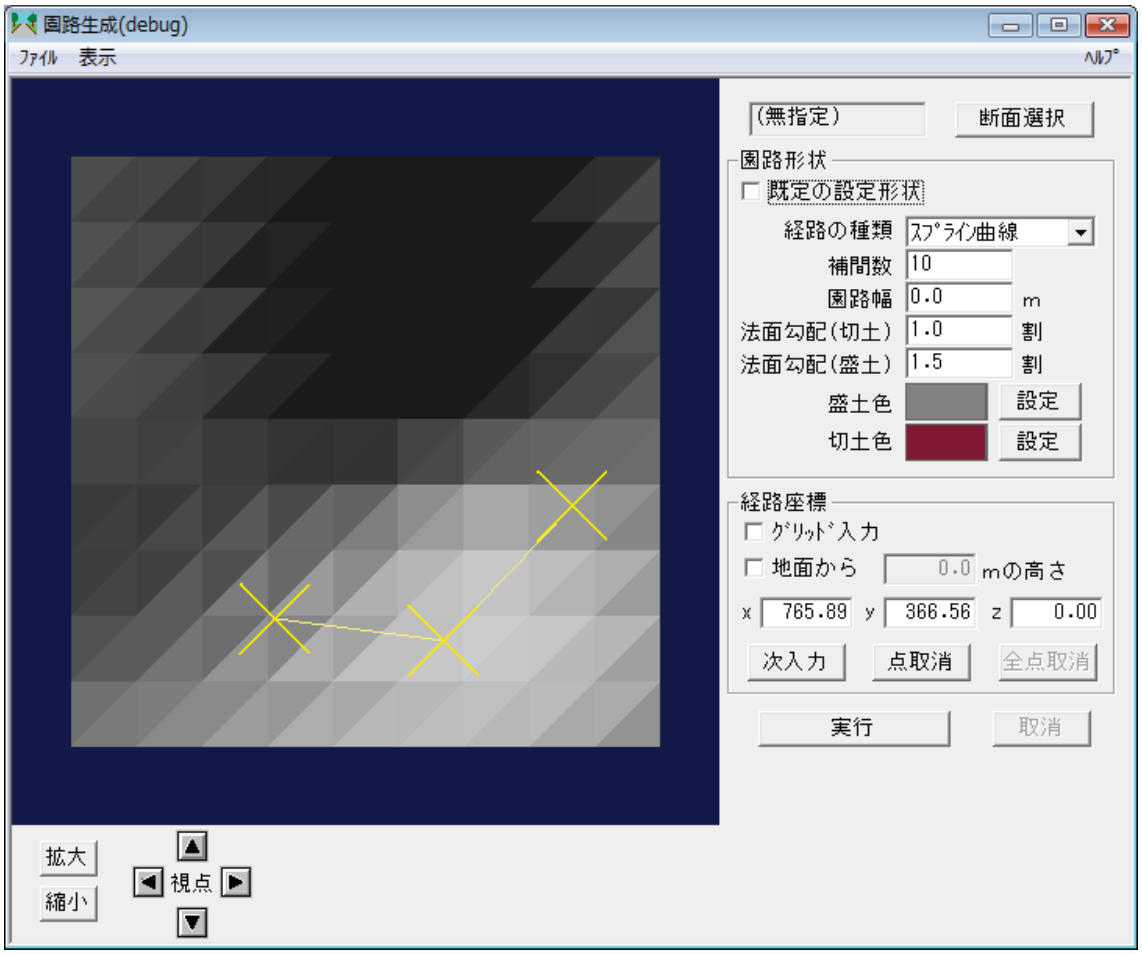


図６－９：園路生成画面

① メニュー：IDR_MENU_ENRO CParkRoadWnd (parkroadwnd.cpp)

② OpenGL 画面：CMyOrthoVW*m_orthoView(myorthovw.cpp)

③ 盛土色の小さな OpenGL 画面：CEnroNoriColWnd m_MoriColWnd (enronoricol.cpp)

④ 切土色の小さな OpenGL 画面：CEnroNoriColWnd m_KiriColWnd (enronoricol.cpp)

⑤ 右側ダイアログ：リソース IDD_DLG_ENRO_PARAM

ハンドラ：CDialogBar*m_ParamBar (parkroadwnd.cpp)

⑥ 下側ダイアログ：リソース IDD_DLG_ENRO_VIEW_CTL

ハンドラ：CDialogBar*m_VwCtrlBar (parkroadwnd.cpp)

⑦ ヘルプ：parkroad.txt

メニュー構成

［ファイル(F)］
＋［経路の新規作成］入力済みの経路を削除し新たな経路を作成する
＋［経路の読み込み］経路をファイルから読み込む
＋［経路を上書き保存］入力したファイル、先刻保存したファイルに上書き保存
＋［経路に名前を付けて保存］新しいファイルに経路を保存する
＋［園路生成の終了］ダイアログを終了する
［表示(V)］
＋［グリッド］５０m間隔のグリッドを表示する
＋［メジャー］縮尺を表示する
＋［全体視界］全体を表示する
＋［表示モード］
＋＋［テクスチャ表示］テクスチャを表示する
＋＋［シェーディング表示］テクスチャなしで面を表示
＋＋［ワイヤーフレーム表示］ワイヤーフレーム表示

B. 園路断面選択

Kdb/Geometry ディレクトリに置かれた ENRO_SEC.set という定義ファイルに登録されている LSS-G 形式（拡張子.geo）の断面定義ファイルの一覧を表示し、ユーザーの選択を求める。

図６－１０：園路断面ファイル選択画面

リソース:IDD_ENRO_DLG_POTONGAN　ハンドラ: CRoadPotongan (roaddan.cpp)

C. カラー、マテリアル、テクスチャ選択

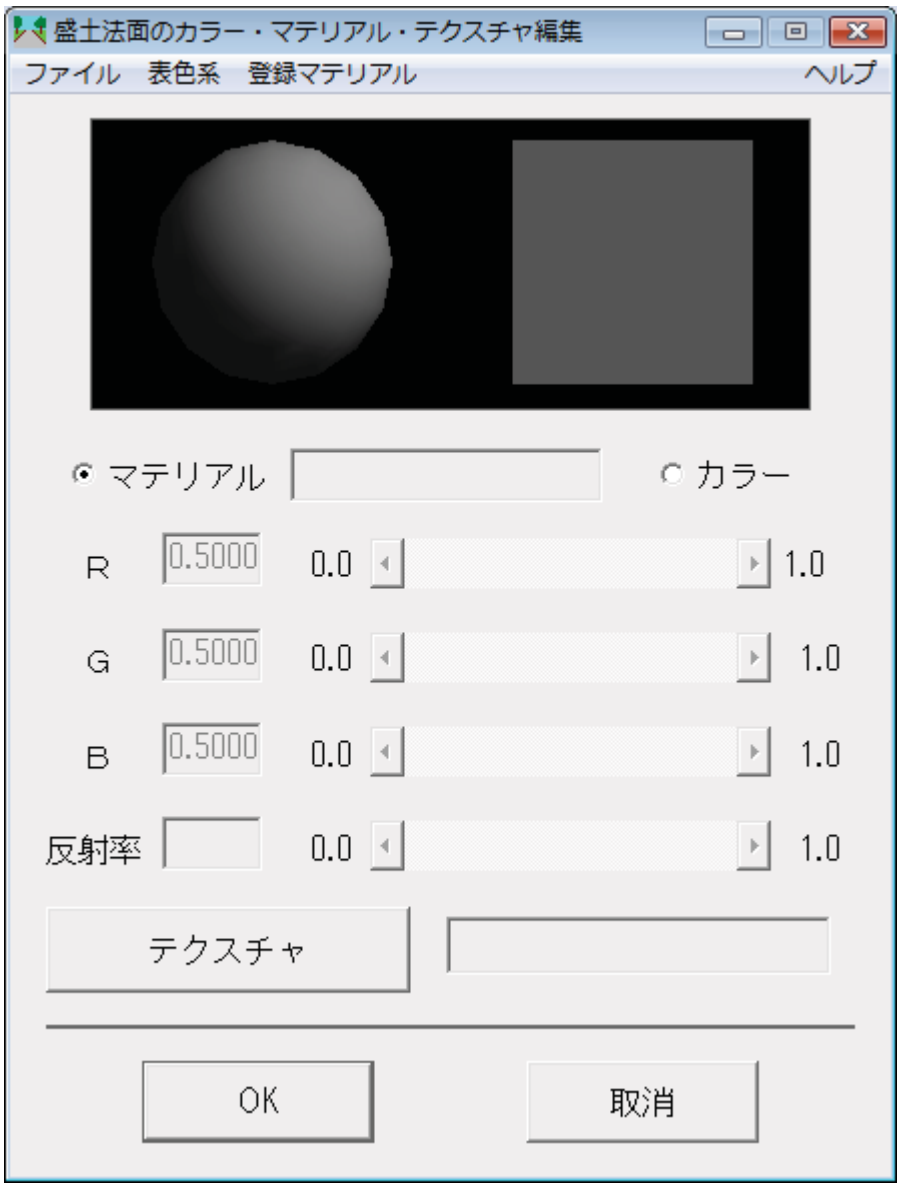


図６－１１：法面のカラー・マテリアル・テクスチャ設定画面

①メニュー：IDR_MENU_ENROMAT (enromat.cpp)

メニュー構成
［ファイル］
＋［閉じる］
［表色系］
［登録マテリアル］
＋［マテリアル選択画面］
＋［マテリアルファイル選択］
＋［選択済マテリアルファイル］

＋［指定マテリアル内容表示］

②OpenGL 画面：CNori2Dlg m_dlg (enromatlist.cpp)

③リソース：IDD_DIALOG_ENROMAT

ハンドラ：CEnroMat (enromat.cpp)

D. テクスチャ選択画面

［テクスチャ］ボタンで起動する。

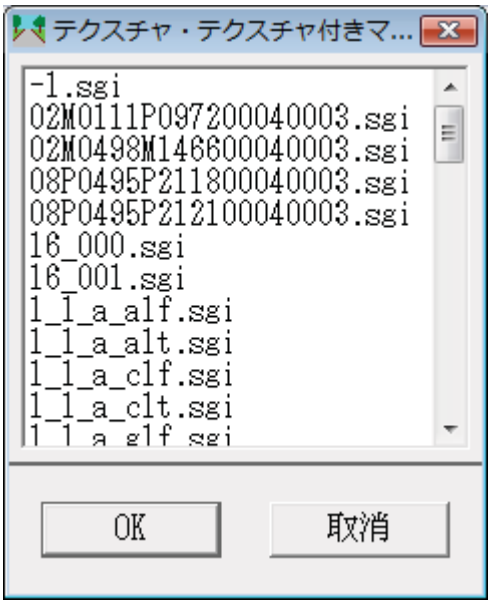


図６－１２：テクスチャ選択画面

リソース：IDD_DLG_TEXLIST ハンドラ：CEnroTexList (enrotexlist.cpp)

E. マテリアル選択画面

メニューの［登録マテリアル］［マテリアル選択画面］で開く。

図６－１３：マテリアル選択画面

リソース：IDD_DLG_NORI_MAT_LIST　ハンドラ：CEnroMatList (EnroMatList.cpp)

（３）道路法面生成機能（nori.dll）

古いバージョンの sim.exe に含まれていた機能を分離独立させたものである。道路法面の形状計算に、OpenGL のデプス・バッファを用いているため、細部を接近して見ると粗い場合がある。

A. メイン画面

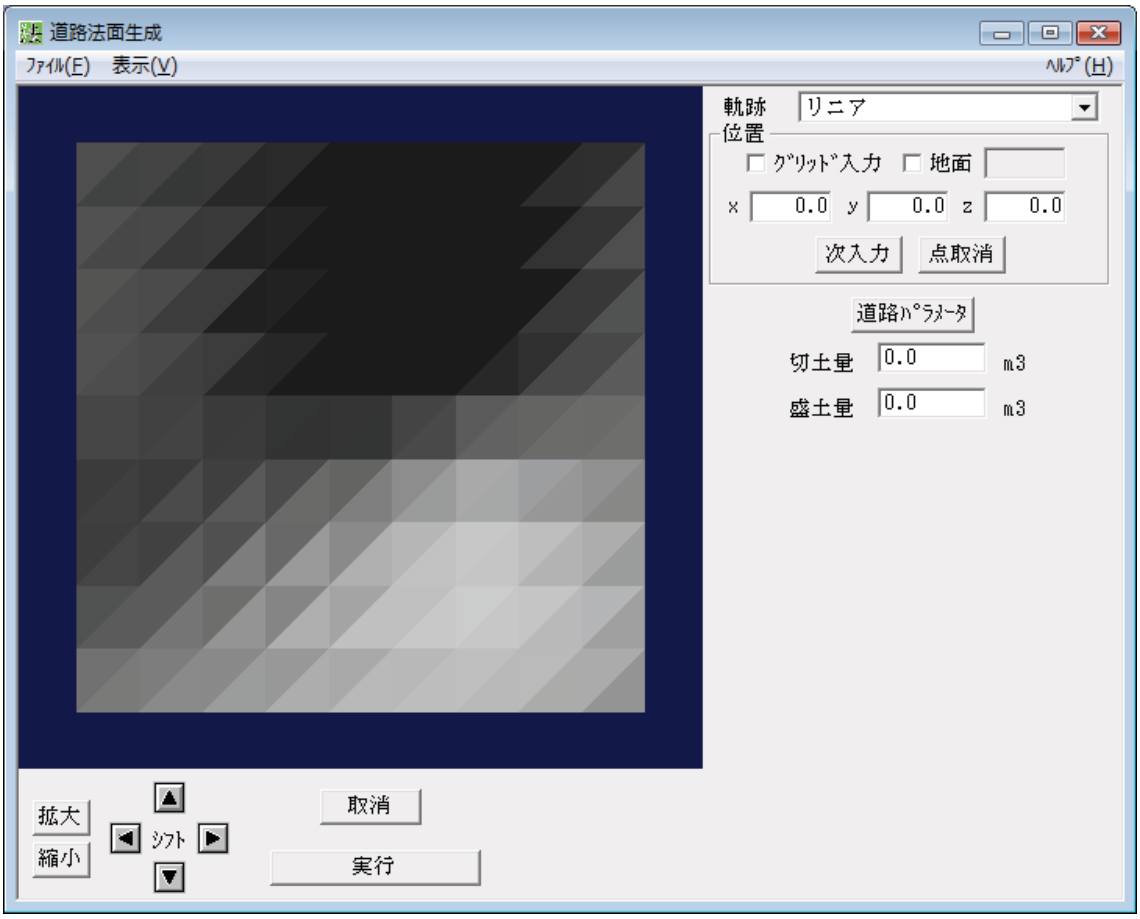


図６－１４：道路法面生成画面

① メニュー：IDR_MENU_HAICHI

メニュー構成
［ファイル(F)］
＋［閉じる］ダイアログを終了する
［表示(V)］
＋［グリッド］５０m間隔のグリッドを表示する
＋［メジャー］縮尺を表示する
［ヘルプ］ヘルプを表示する

② OpenGL 画面：CMyOrthoVW CNoriWnd::m_orthoView (noriwnd.cpp)

③ 右側ダイアログ：リソース:IDD_DLG_NORI_PRM
CDialogBar m_Prm_Bar (noriwnd.cpp)
④ 下側ダイアログ：リソース:IDD_DLG_NORI_CTL
CDialogBar m_Ctl_Bar (noriwnd.cpp)
⑤ ヘルプ：noriwnd.txt

B. 道路パラメータ設定画面

メイン画面の、［道路パラメータ設定］ボタンで開く(m_dlg)。

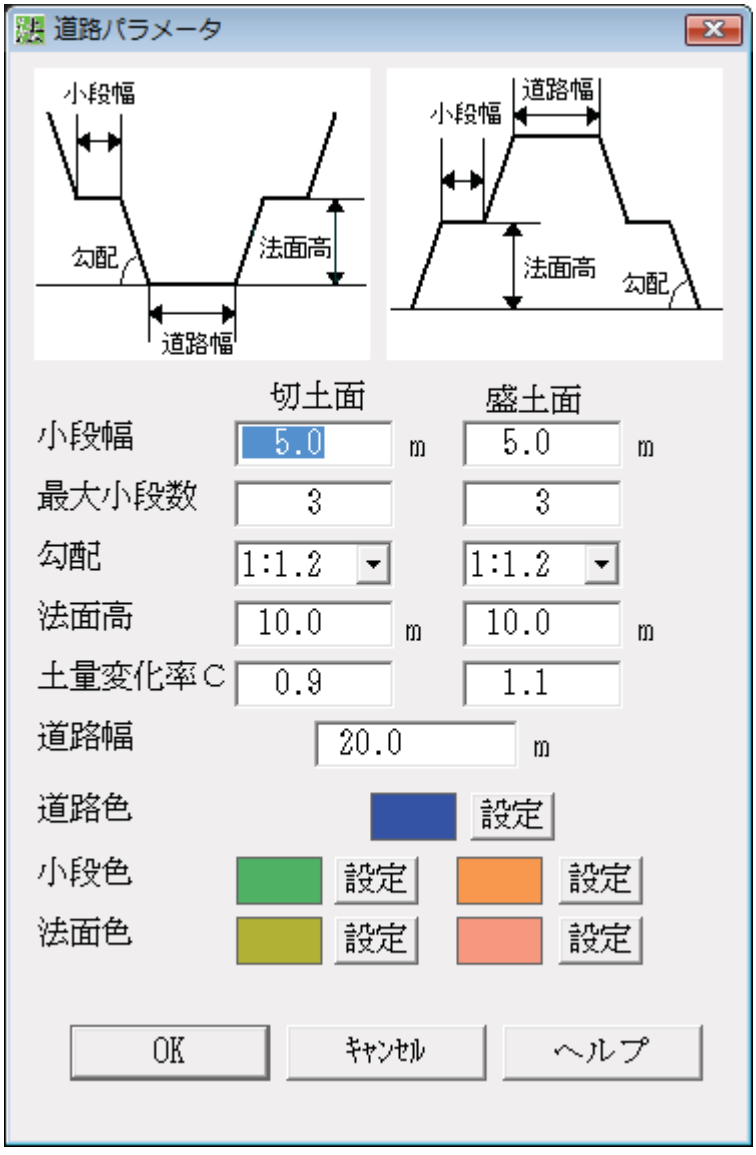


図６－１５：道路パラメータ設定画面

① リソース：IDD_DLG_ROAD_PARAM
② ハンドラ：CNoriDlg (noridlg.cpp)
③ 道路色 OpenGL 画面：CRoadColWnd m_RoadCol_Wnd (roadcol.cpp)

④ 切土小段色 OpenGL 画面：CKShoColWnd m_KshoCol_Wnd (kshocolw.cpp)
⑤ 切土法面色 OpenGL 画面：CKNoriColWnd m_KnoriCol_Wnd(knoricol.cpp)
⑥ 盛土小段色 OpenGL 画面：CMShoColWnd m_MshoCol_Wnd (mshocolw.cpp)
⑦ 盛土法面色 OpenGL 画面：CMNoriColWnd m_MnoriCol_Wnd(mnoricol.cpp)
⑧ ヘルプ：noridlg.txt

C. マテリアル選択画面

道路パラメータ設定画面の、各部分の［設定］ボタンで開く。それぞれの部分のマテリアルを設定する。

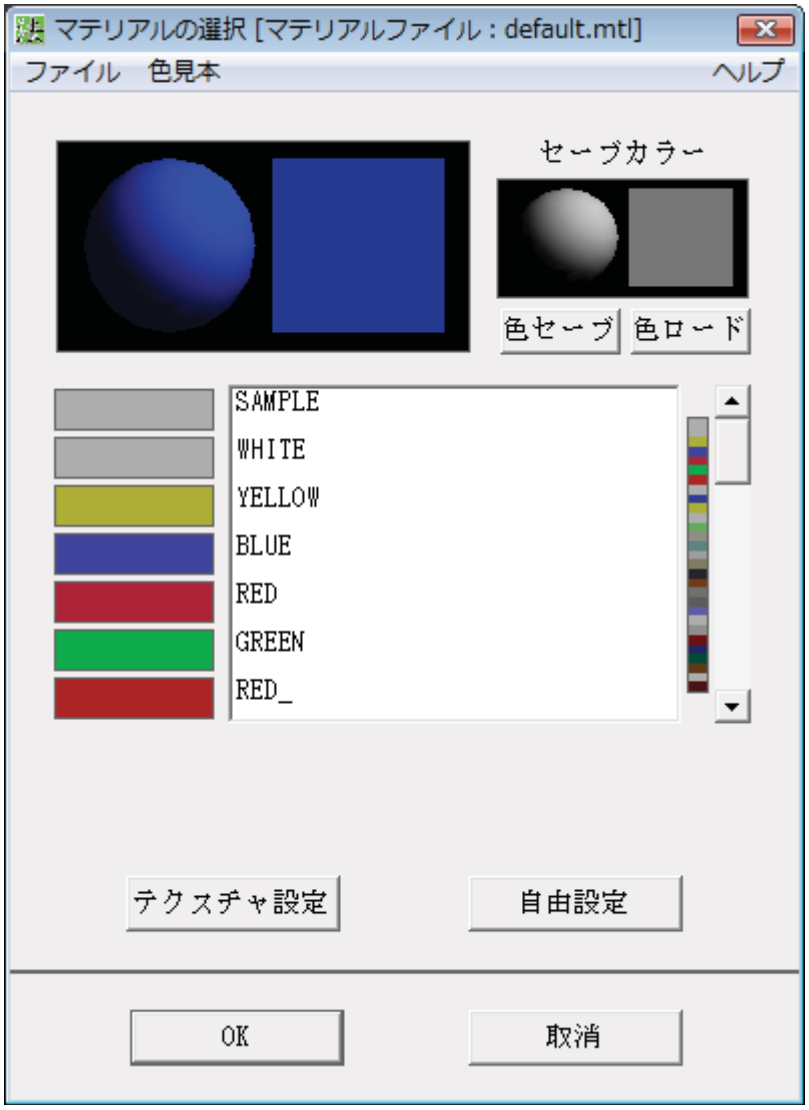


図６－１６：道路マテリアル設定画面

① メニュー：IDR_MENU_MATERIAL2

メニュー構成
［ファイル］

＋［閉じる］ダイアログを終了する
［色見本］使用可能なマテリアルファイルの一覧をサブメニューとして表示する

② リソース：IDD_DIALOG_MATERIAL ハンドラ：CEditMaterial(editmat2.cpp)
③ 左色球 OpenGL 画面：CSphereDlg m_KyuDlg (spheredlg.cpp)
④ 右セーブカラー色球 OpenGL 画面：CSphereDlg m_saveKyuDlg(spheredlg.cpp)
⑤ 左側 7 箱 OpenGL 画面：CMatFrm m_colFrm[7] (matfrm.cpp)
⑥ 右側櫛の歯 OpenGL 画面：CColorSashDlg m_obiDlg (colsashdlg.cpp)
⑦ ヘルプ：editmate.txt

このダイアログは、基幹部分にも存在するが、これを兼用することにより編集結果の適用先をオプショナルなダイアログに設定することを避け、同じダイアログのコピーを道路法面生成にも独自に用意した。

D. カラーの自由設定

マテリアル編集画面の［自由設定］ボタンで開く

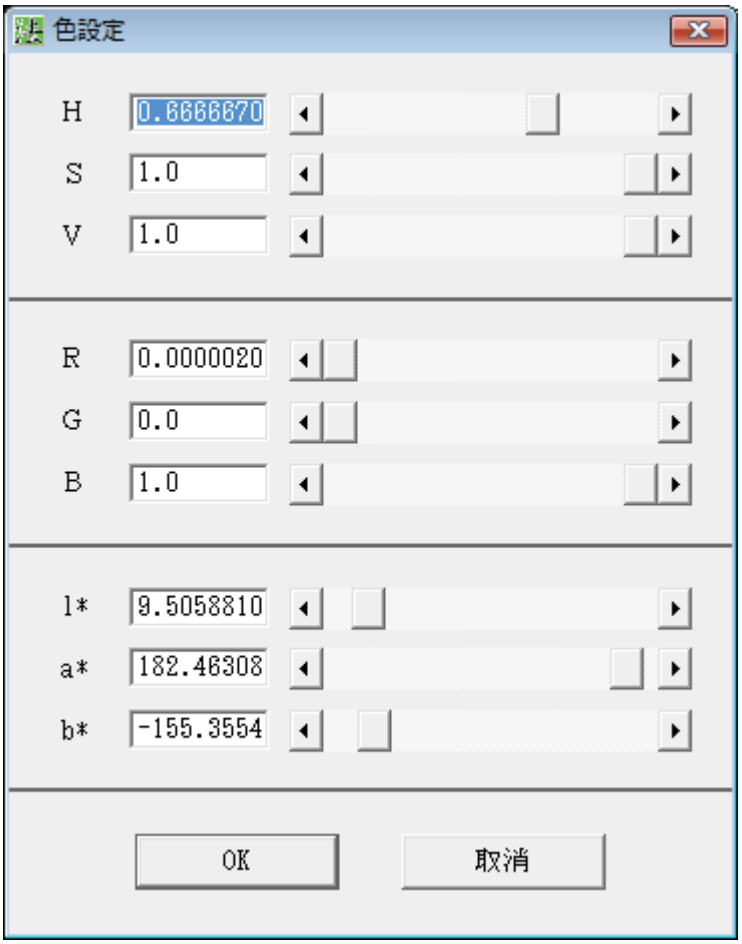


図６－１７：道路カラー設定画面

リソース：IDD_DIALOG_COLOR_SET ハンドラ：CColorSet(colorset.cpp)

E. テクスチャ設定画面

マテリアル編集画面の［テクスチャ］ボタンから開く。

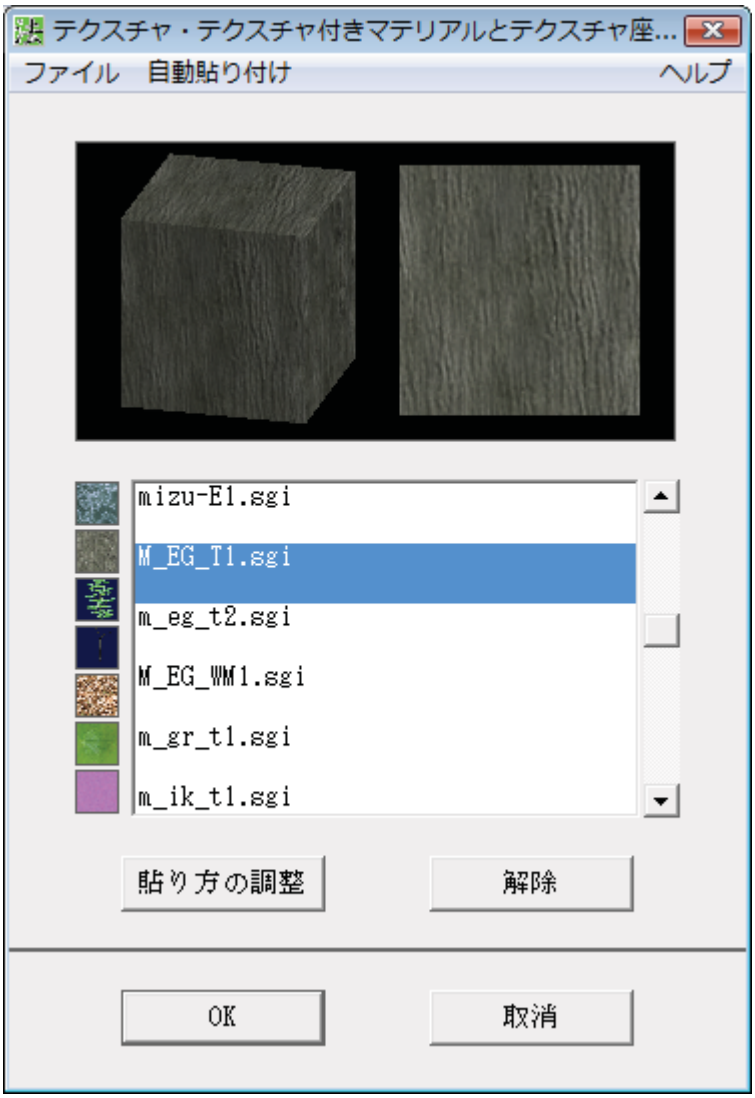


図６－１８：道路テクスチャ設定画面

① メニュー：IDR_MENU_TEXTURE2

メニュー構成 ［ファイル］ ＋［閉じる］ ［自動貼り付け］autotex.set ファイルで定義したメニューを表示する

② 上の OpenGL 画面：CCubeDlg m_boxDlg (cubedlg.cpp)

③ 左の 7 箱の OpenGL 画面 CTexFrm m_texFrm[7] (texfrm.cpp)

④ ダイアログ

リソース：IDD_DIALOG_TEXTURE ハンドラ:CEditTexture (edittex2.cpp)

⑤ ヘルプ：edittex2.txt

F. 貼り方の調整

テクスチャ編集画面の［貼り方の調整］ボタンで開く。

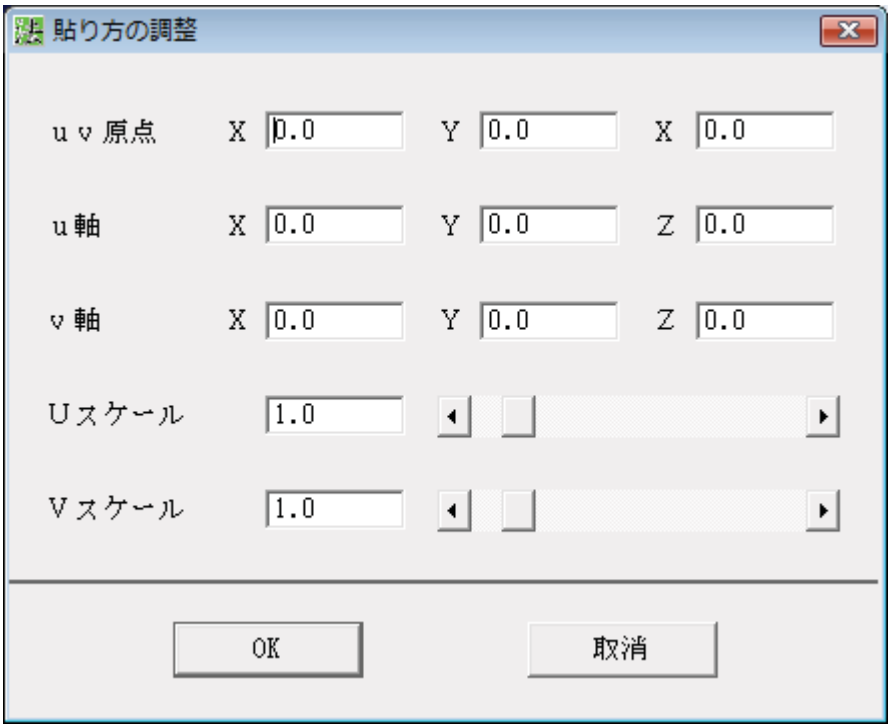


図６－１９：道路テクスチャの貼り方の調整画面

ダイアログ：IDD_DIALOG_TEXTURE_MAP ハンドラ：CTextureMap (textmap.cpp)

（４）トンネル生成機能(tunnel.dll)

古いオープンソースの形状計算ライブラリを用いて、地形にトンネルを通す。地形のメッシュが粗く、メッシュを構成する三角形の中をトンネルが完全に貫通する（地形の辺とトンネル形状の面が干渉しない）場合に、エラーを起こす。

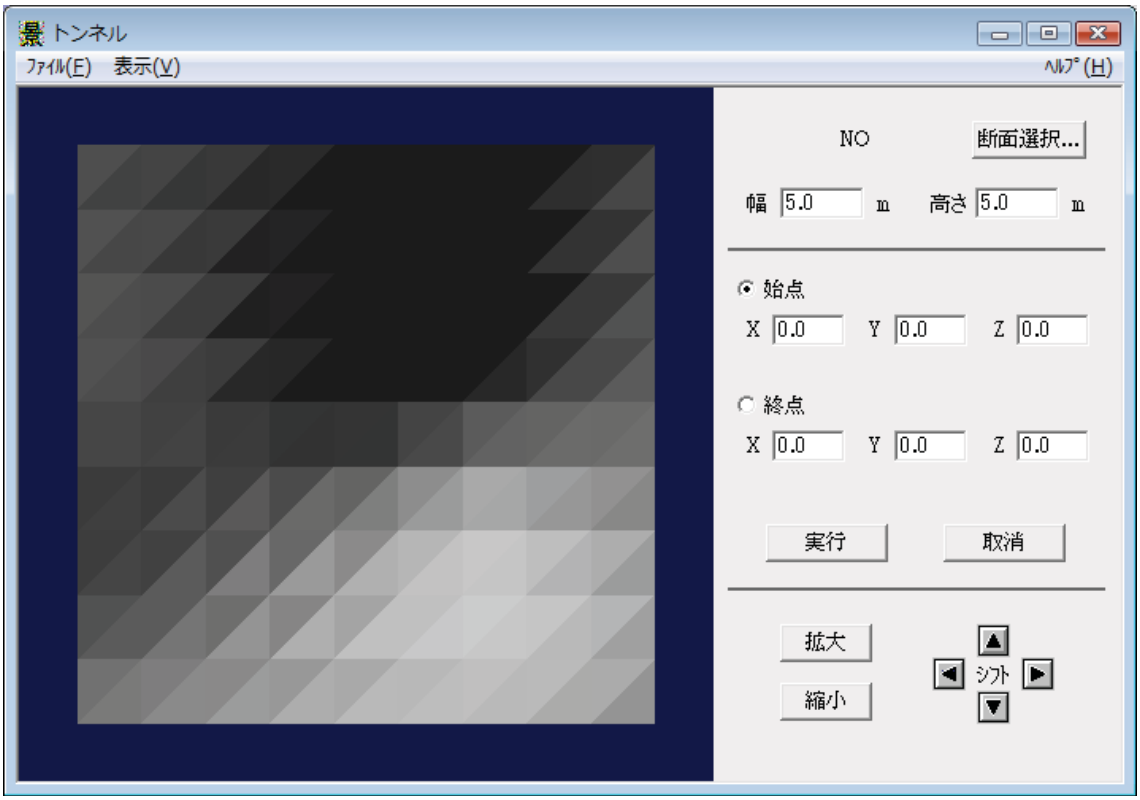


図６－２０：トンネル画面

①メニュー：IDR_MENU_LAND_HAICHI

メニュー構成
［ファイル］
＋［軌跡読込］
＋［軌跡保存］
＋［上書保存］
＋［終了］ダイアログを終了する
［表示］
＋［グリッド］
＋［メジャー］
＋［全体視界］
＋［上下範囲］
＋［縦横範囲］
＋［表示モード］

＋＋［テクスチャ表示］
＋＋［シェーディング表示］
＋＋［ワイヤーフレーム表示］
＋＋［オプション設定］
＋＋＋［地面のみ表示］
＋＋＋［地面テクスチャ表示］
＋＋［オプション解除］

② ハンドラ：CTunnelWnd (tunnelwnd.cpp)
③ OpenGL 画面：CMyOrthoVW m_orthoView (myorthovw.cpp)
④ 右側のダイアログ：
リソース:IDD_DLG_TUNNEL ハンドラ：CDialogBar m_Prm_Bar (tunnelwnd.cpp)
⑤ ヘルプ：tunnel.txt

６－６．プラグインＤＬＬと、三次元図形演算機能 (2011 年 3 月追記)

本章に掲載したプラグインＤＬＬの例は、三次元図形演算機能の発展過程を示している。
(1)道路法面生成機能(nori.dll)

1996 年に開発し、Ver.2.05 から標準機能に搭載した機能であり、地形の上に道路の線形を指定すると、地形と道路法面の図形演算を行う。

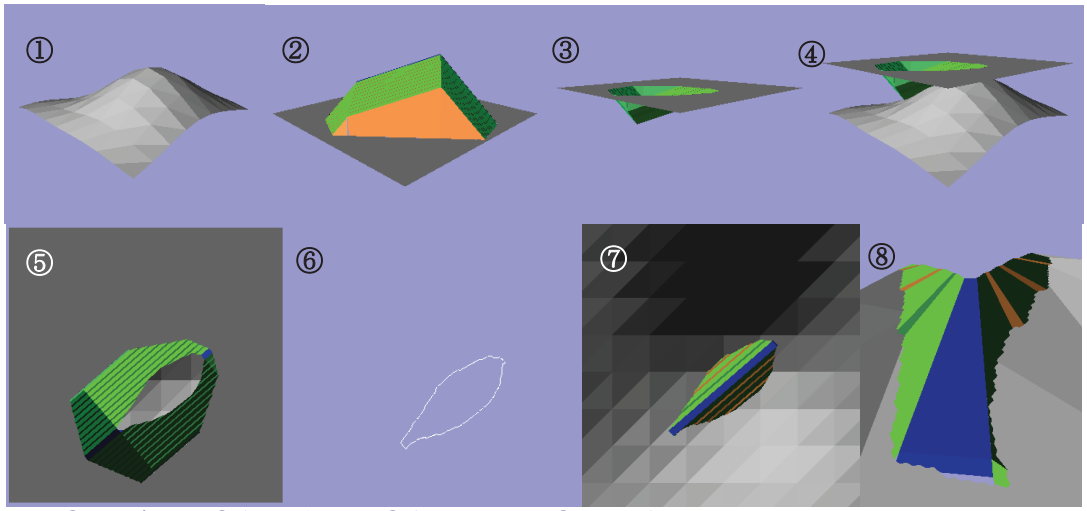
法面には、切土と盛土がある。計画された道路が地面よりも低い場合には、地形の一部を切り取り、その下に斜めの切土の法面を作成する必要がある。また、道路が地面よりも高い場合には、地形の上に斜めの盛土の法面を作成する必要がある。この場合、地形は残されていても、表示においては隠されるが、データ量や視点移動等の処理速度を考えると、不要となった部分を削除する方が無駄がない。更に、道路が斜面の等高線に沿って通るような場合には、山側に切土、谷側に盛土の法面を作成する必要がある。

道路法面生成(nori.dll)においては、この処理を、OpenGL のデプス・バッファの機能を用いて実現していた。処理過程を図６－２１に示す：１．道路面から仮定される、十分に大きなサイズの切土と盛土の法面をそれぞれ生成する②③。２．それぞれの法面と地形を重ねて④平面図表示を行う⑤。３．OpenGL のステンシル・バッファの機能を用い、地形が上となる平面領域と、法面が上となる平面領域を、ビットマップの形で取得する。４．地形と法面の境界の二次元形状を、折れ線として取得する⑥。５．この折れ線を用いて、地面の内、切土よりも上となる部分、及び盛土よりも下となる部分を削除する⑦。６．同じ折れ線を用いて、切土の内、地面よりも上となる部分、及び盛土の内、地面よりも下となる部分を削除する。７．最後に、地面と道路と切土面と盛土面の残された部分を合成する⑧。

この方法を用いた場合、たとえ非常に複雑な地形や道路であっても、計算処理のロジックは単純である。しかしながら、境界線の形状の計算精度は、デプス・バッファの解像度

（計算のために生成する画面の縦横ドット数）に依存する。しかし、デプス・バッファの解像度を高くしても、境界線の斜め直線区間は階段状となるため、境界線の折れ線の頂点数は非常に大きなものとなる。このため、二次元図形どうしの図形演算は時間のかかるものとなり、二次元の内外判定でエラーを発生する率が高くなる。

また、図形演算に平面図を用いていることから、垂直よりも大きな勾配（下向き）の面を有する図形に関する演算処理を行うことが原理的に不可能である。



①地形データ　②路面＋盛土面　③路面＋切土面　④地形＋路面＋切土面
（路面・法面は、断面形と中心線軌跡からパラメトリックに自動生成する）
⑤地形＋法面の相貫を平面表示画面で解析　⑥交差線（二次元）を求める
⑦これを用いて地形と法面を加工　⑧加工部分付近（デプス・バッファに起因する複雑な交差線）

図６－２１：道路法面の生成過程

この演算処理は、当初 u3 ライブラリの関数として実装された。Ver.2.09 への整理統合に伴い nori.dll として分離した際には、この三角形を対象とする図形演算処理機能を、このプラグインを構成する n3xxx.c のソースコードに移管した上でデバッグを加えた。

(2)トンネル生成機能(tunnel.dll)

1997 年以降に開発した、斜面に穴をあけ、トンネルの坑口を作成する機能である。トンネルの断面形状は、外部ファイルにより、自由に指定できる。幾何学的には、任意の断面を有する掃引体と、任意の形状を有する地形の間の演算処理の問題となる。

地面の内、トンネルの坑口付近に関して、トンネルの内部となる領域を切除する。一方、トンネルは、坑口部分で軸線に垂直な面で切断する。

この処理に際しては、デプス・バッファを使用せずに、図形どうしの相貫を直接座標計算する方法を用いた。地面の内、トンネルの内部となる部分を切除するためには、地面を構成する各面と、トンネルの側面を構成する各面との相貫を求め、不要となる部分を切除する、という処理を繰り返す。この処理においては、まず地形とトンネル側面の全てをまず三角形に分割した上で、三角形どうしの相貫関係を計算する処理とした。図形演算を三角形と三角形の相貫に還元することにより、場合分けを単純化することができる。さらに、トンネルの場合には、切り取る側の図形（トンネル）が、直線状の掃引体であることから、

地形の各面をこれと垂直な面に投影することにより、最終的には、二次元平面における三角形の切欠きの問題に還元することができる。

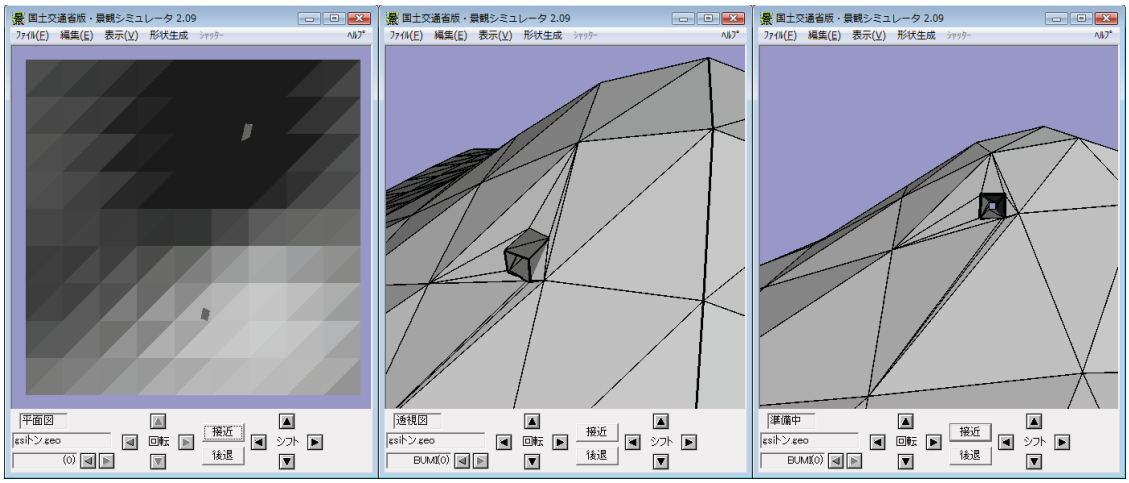


図６－２２：トンネルの生成過程

地面を構成する一つの三角形の切除処理においては、トンネルを構成する各三角形との相貫により、切れ目が生じた場合、切欠きの結果生じる図形を、次の処理に備えて更に三角形に分割する。また、一つの三角形が切断され、四角形と三角形に生じたような場合には、四角形を三角形に分割する処理を行う。このため、処理が終了した時点で元の一つの面が、多数の三角形に分割されたような状態となる（図６－２２）。

この処理を支援するために、別途地形処理機能(land.dll)を同時に用意した。この中には、地面を構成する面が４以上の辺を有していた場合に、全て三角形に分割する前処理機能や、後処理として、細分化された地形に関して、隣接する面であって向きの差が小さいものを再統合して、複雑さを減ずるポリゴン縮減機能を用意した（地形編集 land.dll から起動する「地形データの細分化と最適化」）。

この処理方法により、垂直よりも切り立った、下向きの面を持つ地形であっても、トンネルをあけることができるようになった。

三角形メッシュを用いた図形演算の機能は、tunnel.dll のビルドに含まれる u3cutmesh.c 及び u3fcom.c に移管しデバッグした。但し、地形を構成する面（三角形）に対してトンネルの断面が非常に小さく、三角形の辺に触れることなしに内部を貫通するような場合等に、演算に失敗する。

(3)園路生成機能

公園設計機能の開発を行った枝分かれバージョン(SimParkRoad.exe、非公開)においては、園路生成機能として、４以上の頂点を有する面から構成された地形に関しても、立体的に曲がりくねった園路とその法面と直接図形演算を行う必要が生じた。

この問題を解決するために、閉じた三次元図形どうしの演算を実行する関数を、基幹部分とは別にデバッグ・改良可能な DLL として改良することとし、本体側である開発中の実行形式 SimParkRoad.exe（非公開）とは分離して図形演算機能の改良を進めた。

Ver.2.09 への整理統合に際して、この開発用の実行形式から、園路生成のダイアログの部分を切り離し、プラグイン(ParkRoad.dll)に分離すると共に、改良過程にある図形演算機能も dll として外付けし、園路生成機能から必要に応じて起動できる構成とした。最終的に高い成功率が達成され実用的となったのは、2010 年末である。

この処理過程を、図６－２３に示す。１．ユーザーが指定した経路から８の字型の断面を持つ法面のテンプレートを作成する②。２．これを用いた図形演算で、地形①のうち不要となる部分を削除する③。３．経路から切土面⑤、盛土面④及び園路面を作成する。４．切土面・盛土面に関して地面の下（内部）となる領域を図形演算により切除する⑥。５．切除後残された地形、切土面、盛土面及び園路面を合成する⑦。

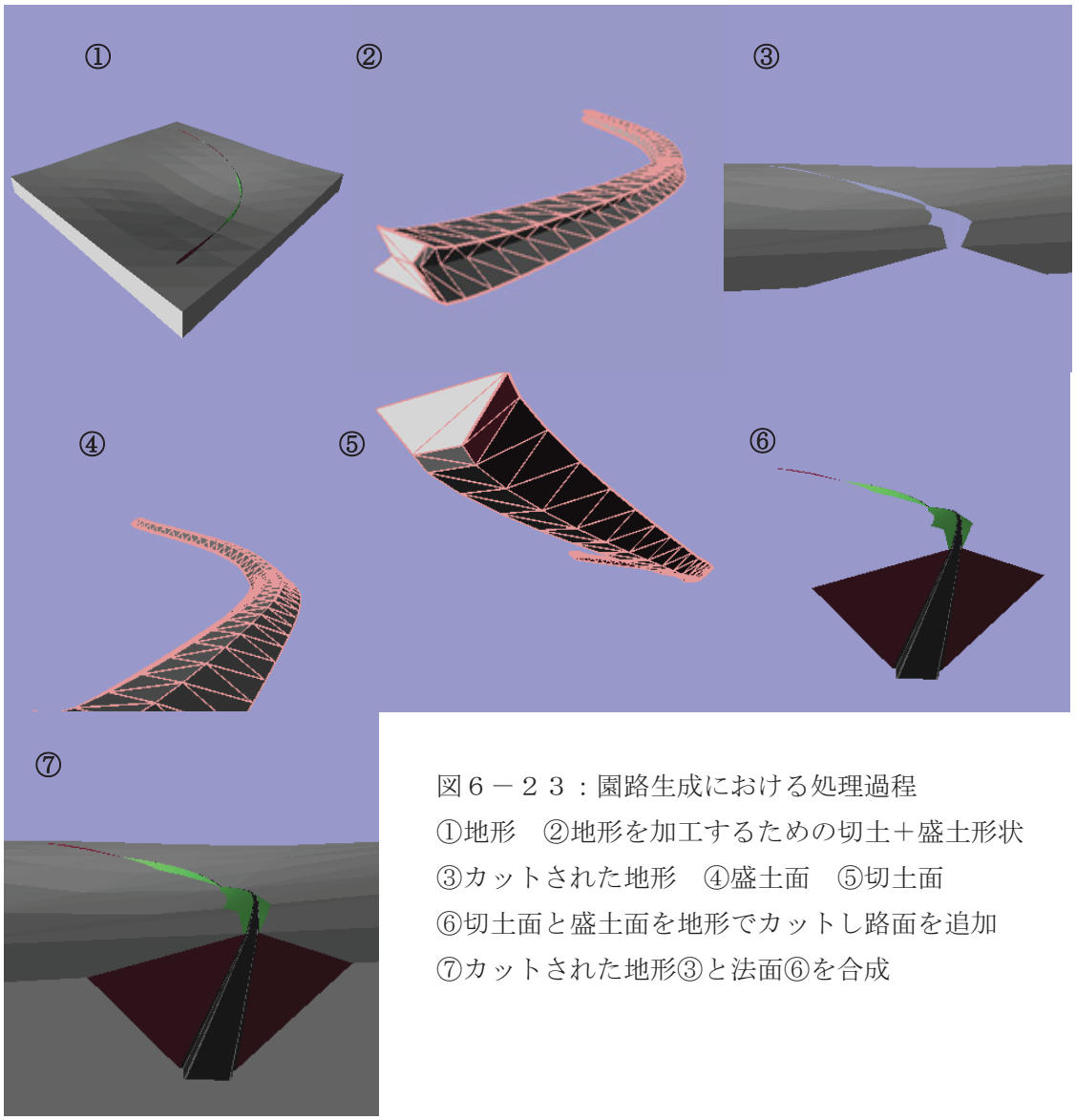


図６－２３：園路生成における処理過程
①地形　②地形を加工するための切土＋盛土形状
③カットされた地形　④盛土面　⑤切土面
⑥切土面と盛土面を地形でカットし路面を追加
⑦カットされた地形③と法面⑥を合成

なお、路面の断面は任意の形状をファイルで定義して指定することができるが、計算処理を単純化するために、法面の立体を作成する際には、園路断面は単純な線分として処理

している。最終的に合成する園路が複雑な断面を有する場合には、端面の一部である、園路面に対応する辺を、ユーザーが指定した形状の断面に差し替えている。この際に、捩じれた図形が生じる場合があるため、これを修正処理している。

図形演算機能は、今後開発するプラグイン DLL から利用することができる。その内部処理の詳細に関しては、別途資料の作成を予定している。

注釈：

1）統合化作業に着手した 2008 年 7 月時点で、以下の枝分かれバージョンが存在していた。

①Ver.2.07

2001 年に、まちづくり・コミュニケーション・システムのために改良した。インターネットを介してデータを取得する機能を付加したバージョンであり、Ver.2.09 に向けた統合作業開始時点で最も安定性が高い。Windows　XP に対応している。唯一の公開バージョンである。

②ステレオ版

2001 年に開発した。偏光メガネを用いて立体視するための画像を生成するためのバージョンである。2.07 と同時に開発したが、ソースコードが別となっている（同時公開）。

③Ver.4.0

1997 年頃から、ダム等の現場に対応するために、現場適用に伴い機能追加を行った、Ver.2.5、Ver.3.2 の延長上のバージョンである。影の表示機能、地形編集機能、トンネル生成、地形編集、動画合成機能、等が追加されているが、安定性に欠ける（Ver.3.2 実行形式を 1998 年 3 月に建設省土木研究所の WEB サイトから公開したが、Ver.4.0 は非公開）。

④ParkRoad

バグの少ない Ver.2.07 を出発点として 2004 年頃から、国営公園における景観検討に向けて機能追加を行った枝分かれバージョンである。GIS データから変換して作成した地形の上に、園路を作りこむ機能が追加されている（非公開）。

⑤SpeedUpSIM

2004 年頃、OpenGL のディスプレイ・リストの機能等を用いて、視点移動の高速化を試行したバージョンである（非公開）。

⑥MultiLang

2006 年頃、インドネシアでの住宅地の将来像を検討するために作成したもので、同じバージョンのままで、表示言語を切り換える機能を有している。言語に依存する部分は、ヘルプ、エラー・メッセージ、及びメニューやボタン等の表示である。このバージョンにおいては、それら全てがテキスト・ファイルとして外部化されているため、翻訳作業によりメモ帳で開くことのできるテキスト・ファイルを用意するだけで、新たな言語上で使用することが可能となった（非公開）。

この他に、1996 年頃に翻訳移植した、ハングル版、インドネシア語版などが存在するが、機能的には他のバージョンに含まれるもののみであり、言語切り替えに関しては、⑥の多言語機能を組み込んだ上で、プラグイン DLL や外部関数にも拡張した Ver.2.09 で対応可能である。

７．グラフィックス処理

７－１．OpenGL の初期化・終了処理と Windows バージョンについて

OpenGL を使用する画面は、メイン画面、平面図を表示するダイアログ（配置、視点設定、視点抽出、形状生成・平面、プラグインのいくつか）における編集画面、マテリアル・カラー編集の色玉、型鋼、道路法面・園路法面のマテリアル編集の小さな四角、グラフィックなマテリアル編集、カラー編集等）がある。これらは全て固定数であるが、その他にまた、景観データベースの検索結果の表示用の小さな窓（検索結果に応じて不定数）がある。

メイン画面、及び編集のために一時的に開く編集画面のいくつかは、その一部に OpenGL 描画のための子ウィンドウを持っている。例えば、配置、平面、視点移動、可視範囲解析などは、オルソ系（平面図表示）の OpenGL 子ウィンドウを有する。OpenGL の描画のための子ウィンドウに関しては、全て、専用のクラス(CWnd の派生クラス)を用意し、これを用いて描画処理を行っている。

OpenGL 画面の描画のためには、デバイス・コンテキスト（DC）と OpenGL 用のレンダリング・コンテキスト(HGLRC)の二つのコンテキストが必要とされる。この内、Windows の描画として OS で用意される DC には、各 Window 固有のデバイス・コンテキストと、共通リソースから一時的に取得されるデバイス・コンテキストの２種類があり、後者の場合、同時に最大５以内という制約がある。共通リソースのデバイス・コンテキストを、ひとつのアプリケーションが DC を取得後、描画が終わった後にも解放せずに保持しつづけると、別のアプリケーションの表示に障害をもたらす。過去のバージョンを振り返ると、WindowsME において、リソースの制約が最も厳しく、描画が終了した後に DC を解放しないと、すぐにリソース不足による描画の異常を生じた。

WindowsXP、2000 においてはプログラム修正を要する変化はなかったが、その後 WindowsVISTA において、DC の管理方法が若干変化したと見られ、描画が本来行われる窓の中だけでなく、デスクトップや他のウィンドウの中に描画が行われる場合が見られた。解放された DC に HGLRC が結び付いたままであると、その DC を取得した別のウィンドウ（例えばデスクトップ）に、想定外の再描画が行われる現象が生じた。

この障害は、DC を開放する前に、DC->ReleaseOutputDC()を実行し、HDC を無効とすることにより解決する。しかし、共通の関数で不特定多数の OpenGL ウィンドウを扱っている景観データベース検索結果表示部分など、一部の OpenGL ウィンドウに関しては、この方法で処理することができなかったため、内部的なビットマップウィンドウをメモリ上に構築し、視点位置や光源やモデルに対する修正が行われた場合にはこのビットマップへの描画を行っておき、外から見えるウィンドウに対しては、OnPaint()関数の中で、ビットマップのコピー(BitBlt())を行う方法で解決した。この方法は、グラフィックス・カードのハードウェアによる高速描画の機能が利用できない可能性がある一方、上記の内容変更が

なく、単に他のウィンドウとの位置関係が変化しただけの理由による再描画は速くなる。

OpenGL ウィンドウを管理するライブラリ関数は、二系統用意してある。一つは、wg3XXX、g3YYY という名称の、視点移動等を伴う多機能な OpenGL ウィンドウを処理するためのライブラリ関数である。もう一つは、wg3sXXX、g3sYYY という名称の、主として画像やカラー等を小さなウィンドウで表示するためのライブラリ関数である。OpenGL ウィンドウは、構築とともにリストに登録され、除却に際してリストから抹消される。wg3、g3 系列と、wg3s、g3s 系列とでは、別のリストで管理している。

各ウィンドウのハンドラ・ルーチン(cpp)は、描画に用いる DC のハンドル HDC をポイントする固有のメンバ変数をもっている。上記のリストには、DC そのものを登録することはできないが、このメンバ変数のアドレスを登録するようになっている。DC は、描画が必要となる都度取得し、このポインタに関連づける。一方、HDLRC は、初期化段階で取得されたものが、このリストに登録され、ウィンドウが除却されるまで保持される。

リスト７－１：標準的な OpenGL ウィンドウの初期化

```
int Cxxx::OnCreate(LPCREATESTRUCT lpCreateStruct){
   m_HDC = new HDC;    //クラスのメンバ変数(ポインタ)にメモリブロックを割り当てる
   pmCDC = GetDC();                              //共有 DC を取得
   *m_HDC = pmCDC->GetSafeHdc();                 //HDC をメモリブロックに格納
   set_pixel(*m_HDC);                            //ピクセル・フォーマットを設定
   m_HGLRC = wglCreateContext(*m_HDC); //OpenGL レンダリングコンテキスト設定
   wg3EntryDrawarea((void*)&m_HDC,(void*)&m_HGLRC);  //設定内容を登録
   status = wg3AssignDrawarea((void*)&m_HDC);  //作業対象をこの表示画面に
   status = g3InitializeWindow();                //初期化処理
/*****************以下、その他の初期化（中略）******************/
   wg3AssignDrawarea(NULL);                      //作業対象をこの表示画面から外す
   pmCDC->ReleaseOutputDC();                     //DC を返す前に、出力との関係を切る
   ReleaseDC(pmCDC);                             //DC を共有プールに返す
   return 0;
}
```

リスト７－２：標準的な OpenGL ウィンドウの除却

```
BOOL Cxxx::DestroyWindow()
{
   // 031106 OnDestroy より移動（理由：OnDestroy では wglMakeCurrent が失敗する
   CClientDC dc(this);
   *m_HDC = dc.m_hDC;
   int rc = wg3AssignDrawarea((void*)&m_HDC);
   if(!rc) z3Message( 1383, "¥nCxxx::OnDestroy()" );

   wg3DeleteDrawarea((void*)&m_HDC);
   wglDeleteContext(m_HGLRC);

   return CWnd::DestroyWindow();
}

void Cxxx::OnDestroy(){
```

```
   CWnd::OnDestroy();
   delete m_HDC;
}
```

リスト７－３：標準的な OnPaint コールバック

```
void CDrawFrm::OnPaint()
{
   CPaintDC dc(this); // 描画用のデバイス コンテキスト。このスコープの中でのみ有効
   *m_HDC = dc.m_hDC;
   if(!wg3AssignDrawarea((void*)&m_HDC)) z3Message(5377);
   if(!wg3Redraw((void*)&m_HDC)) z3Message(1425);
   wg3AssignDrawarea(NULL);
}
```

リスト７－４：メモリ・デバイスを用いる場合

（メモリ・デバイス上に描画しておき、表示内容、視点あるいはウィンドウサイズ等に変更がない OnPaint では、画面の無効領域にメモリ・デバイスから画像コピーするだけとする構成）

```
int CMyOrthoVW::OnCreate(LPCREATESTRUCT lpCreateStruct)
{
   if (CWnd::OnCreate(lpCreateStruct) == -1)   return -1;

   // TODO: この位置に特殊な生成用のコードを追加してください。
   CPaintDC dc(this);
   mDC.CreateCompatibleDC(&dc);
   mBM.CreateCompatibleBitmap(&dc,1,1); //サイズ１×１のビットマップを仮に作る
   mDC.SelectObject(mBM);

   m_sHDC = new HDC;
   *m_sHDC = mDC.m_hDC;//dsb.
   zet_pixel(*m_sHDC);
   m_sHGLRC = wglCreateContext(*m_sHDC);
   wg3EntryDrawarea((void*)&m_sHDC,(void*)&m_sHGLRC);
   wg3AssignDrawarea((void*)&m_sHDC);
   g3InitializeWindow();
   return 0;
}

BOOL CMyOrthoVW::DestroyWindow()
{
   // TODO: この位置に固有の処理を追加するか、または基本クラスを呼び出してください
   TRACE("CMyOrthoVW::DestroyWindow() ¥n");
   CClientDC dc(this);
   dc.AssertValid();
   *m_sHDC = dc.m_hDC;//ME マシンで、次行で失敗する。
   if(!wg3AssignDrawarea((void*)&m_sHDC)) z3Message(5000,"Ortho::OnDestroy1");
   g3ClearHilight();
   g3SetLightGroup(NULL);//引数 NULL で、メモリの解放を行う。
   g3SetCameraPers(NULL);//同上
   g3ReleaseGroundPixel();
   wg3DeleteDrawarea((void*)&m_sHDC);
   wglDeleteContext(m_sHGLRC);
```

```
   return CWnd::DestroyWindow();
}

void CMyOrthoVW::OnDestroy()
{
   CWnd::OnDestroy();
   // TODO: この位置にメッセージ ハンドラのコードを追加してください。
   delete m_sHDC;
}

void CMyOrthoVW::OnSize(UINT nType, int cx, int cy)
{
   CWnd::OnSize(nType, cx, cy);

   // TODO: この位置にメッセージ ハンドラのコードを追加してください。

   CClientDC clientDC(this);
   *m_sHDC = clientDC.m_hDC;//dsb.
   if(cx == 0 || cy == 0) {
      ;
   } else {
      wg3AssignDrawarea((void*)&m_sHDC);
      g3Resize( cx, cy );
   }
   mBM.DeleteObject(); //画面サイズが変わった場合には、メモリビットマップを更新
   mBM.CreateCompatibleBitmap(&clientDC,cx,cy);
   mDC.SelectObject(mBM);
   RedrawWindow();//新しい空のビットマップに再描画
}

void CMyOrthoVW::OnPaint()
{
   TRACE("MyOrthoVW::OnPaint[%d]¥n",giFirst);
   //081004 合理化実験
   RECT Rect;
   if(1){
      int rusak;
      rusak = GetUpdateRect(NULL,FALSE);
      if(!rusak){//ここには来ない
         MessageBeep(MB_ICONASTERISK);//0x40 ピンポン
         return;
      }else{//被さっているウィンドウが静止した場合
//          MessageBeep(MB_OK);//0:トン
         rusak = GetUpdateRect(&Rect,FALSE);
      }
   }

   CPaintDC dc(this); // 描画用のデバイス コンテキスト。
   //CPaintDC の構築は、BeginPaint を意味する
```

```
   // TODO: この位置にメッセージ ハンドラのコードを追加してください。

   *m_sHDC = dc.m_hDC;//dsb.

   int            GrpCnt;
   s3LightGroup   *lg;
   int            status, kind;
   g3Ortho        ortho;
   int            val;
   int            sts1, sts2;
   int            MATATETAP=-1;
   float          time;

//          giFirst が-1 のとき、常にオルソ全体視界の初期化を行い、１に変更する。。
//          giFirst が０のとき、前回最後の状態に初期化を行い、１に変更する。
//          giFirst が１のとき、視点の初期化を行わず、通常の GL 描画を行う。
//          giFirst が２のとき、無効領域を補修する
   if(giFirst == -1) {//初期化を必要とする場合
        /**************固有の初期化を行う(略)***************/
        giFirst = 1; //再描画を指定
     }
   }   // giFirst<0  最初に平面描画を行った場合の初期化はここまで。

   wg3AssignDrawarea((void*)&m_sHDC);

   if(giFirst == 0){
     g3SetCameraOrtho(kindsave,&orthosave);
     giFirst = 1;
     // 20000504 DR.H.K.メイン側から光源/時間を取得する必要がある
     wg3AssignDrawarea((void*)&m_pView->m_drawFrm->m_HDC);
     g3GetLightGroup(&lg);
     g3GetTime(&time);
     wg3AssignDrawarea((void*)&m_sHDC);
     g3SetLightGroup(lg);
     g3SetTime(time);
   } else{
     g3GetCameraOrtho(&kindsave,&orthosave);//視点移動等の結果を取得・保存
   }

   if(giFirst == 1){ //表示内容を、メモリ・デバイス上に再描画する
     int x,y;
     m_pParkRoadWnd->BringWindowToTop();
     int rc = wg3Redraw((void*)&m_sHDC);
     g3GetSize(&x,&y);
     rc = dc.BitBlt(0,0,x,y,&mDC,0,0,SRCCOPY); //表示画面全体に反映する
     giFirst = 2;
   }else{//giFirst == 2; メモリ・デバイス上の描画内容を修正する必要なし
     int x,y,top,bottom,left,right,width,height,rc;
     g3GetSize(&x,&y);
     top = Rect.top, bottom = Rect.bottom;
```

```
      left = Rect.left, right = Rect.right;
      width = right - left;
      height = bottom - top;
      rc = dc.BitBlt(left,top,width,height,
         &mDC,left,top,SRCCOPY);  //表示画面の内、無効領域のみを修復
      if(!rc) MessageBeep(MB_ICONHAND);//ジャン
   }
} //OnPaint selesai

//編集・視点移動等の後、RedrawWindow()を実行する
BOOL CMyOrthoVW::RedrawWindow
(LPCRECT lpRectUpdate,CRgn* prgnUpdate, UINT flags){
   if( 1 < giFirst ) giFirst = 1; //メモリ・デバイス上に再描画するフラグを設定する
   return CWnd::RedrawWindow(lpRectUpdate, prgnUpdate,   flags);
};
```

メモリ・デバイスを用いた処理では、表示内容の更新の有無に関してフラグを使用し、地物の編集、視点移動、画面サイズ変更が行われた場合には、メモリ・デバイス上への再描画を行った上で、表示画面に画像をコピーする。表示内容の更新がない、単なるウィンドウの無効領域の更新（例えば上に被さっていた別のウィンドウが移動したような場合）に関しては、メモリ・デバイスから表示画面への画像コピー（無効領域を含む最小限の領域のみ）を行うのみである。

更に、一つのクラスで多数の OpenGL ウィンドウの制御を一括して行っている CChildFrm クラスにおいては、メモリ・デバイスのみを保持し、OnPaint の中で OpenGL コンテキストを生成し、描画し、メモリ・デバイスにイメージを残すだけで、OpenGL コンテキストを除却して処理を終了している。

リスト７－５：多数の OpenGL 子ウィンドウを一括処理する場合

```
void CChildFrm::OnPaint()
{
        int status;
        void zet_pixel(HDC);

        CPaintDC dc(this); // 描画用のデバイス コンテキスト。
        if(!GLFLAG){ //無効領域の再描画の場合、画像をコピーするのみ
                RECT rect;
                rect = dc.m_ps.rcPaint;
                dc.BitBlt(rect.left,rect.top,rect.right-rect.left,rect.bottom-rect.top,
                        &mDC,rect.left,rect.top,SRCCOPY);
                return;
        }
        dbImageData *img = NULL;
        dbDataBase*     pDBtab = dbGetDataBaseAddr();
        int DBno = dbGetDB();
        int recNo,recHeadNo;
        int rc;
        CRect clientrect;
        GetClientRect( &clientrect );
```

```
        /*-----------------データベースより選択画像の情報取得----------------*/
         （中略）
        /*--------------------ＤＣ処理-------------------------*/
        m_hDC = new HDC;
        mBM.DeleteObject();
        mBM.CreateCompatibleBitmap(&dc,m_size.cx,m_size.cy);
        mDC.SelectObject(mBM);
        *m_hDC = mDC.m_hDC;//dsb.
        zet_pixel(*m_hDC);
        m_hGLRC = wglCreateContext(*m_hDC);
        wg3EntryDrawarea((void*)&m_hDC, (void*)&m_hGLRC);
        wg3AssignDrawarea((void*)&m_hDC);
        status = g3InitializeWindow();//980520 DR.H.K. これはＧＬの初期化
        /*--------------------画像の設定-------------------------*/
         （中略）
        /*--------------------縮小イメージの作成-------------*/
         （中略）
        /*---------------------表示処理---------------------*/
        g3GetBackImage(&old_img);/*990930 DR.H.K.*/
        if(old_img) d3Free(old_img);
        g3SetBackImage(sml_img);
        wg3Redraw((void*)&m_hDC);
Painted: //以下終了処理。OpenGL のコンテキストを除却する
        d3Free( sml_img );
        g3SetBackImage( NULL );
        rc = wg3AssignDrawarea( NULL );
        wg3DeleteDrawarea( &m_hDC );
        rc = wglDeleteContext( m_hGLRC );
        m_hGLRC = wglGetCurrentContext();//確認のため、取得して見る。(NULL)
        status = dc.BitBlt(0,0,m_size.cx,m_size.cy,&mDC,0,0,SRCCOPY);
        GLFLAG = 0;
        delete m_hDC;
}
```

Windows7 では、描画処理終了後、wg3AssignDrawarea(NULL)を実行していない個所で障害を生じた。なお、OS のバージョンによる違いについて、第 13 章も参照されたい。

旧版に見られた各ウィンドウ作成に際してのプログラマによる不統一も調整した。

（1）ウィンドウの重なり

ウィンドウには親子関係が存在する。例えば、一つのダイアログを構成するボタンやエディットボックス等は、子ウィンドウとして定義されている。子ウィンドウは、親ウィンドウをアクティブにしても、その裏側に隠れることができない。従って、ある編集画面を新たに追加する場合に、メイン画面の子ウィンドウではない、メイン画面と対等のウィンドウとして作成する必要がある。このことを明示的に行うためには、すべてのウィンドウの親ウィンドウである、デスクトップ・ウィンドウの子ウィンドウとして作成する。

（2）ウィンドウのアイコン

ウィンドウの左上にユニークなアイコンを付けるためには、ウィンドウ・クラスを構築

する際に、SetIcon を実行する。

上の（1）と（2）のプログラム例を下に示す。

リスト７－６：ウィンドウの重なりとアイコンの処理例

```
CDispKeinen::CDispKeinen(CWnd* pParent /*=NULL*/)
        : CDialog(CDispKeinen::IDD, pParent)
{
        //{{AFX_DATA_INIT(CDispKeinen)
        m_date = 0.0f;
        //}}AFX_DATA_INIT
        BOOL rc = Create(IDD,GetDesktopWindow());    //デスクトップを親に指定
        SetIcon(AfxGetApp()->LoadIcon(MAKEINTRESOURCE(IDI_SIM)),1);
                                    //ウィンドウの左上隅に表示するアイコンを指定する
}
```

７－２．メイン画面の表示処理

メイン画面の表示処理は、CDrawFrm::OnPaint() 関数の中で起動される wg3Redraw() (wg3.c)が入口となって実行される。

wg3Redraw() 関数は、g3draw() (g3drl.c)を呼び出し、この関数の中で描画を行う。

リスト７－７：wg3Redraw 関数

```
int wg3Redraw(void*w)
{
  g3Clear();                    表示のリセット
  g3Draw();                     再描画
  wg3Swapbuffers(w);            ダブル・バッファを表示に反映させる
  return(1);
}
```

g3Draw() 関数は、以下の処理を行っている。

リスト７－８：g3Draw 関数

```
void g3Draw()
{
  draw3dObjects();              三次元オブジェクトの表示
  drawCameramark();             カメラ記号（視点位置を示す）の表示
  drawMeasure();                縮尺の表示
  drawRect();                   指定範囲を示す長方形の表示
  glFlush();                    表示操作
}
```

draw3dObjects() 関数は、以下の処理を行っている。

リスト７－９：draw3dObjects 関数

```
static void draw3dObjects()
{
  int jitter;
  jitter_point *jb;

  cameraProjection();
  cameraViewing();
```

```
  if(da[cd].draws[G3_DRA_LIGHT]){                    光源が定義されている場合、設定
    light();
    lightEnable();
  }
  if(da[cd].draws[G3_DRA_ANTIALIAS] && da[cd].aaCount > 0){ アンチエリアシング
    jb = getJitter(da[cd].aaCount);
    glClear(GL_ACCUM_BUFFER_BIT);
    for (jitter = 0; jitter < da[cd].aaCount; jitter++) {
      glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
      if(da[cd].backImage && da[cd].draws[G3_DRA_BACKIMAGE]){ /*1996.5.10*/
        drawImage(da[cd].backImage->width,
                  da[cd].backImage->height,
                  da[cd].backImage->pixels);
      }
      accCamera(jb[jitter].x, jb[jitter].y);
      cameraViewing();
      display3dObjects();
      if(da[cd].frontImage && da[cd].draws[G3_DRA_FRONTIMAGE]){ /*1996.5.10*/
        drawImage(da[cd].frontImage->width,
                  da[cd].frontImage->height,
                  da[cd].frontImage->pixels);
      }
      glAccum(GL_ACCUM, 1.0/(GLfloat)da[cd].aaCount);
    }
    glAccum(GL_RETURN, 1.0);
  }else{                                             通常の場合
    if(da[cd].backImage && da[cd].draws[G3_DRA_BACKIMAGE]){ /*1996.5.10*/
      drawImage(da[cd].backImage->width,          背景があれば表示する
                da[cd].backImage->height,
                da[cd].backImage->pixels);
    }
    display3dObjects();                              三次元要素を表示する
    if(da[cd].frontImage && da[cd].draws[G3_DRA_FRONTIMAGE]){ /*1996.5.10*/
      drawImage(da[cd].frontImage->width,          前景があれば表示する
                da[cd].frontImage->height,
                da[cd].frontImage->pixels);
    }
  }
}
```

display3dObjects() 関数は、以下の処理を行っている。

リスト７－１０：display3dObjects 関数

```
static void display3dObjects()
{
  drawAxis();                                        座標軸の表示
  drawGrid();                                        グリッドの表示
  if(da[cd].draws[G3_DRA_BUMI] == TRUE) drawAllGround();      地面のみ表示
  else {
        drawAllGroupEtc();                           全てのグループを表示
        if(g3GetDrawable(G3_DRA_BUMITEX)){   地面だけテクスチャ表示する場合
              if(!g3GetDrawable(G3_DRA_TEXTURE)){
```

```
                                int ingat;
                                ingat = g3GetDrawable(G3_DRA_WIRE);
                                if(ingat) g3Disable(G3_DRA_WIRE);
                                g3Enable(G3_DRA_TEXTURE);
                                drawAllGround();
                                g3Disable(G3_DRA_TEXTURE);
                                if(ingat) g3Enable(G3_DRA_WIRE);
                        }else;
                }else;
    }
    drawHilight(); /*1996.3.1*/                 選択されたオブジェクトの強調表示
    drawOrbit();                                軌道の表示
    drawVisual(); /*daerah terlihat*/           可視範囲の表示
}
```

drawAllGroupEtc() 関数は、以下の処理を行っている。

リスト７－１１：drawAllGroupEtc 関数

```
drawAllGroupEtc(){
    if( g3GetDrawable(G3_DRA_SHADOW)){
        g3MakeShadowVolume();               影の場合、影立体を作成する
        drawAllGroupAndShadow();            影付きで全グループ表示を行う
    }else{
        drawAllGroup();                     通常の全グループ表示
    }
}
```

drawGroup() 関数は、以下の処理を行っている。

リスト７－１２：drawGroup 関数

```
static int materialsekarang=0, tekturesekarang=0;
      マテリアルとテクスチャの現在の設定状況を保持するスタティック変数

static void drawGroup(d3Group *g)
{
    d3Face *f;
    d3Link *l;
    d3Group *child;
    d3Matrix linkMat;
    int tree3;
/*980508 DR.H.K.*/
    int bahaningat;   /*マテリアルとテクスチャを一時退避するオート変数*/
    int teksilingat;
    int bahan;          /*当面のマテリアルとテクスチャ*/
    int teksil;
/*dsb*/

    if(!g) return;
    tree3 = i3IsAttribute(g,I3_ATTR_TREE3);

/*マテリアルとテクスチャの設定*/
    bahan = d3GetGroupMaterial(g);          グループに定義されたマテリアルを取得
    teksil = d3GetGroupTexture(g);          グループに定義されたテクスチャを取得
```

```
   if(bahan || teksil ||g->lup==NULL   ){
          /*両方ともない場合、明示的にないのではなく、単に未定義と考える*/
          /*ルートグループで無い場合、明示的にないことにする*/
     if(!da[cd].draws[G3_DRA_WIRE]){
       setMaterialTexture(bahan,teksil);
/*この中で、マテリアルにテクスチャがあれば teksilsekarang はそれに変更される*/
     }else{
       setMaterialTextureLine(bahan,teksil);
     }
     bahaningat = materialsekarang;        /*auto 変数をスタックとして用いる*/
     teksilingat = texturesekarang;        /*面、子グループの処理後、戻すため*/
   if(bahan+teksil){ /*グループにいずれかが定義されていた場合、それを設定する*/
     materialsekarang = bahan;
     texturesekarang = teksil;
   }/*この処理は、そのグループにマテリアルかテクスチャの
    いずれかが定義されていれば、親から継承せずに独自の属性を用い、
    かつその属性を面と下位グループに継承する、というロジックを具体化する*/

/*マテリアルとテクスチャの設定完了*/

   以下、グループに属する面を表示する
   for(f=NULL;f=d3GetGroupFace(g,f);){
     int hasil;
     d3Group *Gsakit;
       if(hasil=drawFace(f,tree3)){
         for( Gsakit=g; Gsakit->lup; Gsakit = Gsakit->lup->up->parent){
           if(Gsakit->type == I3_GTYPE_FILE){
             z3Message( 5372, Gsakit->kind );
             break;
           }
         }
      }
   }
/*  以下、子グループを表示*/
   for(l=NULL;l=d3GetGroupDLink(g,l);){
     child = d3GetLinkDGroup(l);
     d3GetLinkMatrix(l,linkMat);
     if(berapa(linkMat)){
          glPushMatrix();
          glMultMatrixd(linkMat);
          drawGroup(child);
          glPopMatrix();
     }else{
          drawGroup(child);
     }
   }
/*待避したマテリアル、テクスチャ情報の復帰*/
   if(!da[cd].draws[G3_DRA_WIRE]){
    /*テクスチャ表示およびシェーディング表示の場合、処理前の状態に戻す*/
     setMaterialTexture(bahaningat,teksilingat);
```

```
  }else{
    setMaterialTextureLine(bahaningat,teksilingat);
  }
  materialsekarang = bahaningat; /*親から継承されたものに戻す*/
  texturesekarang = teksilingat;
}
```

グループには、マテリアル、及びテクスチャの情報を付けることができる。この情報がグループに定義されていない場合には、親グループの属性が継承される。

また、グループにマテリアルやテクスチャの情報が無い場合であっても、そのグループに直接属する面にマテリアル、テクスチャ、またはカラーの情報を付けることができる。

７－３．面の表示処理

面の表示は、drawFace(*F)関数によって実行される。

面が凹ポリゴン(D3_SHP_CONCAVE)のフラグを有し、頂点が４以上あって、表示モードがワイヤーフレーム以外（テクスチャ、シェーディング）の場合には、面を三角形に分割して、drawMuka() 関数を通じて、OpenGL に出力する。

凹ポリゴンのフラグが無い場合には、そのまま drawMuka()関数に渡される。

CONCAVE(ON); の状態においては、凹ポリゴンの検査を行い、drawMuka(*F)関数で、凹ポリゴンを正しく表示する。

CONCAVE(OFF); の状態においては、上記の検査は省略され、無責任モードで表示が行われる。凹ポリゴンは正しく表示されない。

マテリアル、テクスチャ、カラー

マテリアル、およびテクスチャは、グループに対して定義することができる。

グループに帰属する面に対して、個々に定義することもできる。その場合、グループに対して定義されたマテリアルもしくはテクスチャは、デフォルト値のように扱われ、別の値が定義された面はそれを用いて表示するのに対して、未定義の面に関してはグループに定義された値を表示に用いる。

この他、面に対してはカラーを定義することができる。カラーは、頂点に対しても定義することができ、面に定義されたカラーは、上記と同様に、頂点に定義されたカラーに対するデフォルト値のように参照される。

同一の面に対して、マテリアルとカラーが同時に定義されていて、マテリアルにカラー情報が含まれている場合、カラー設定値が上書きされる。

最終的な表示においては、OpenGL の表示系に対しては、

Ambient[4]

Diffuse[4]

Specular[4]

Shininess[1]

Emission[4]

マテリアルで定義されている属性は、最終的な OpenGL での表示に使用される属性に対応したものとなっている。

リスト７－１３：dbMaterial 構造体定義

```
Typedef struct _dbMaterial{
  float ambient[4];        環境光との関係で表示色を決定する物体色
  float diffuse[4];        拡散光との関係で表示色を決定する物体色
  float specular[4];       反射光との関係で表示色を決定する物体色
  float shniness;          鏡面指数
  float emission[4];       物体独自の放射光
  char *texture;           マテリアルの中で定義されるテクスチャ名称
}
```

カラー(r,g,b,a)が設定されている場合には、その値を、ambient 及び diffuse に反映する。その際に、ambient は、r,g,b,a 値に DB_AMBIENT_RATE (dbms.h で 0.3f と定義)を乗じた値を与え、diffuse は r,g,b,a 値をそのまま代入している。

マテリアルの属性は、マテリアル・ファイルに保存されている。この中では、以下の項目を定義している。

Lab：カラー値（LAB 表色系で表現　●実装されていない）

RGB: カラー値（diffuse[0-2]に値を代入）

SPECULAR: 反射率（specular[0-2]に共通値を代入, specular[3]は 1.0）

EMISSION: 輝度(diffuse[0-2]に共通値を乗じた値を emission[0-2]に代入)

TRANSPARENCY: 透明度（diffuse[3]に値を代入）

TEXTURE: テクスチャ(テクスチャ名称を取得し代入)

７－４．ステレオ表示機能

人間の目は、三次元の地物の透視図を網膜において二次元画像として入力している。その際に、左右の目の視点位置の違いによる画像の差異を認識することにより遠近感を得ている。景観シミュレータにおいてこのことを再現するためには、左右の目に対応する、異なる視点からの画像を同時に生成するプログラムを作成すると共に、二つの画像を、ユーザーの左右の目に分離して送り込むためのハードウェアを必要としている。

ヘッドマウンド・ディスプレイのように、左右の目のために二つのディスプレイ装置を用意するのが最も直接的である。しかし、機材のは高価であり、また複数のユーザーが同時に同じ映像を見て討論するためには多くの機材を用意しなければならない。

そこで、プロジェクタによりスクリーンに投影された画像、またはモニタ画面に表示された画像を時分割または面分割し、異なる画像を左右の目に出力する方法が多く採用されている。時分割するためには、左右の画像を交互に表示すると共に、シャッター機能のあるメガネをユーザーに装着してもらい、表示のリフレッシュとシンクロして、左右の目の

シャッターを交互に開くような装置を使用する。一部のゲームマシン等に用いられている。もう一つは、画面に左右の画像で異なる偏光をかけ、ユーザーは、左右で異なる向きに取り付けられた偏光フィルタを装着してもらい、これを眺める方法である。

プロジェクタの場合には、２台のプロジェクタを用意しておき、それぞれに、異なる向きの偏光フィルタを取り付けることで同一のスクリーンに重複した画像を投影する。偏光メガネをかけてこれを眺めることにより、画像を分離する。スクリーン面で反射する際に偏光が変わることを避けるために、立体投影のためには通常のスクリーンではなく、アルミ粉などを用いた銀色塗料を施したスクリーンを使用する。このための偏光フィルタを用いたメガネは、フレームに紙などの安価な材料を用いることにより配布可能な単価で製造できるため、この方法はイベント会場などでよく用いられている。

カラー表示の必要がなければ、例えば青色フィルタと赤色フィルタを用いた紙メガネを観客に配り、これで眺めると左右が分離するような配色で、例えば山岳の空中写真を重複印刷したパネルを展示するようなイベントは古くから行われていた。

偏光フィルタによる紙メガネを用いてステレオ表示を行うためにはいくつかの方法がある。複数のプロジェクタを用いてスクリーンに投影する方法は、仮想現実としてしばしば採用されている。複数のコンピュータ上に複数のプログラムを動作させ、左右の画像がシンクロするように視点移動を行う方法が、最も処理速度の高いハイエンドの方法である。この左右プロジェクタの組を正面のみならず左右上下のスクリーンにも適用することにより全方位の立体画像を生成する、没入感の高い仮想現実装置が製品化されている。

景観シミュレータにおいては、ステレオ表示を、①画面走査線の奇数ラインと偶数ラインに右目視点と左目視点の画像を作成すること、②画面マトリクスの行単位でストライプ状に偏光の向きを替えるマイクロポール・シートを組み込んだ液晶ディスプレイにより、奇数ラインと偶数ラインに異なる偏光を与えること、③偏光メガネにより、右目に奇数ライン、左目に偶数ラインだけを見せること、の３点により実現している。

奇数ラインと偶数ラインに異なる視点位置の画像を表示するために、g3Draw()関数の中で、分離する処理を行っている。

リスト７－１４：ステレオ表示を行わない g3Draw 関数

```
void g3Draw()
{
  if(cd<0) return;
  draw3dObjects();
  drawCameramark();
  drawMeasure();
  drawRect();
  glFlush();
}
```

表示モードが「パース」で、StereoMode が NULL でない場合に、ステレオ表示を行い、それ以外の場合では、従来と同様の表示を行う。

リスト７－１５：条件によりステレオ表示を行う g3Draw 関数

```
void g3Draw()
{
  static unsigned char MaskBitmap
```

```
                                    [MAX_SCREEN_WIDTH* MAX_SCREEN_HEIGHT / 8 ];
static int MaskWidth = 0;
static s3Camera PersSave;          // Pers infomation save area from da[cd].pers

static s3Camera rightPers, leftPers;              // perspective information area
s3Camera *prightPers=NULL,*pleftPers=NULL,*pPersSave=NULL;
GLint i;
GLfloat    map[] = { 0.0, 1.0 };

int       w = 1600;
FILE    *fp=NULL;

if( cd < 0 )    return;

//w = ( ( da[cd].width + 15 ) >> 4 ) << 4;
w = ( ( da[cd].width + 31 ) >> 5 ) << 5;
if( w != MaskWidth )
{
   for( i = 0; i < MAX_SCREEN_HEIGHT; i += 2 )
      memset( MaskBitmap + w / 8 * i, 0xff, w / 8 );
   for( i = 1; i < MAX_SCREEN_HEIGHT; i += 2 )
      memset( MaskBitmap + w / 8 * i, 0x00, w / 8 );
}

if(da[cd].cameraKind == G3_CAM_PERS && StereoMode)    // Check stereo mode
{
   PersSave = da[cd].pers;    // Save current perspective information

   pPersSave=&PersSave;
   pleftPers=&leftPers;
   prightPers=&rightPers;
            //Note:For standard C could not return by &,so changed it to *
   g3GetStereoPers( StereoEye, pPersSave, pleftPers, prightPers );


   glPixelMapfv( GL_PIXEL_MAP_S_TO_S, 2, map );
   glPixelMapfv( GL_PIXEL_MAP_I_TO_R, 2, map );
   glPixelMapfv( GL_PIXEL_MAP_I_TO_G, 2, map );
   glPixelMapfv( GL_PIXEL_MAP_I_TO_B, 2, map );
   glPixelMapfv( GL_PIXEL_MAP_I_TO_A, 2, map );

   glEnable( GL_STENCIL_TEST );
   glClearColor( 0.0f, 0.0f, 0.0f, 0.0f );
   glClear( GL_STENCIL_BUFFER_BIT );

   glMatrixMode( GL_MODELVIEW );
   glLoadIdentity( );
   glMatrixMode( GL_PROJECTION );
   glLoadIdentity( );

   glRasterPos2i( -1, -1 );
   glDrawPixels(w,da[cd].height, GL_STENCIL_INDEX, GL_BITMAP, MaskBitmap );

   glStencilOp(GL_KEEP, GL_KEEP, GL_KEEP);
   glStencilMask(0x7fff); // investigating--prohibit wrinting first bit

   glStencilFunc( StereoSwap ? GL_EQUAL : GL_NOTEQUAL, 0x0, 0x1);
   da[cd].pers = rightPers;
   draw3dObjects();            // draw objects to odd lines
   drawCameramark();          // draw cameramark to odd lines of back buffer
   drawMeasure();              // draw measure to odd lines of back buffer
   drawRect();                 // draw rect to odd lines of back buffer

   glStencilFunc( StereoSwap ? GL_NOTEQUAL : GL_EQUAL, 0x0, 0x1);
   da[cd].pers = leftPers;
   draw3dObjects();
   drawCameramark();
   drawMeasure();
```

```
      drawRect();
      glFlush();

      glDisable( GL_STENCIL_TEST );
      da[cd].pers = PersSave; // Restore perspective information
    }
    else
    {
//    Original g3Draw is as follows
      draw3dObjects();
      drawCameramark();
      drawMeasure();
      drawRect();
      glFlush();
    }
}
```

ステレオ表示における右目・左目の視点位置を計算する、g3GetStereoPers 関数が、新たに追加されたソース StereoSight.c に定義されている。

左右の目のために共通の表示画面を用いるため、左右の目と注視点を結ぶ線は平行ではなく、また表示画面と斜交する。このため、左右の目の事成る視点位置の違いを表示に反映させるために、単に視点位置を左右にずらすのではなく、全体を glTanslated 関数で平行移動した上で、注視点付近が画面の同じ位置となるように表示範囲を変えるために glFrustum 関数を用いて左右に「アオリ」をかけている(g3drl.c の cameraProjection 関数)。

スクリーンで、ステレオ表示を行っている景観シミュレータのウィンドウが縦方向に移動された場合には、偏光フィルタとの関係において奇数行と偶数行に対する左右の画像の出力を反転しなければならない。従来の StereoSwap フラグは、スクリーン上のウィンドウ位置に無関係に固定的に左右画像と奇数ライン・偶数ラインを関係付けるものであったが、Ver.2.09 においては、表示画面（その最上行）が現在存在する位置が、画面全体の中の奇数行か偶数行かを検出した上で、それに対応したステレオ画像を生成・出力するように改良した。従来のバージョンでは、画面サイズの変更が無い移動の場合には、他のウィンドウとのオーバーラップの変化などが無い限り、OpenGL の再描画を省略していたが、ステレオ表示で上記の対応を実現するためには、移動に際しても再描画を行うこととした。但し、移動の途中で、OnMove イベントが発生する度に再描画を行うと、画面操作自体の粘度が非常に高くなるため、OnMove イベントの際に 0.2 秒のタイマーをセットし、OnTimer イベントの中で再描画要求を行っている。移動操作が継続している間は、タイマーが再セットされるため、OnTimer イベントは発生しない。この処理は、マテリアル編集画面でスライドバーによりカラー編集を行う場合に、選択したオブジェクトに選択したカラーを適用し、メイン画面を再描画する処理とほぼ同様である。

リスト７－１６：ユーザーによるステレオ表示ウィンドウの移動への対応

① CmainFrame::OnMove,　OnTimer
② CdrawFrm::OnPaint
③ G3 ライブラリ G3drl.c の g3draw()関数、FromFirst 変数

このようにして作成した左右の目のための画像を、左右それぞれの目に相互干渉することなく出力するための具体的製品においても、近年技術開発・社会普及が進みつつある。

ステレオ表示機能を開発した 2001 年当時には、市販の液晶ディスプレイの表面に特殊な偏光フィルタを貼り付けて、奇数行と偶数行の偏光の向きを変える方法により改造された二次的な製品と、液晶プロジェクタの内部に同様の偏光フィルタを追加した製品が存在していた。本解説書をとりまとめた 2009 年時点では、既にステレオ画面を放送するサービスも開始されており、前者の液晶ディスプレイに関しては、製造段階で偏光フィルタを最初から組み込んだ製品が販売されている。一方液晶プロジェクタに関しては、偏光フィルタを組み込む方法は発展せず、製造中止となっている。これに代わって、二つのプロジェクタに単純な偏光フィルタを装着し、同一のスクリーンに重ねて投影する方法が一般的となっている。複数の sim.exe の視点移動を、立体視のためにシンクロさせるような機構、あるいは OpenGL によるステレオ表示機能を提供するハードウェア（グラフィックス・チップセット）に対応した拡張などは、簡単に行えるため、今後、ステレオ表示機能を含め、表示機能を高度化するためのプラグイン DLL を開発することの価値があると考えられる。

７－５．影の表示

7-5-1.動作原理

光源から見て、影の原因となる物体から、影を落とす対象までの間の部分（影の立体）を求める。この影の立体の側面（複数の四角形）をレンダリングする際に、ステンシルバッファを用いて、各ピクセルに描かれる側面に関して表と裏のカウントを行う。

このカウントが正となる（表側が裏側よりも一つ多い）ピクセルは、影の部分である。

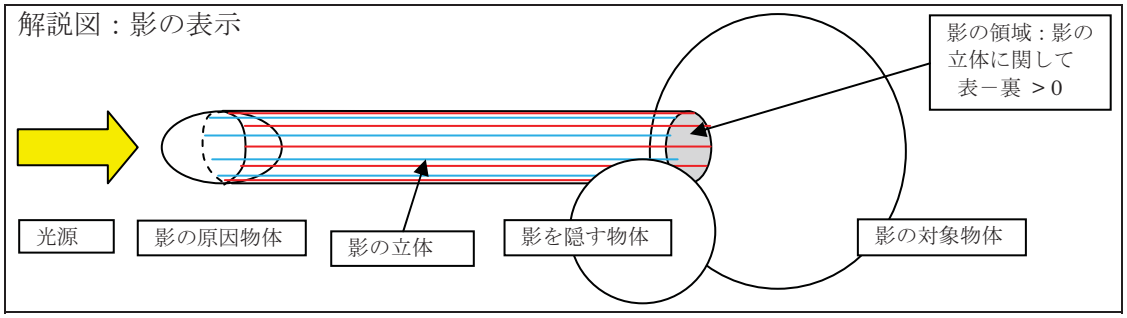


図７－１：影の表示

7-5-2. オペレーション

メイン画面のメニュー[表示][影]の下に、［影なし］［地面以外の物体（通常）］［全物体］［選択した物体］の４選択肢がある。また、［設定］により、影の長さを指定することができる。

関連するプログラムの構成

ShadowDlg.cpp において、影の長さを指定するパラメータの設定ダイアログを処理する。

Mainfrm.cpp　において、影に関するメニュー選択、及びパラメータ設定ダイアログを開く処理等を行っている。

DrawFrm.cpp において、現在選択されている影表示モードに従った表示を行う。

g3drl.c の中で、影に対応した表示処理を行っている。

新たに追加した関数は、

g3MakeShadowVolume() 影の立体を求め、ディスプレイリストに格納する。

drawAllGroupAndShadow() 全グループと、影を表示する。

である。

また、既存の関数に増補し、フラグにより影の機能を選択的に実行している。

drawFace()

MakeShadowFlg のフラグが立っていた場合（g3MakeShadowVolume()からの起動）、影の立体の側面を生成し、ディスプレイリストに追加する。

display3dObjects()

影表示モードの場合には、drawAllGroup の代わりに drawAllGroupAndShadow を実行する。

なお、影の表示段階では、ステンシルバッファを使用しているため、ステレオ表示と共存することはできない。

ステンシルバッファに関しては以下のような OpenGL 関数を使用している。

glStencilFunc(GLenum func, GLint ref, GLuint mask)

func：評価方法

GL_NEVER, GL_LESS, GL_LEQUAL, GL_GREATER, GL_GEQUAL, GL_EQUAL, GL_NOTEQUAL, and GL_ALWAYS.

初期値は GL_ALWAYS

ref：評価基準値

ステンシルバッファの値を、この基準値と比較評価する。評価が成功した場合にのみ、描画が行われる。

mask：マスク

ステンシルバッファのマスクした値に関して、評価を行う。初期値は 1

glStencilOp(GLenum fail, GLenum zfail, GLenum zpass)

評価結果の各場合につき、ステンシルバッファに結果をどう反映させるかを指定する。（ステンシル・テスト失敗、デプス・テスト失敗、成功のそれぞれの場合）

指定できる反映方法の選択肢は以下の通り：

GL_KEEP, GL_ZERO, GL_REPLACE, GL_INCR, GL_DECR, and GL_INVERT.

初期値は、 GL_KEEP

これらを用いて、以下の工程で影の表示を実現している。

①まず全ての地物を描画し、デプスバッファを構築する

②影の立体の側面の内、視点から見た表面を描画し、描画が成立したピクセルに関して、ステンシルバッファの値に 1 を足す

③影の立体の側面の内、視点から見た裏面を描画し、描画が成立したピクセルに関して、

ステンシルバッファの値から１を減じる

この処理により、各ピクセルについて、ステンシルバッファの値が１以上となる領域が、影となる地物の表面に相当するピクセルの領域として得られる。なお、影の立体の描画に際してはデプスバッファを変更しない。

④光源を消灯し、影となる領域について、全ての地物を描画する。

メニュー選択で、主光源だけを消すか、全光源を消すかを指定できる。

⑤光源を点灯し、影以外の領域について、全ての物体を描画する。

実際の処理においては、処理速度を向上するために、計算処理のための描画に際して、不必要な表示画面への反映は抑止すると共に、マテリアル、テクスチャ、法線ベクトルなどの描画も省略する。複数回描画される影の立体に関しては、OpenGL の機能を用いて予めディスプレイ・リストを作成し使用する。地物や光源位置が不変の間は有効である。

影の立体は、原因となる物体の各面について作成している。このため、一つの面に関して、影の領域におけるステンシルバッファの差引値は１となるが、原因となる物体が閉多面体であるような場合には、通常、あるピクセルを影とする面は、原因物体の日向側と日陰側に二つ存在することとなり、物体全体では、影の領域におけるステンシルバッファの差引値は２となる。面の境界（稜）から伸びる表向きと裏向きの同一面は相殺可能である。

7-3-3. 影の表示に関する実際のプログラム

影の表示モードは、メイン画面のメニューで設定する。選択されたメニュー項目のコールバックの中で、g3SetShadowMode(G3_SHADOW_XXX);　関数が呼び出される。引数として選択される影の表示モードは、g3drl.h に定義されている。

リスト７－１７：影の表示モード

```
#define G3_SHADOW_ALL                    1   /* All Objects *//* すべての物体 */
#define G3_SHADOW_OTHER_GROUND           2   /* Other Ground Objects *//* 地面以外の物体 */
#define G3_SHADOW_ONE                    3   /* One Object *//* 1 つの物体 */
#define G3_SHADOW_OTHERLIGHT_OFF        10   /* Other LIHGT0 Lights OFF *//* LIGHT0 以外の光源を OFF */
#define G3_SHADOW_OTHERLIGHT_ON         11   /* Other LIHGT0 Lights ON *//* LIGHT0 以外の光源を ON */
```

影の表示を行わない場合には、引数は０である。

影の表示を行うモードが設定されている場合には、g3drl.c における、disp3dObjects() 関数から起動される、drawAllGroupEtc()関数の中で、通常の drawAllGroup の代わりに、

```
g3MakeShadowVolume();
drawAllGroupAndShadow();
```

の処理が行われる。

g3MakeShadowVolume()関数の中では、選択された影の原因物体（選択された物体、全ての物体）に関して、影の立体の側面に関するディスプレイリストを作成する。

次に、drawAllGroupAndShadow() 関数の中で、影の付いた表示を行う。

リスト７－１８：影の表示処理

```
/* シャドウとグループの描画
** drawAllGroupAndShadow( void )
*/
static void drawAllGroupAndShadow( void )
```

```
{
   int res,bits;//080716 DR.H.K.
/*1: 地物のデプスバッファ値を求める*/
   glClear( GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT | GL_STENCIL_BUFFER_BIT );
   res = GetLastError();
   glColorMask( GL_FALSE, GL_FALSE, GL_FALSE, GL_FALSE );//表示を止める
   drawAllGroup(); /* render all group in depthbuffer
      影の領域を求めるために下地の地物のデプスを出しておく*/
   /*080715 DR.H.K. drawAllGroup が 3 カ所ある。
      影の形状は、ディスプレイリストに保存されている。*/
/*2: 影の立体の側面の見栄をステンシルバッファに出す*/
   glEnable( GL_STENCIL_TEST );
   glDepthMask( GL_FALSE );//デプスバッファへの追記は止める


   glGetIntegerv( GL_STENCIL_BITS, &res);
   if( res < 1 ){
            MessageBox(NULL, "Stencil Buffer is not available. Please check pixel.c", NULL, MB_OK);
   }
   /* bits = (2 << res -1 ) - 1;  例 : res が 8 なら bits は 255 */
   bits = (1 << res) - 1;
   glClearStencil( 1 ); /*STENCIL BUFFER クリア値を 1 とする*/
   glClear( GL_STENCIL_BUFFER_BIT );  /*それを使ってクリア*/
   /*ステンシルの初期値をゼロとし、1 以上を影とすることで単純化は可能
      ここでは、ステンシルの用例（特に不等号判定）の意味を例示する*/
   glStencilFunc( GL_ALWAYS, 0, bits );

   glEnable( GL_CULL_FACE ); /////
/*2-1: 影の立体の側面の表向きの面をカウントアップする*/
   glStencilOp( GL_KEEP, GL_KEEP, GL_INCR );
   glCullFace( GL_BACK ); /* 裏面を表示しない */
   glCallList( G3_DLIST_SHADOWVOL );//DL 化し準備してある影を描画
/*2-2: 影の立体の側面の裏向きの面をカウントダウンする*/
   glStencilOp( GL_KEEP, GL_KEEP, GL_DECR );
   glCullFace( GL_FRONT ); /* 表面を表示しない */
   glCallList( G3_DLIST_SHADOWVOL );
/*3: 地物の表面から影の部分だけを切り出して描画*/
   glDepthMask( GL_TRUE );/*デプスバッファ書込を復活*/
   glClear( GL_DEPTH_BUFFER_BIT );/*デプスバッファをクリア*/
   glColorMask( GL_TRUE, GL_TRUE, GL_TRUE, GL_TRUE );

   glCullFace( GL_FRONT );//表面を表示しない
   glDepthFunc( GL_LEQUAL );
   glStencilOp( GL_KEEP, GL_KEEP, GL_KEEP );/*ステンシル値を固定*/

   glStencilFunc( GL_LEQUAL, 2, bits ); /* 2 がステンシル値以下なら表示する */
   /*即ち、ステンシル値が 2 以上の部分だけ、影と判定して表示する*/
   glEnable( GL_STENCIL_TEST );
   glDisable( GL_LIGHT0 );      //主照明を消灯
   glDisable( GL_CULL_FACE ); //これを実行しないと、裏面だけが表示される
   drawAllGroup();//ステンシルバッファの影の領域を、立体表示
/*4: 影以外の部分を描画する*/
   glStencilFunc( GL_GREATER, 2, bits ); /* 2 がステンシル値を超えるなら表示する */
   /*即ち、ステンシル値が 2 未満の部分だけ、影以外と判定して表示する*/
//参考（同一結果） glStencilFunc( GL_EQUAL, 1, bits );
                                 /* 1 がステンシル値に等しいなら表示する */
   glEnable( GL_LIGHT0 );       //主照明を点灯
   drawAllGroup();//影以外の地物を再描画する

   glDepthFunc( GL_LESS );
   glClearStencil( 0 );//STENCIL BUFFER クリア値を 0 に戻す
   glClear( GL_STENCIL_BITS );  //それを使ってクリア
   glDisable( GL_STENCIL_TEST );
   glDeleteLists( G3_DLIST_SHADOWVOL, 1 );
}
```

影の原因物体を構成する個々のポリゴンから光源とは反対の向きに、十分な長さ（ユーザーにより設定可能、初期値 1,000m）だけ伸びる影の立体に内包される、影の対象物体の表面上の領域が、影が落ちる領域となる。この部分のピクセルに関しては、地物全体のデプスバッファ値に対して、影の原因物体を構成するポリゴンの内、このピクセルに影を落とす立体の各面が生成する影の立体の前後関係を、影の対象物体を含む全地物のデプスバッファ値を用いて判定（但し、デプスバッファ値は比較結果をもって変更しない）した時に、影の立体の表面の数が裏面の数よりも１多くなる。但し、別の物体により隠され、見えなくなる影の対象物体は、表面も隠されるため、この限りではない。

影の原因物体が閉多面体である場合、影の対象物体の同じ領域に関して、原因物体の日向側の面と日陰側の二つの面から影が落ちる。日陰側の面から生成する影の立体が、外側を裏側とする面で構成されると、これらが表面の数と裏面の数を相殺し、差がゼロとなってしまい判定できない。そこで、影の立体の側面を作成する再に、原因物体を構成する各面に関して、光源に対して日向側と日陰側かを判定し、いずれの場合もポリゴン毎に作成する影の立体を構成する側面の外側が表側になるように、影の立体の面を作成しておく。

なお、表示に際して差引値の判定に用いる glStencilFunc(func, ref, mask); 関数において、第１引数で指定する判定条件により、第 2 引数と、ステンシルバッファ値の比較を指定しているが、例えば GL_GREATER という比較条件においては、「ref > ステンシル値」（逆ではない）という条件設定となる点には注意を要する。そこで、誤解を避ける目的で、表面と裏面の集計判定のために、ステンシルバッファに初期値として１をセットしておき、まず表側を描画し、デプス・バッファテストに合格した点（見える影の立体の表面）について、バッファ値をインクリメントしておく。次に裏側を描画し、デプス・バッファテストに合格した点（見える影の立体の裏面）について、バッファ値をデクリメントする。結果が１よりも大きい場合に、そのピクセルを影が表示される領域と判定している。

次に、このステンシルの判定条件を用いて、まず影の領域に関して主照明または全照明を消して描画を行い、最後に影以外の領域に関して、全照明を当てて描画を行っている。

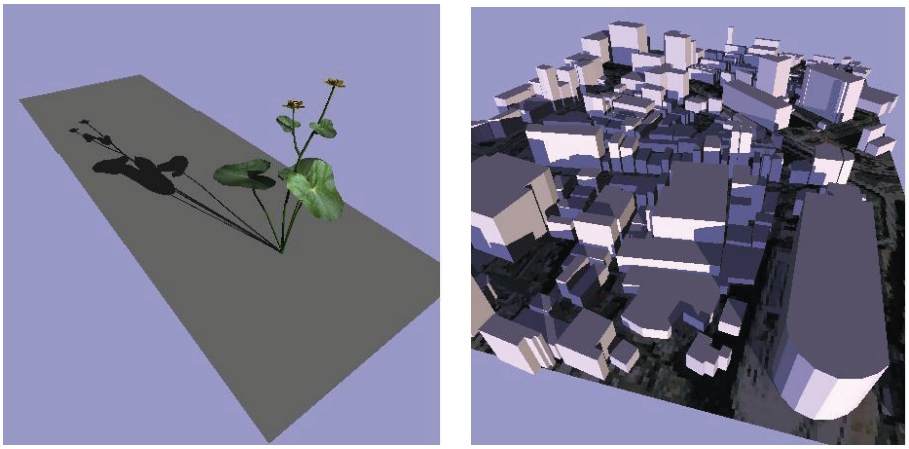


図７－２：影の表示例

７－６．高速表示処理

高速表示処理を行うために、以下の三つの処理機能を追加した。

(1) ディスプレイリストの使用

(2) 連続する三角形、四角形に関する、glBegin～glEnd の省略

(3) マウス・ドラッグによる視点移動

この内、(1)及び(2)は、メインメニューの[表示][高速表示]で開く高速表示設定ダイアログで設定する（4-4(62)）。

また、(3) マウス・ドラッグによる視点移動は、操作ボタンだけではなく、メイン画面上でのマウス右クリックのドラッグ操作によっても指定できるインターフェースを追加した。

7-6-1. ディスプレイリストの使用

ディスプレイリストを用いることにより、データを OpenGL に送出するための準備段階の処理が省略されるため、視点移動を高速化する。特に、CPU の処理速度が遅いマシンに、高速のグラフィックス・ボードが装着されているような場合に改善効果が顕著である。

オブジェクトの編集が行われた場合には、ディスプレイリストを再構築しなければならないため、再描画に時間がかかるが、視点移動のみの場合には、高速で表示される。

同じ物体が多数配置されているような地物構成の場合には、ディスプレイリストに展開した場合には、表示に必要な全ての面に展開されるため、メモリを大量に消費する。

drawAllGroup()の中で、

```
glNewList(dispList, CL_COMPILE_AND_EXECUTE);
    ・・・・・・
glEndList();
```

により、処理している。二回目以降の標示においては、ここで作成したディスプレイリストを用いた表示のみを行う。

glCallList

7-6-2. glBegin と glEnd の省略

三角形、四角形等が連続する場合に、面と面の間にある glEnd, glBegin を省略する。

drawMuka()の中で処理を行う。

数値地図や、ステレオ空中写真・衛星画像から作成した地形データのように、三角形が連続するようなデータでは、この処理による改善効果が高く、ディスプレイリストと併用すると、10 倍以上の速度に改善される場合がある。

WindowsVISTA においては、面に対して設定されるカラー、テクスチャおよびこれらを含んだマテリアルが変わる場合には、一度 glEnd→変化した表面光学特性の設定→glBegin という処理を行わないと正常に表示されない（設定内容が、次の頂点にしか反映されない）。

7-6-3. マウス・ドラッグによる視点移動

CDrawFrm::OnMouseMove() 関数により実行する。この関数は、マウスが移動された場合に起動する。マウスの右ボタンの状態、シフトキー押し下げの状態、及び Ctrl キーの押し下げの状態は、それぞれのイベントに応じてフラグに設定しておく。表示がパースで、右ボタンが押し下げられている場合以外は何もしない。次にマウスカーソルの位置を記憶しているスタティック変数と現在のマウスカーソルの位置を比較し、差分に応じて視点を移動する。左右の移動は、注視点を中心とする視点の左右回転に、マウスの上下の移動は視点の上下の回転に反映させる。これは、メイン画面下の「回転」の操作ボタンに対応する。シフトキーが押されている場合には、視点と注視点を平行移動する。これは、「シフト」の操作ボタンに対応する。コントロールキーが押されている場合には、マウスカーソルの上方移動に応じて接近、下方移動に応じて後退する。これは「接近」「後退」の操作ボタンに対応した動きである。（リスト７－１９）。

視点座標ダイアログ(CEditShit)が開いている場合には、G3DRL ライブラリの関数を用いて再描画に先立って、これらの視点移動に伴い変化する座標値等を表示部に設定する。これにより更新された視点座標・注視点座標が動的に数値として確認することができる。

リスト７－１９：マウス・ドラッグによる視点移動

```
static CPoint savePoint;//マウス移動時更新

void CDrawFrm::OnMouseMove(UINT nFlags, CPoint point) {
 if((nFlags & 2) == 0)//外でRIGHT_BUTTON=2 が外されたまま戻ってきた場合
    if(nSpinFlag && GetViewType() == G3_CAM_PERS) nSpinFlag = 0;
 if(nSpinFlag && GetViewType() == G3_CAM_PERS) {
    CMainFrame* pMainFrame = (CMainFrame*)AfxGetMainWnd();
    ASSERT(pMainFrame->IsKindOf(RUNTIME_CLASS(CMainFrame)));
    CPoint move;

    CClientDC dc(this);//080801
    *m_HDC = dc.m_hDC;//080801
    wg3AssignDrawarea( (void*)&m_HDC );

    move = savePoint - point;
    savePoint = point;
    if(nShiftKeyFlag) {
       g3SetCameraHorizontalShift(0.001f * (double)move.x);
       g3SetCameraVerticalShift(0.001f * (double)-move.y);
    }else if(nCtrlKeyFlag) {
       if(move.y > 0)
          g3SetCameraZoom(1.0 + (K_ZOOM_UP_RATE - 1.0) * 0.1);
       else if(move.y < 0)
          g3SetCameraZoom(1.0 / ( 1.0 + (1.0 / (K_ZOOM_DOWN_RATE) - 1.0 ) * 0.1 ));
    }else {
       g3SetCameraHorizontalRound(0.1f * (double)move.x);
       g3SetCameraVerticalRound(0.1f * (double)move.y);
    }
    pMainFrame->m_editShit->SetEyeWinParam();
    RedrawWindow();
  }
  CWnd::OnMouseMove(nFlags, point);
}

void CDrawFrm::OnKeyDown(UINT nChar, UINT nRepCnt, UINT nFlags) {
  if(nSpinFlag && GetViewType() == G3_CAM_PERS) {
    if(nChar == 16)
       nShiftKeyFlag = TRUE;
```

```
      else if(nChar == 17)
        nCtrlKeyFlag = TRUE;
      else {
        nShiftKeyFlag = FALSE;
        nCtrlKeyFlag = FALSE;
      }
    }
    CWnd::OnKeyDown(nChar, nRepCnt, nFlags);
}

void CDrawFrm::OnKeyUp(UINT nChar, UINT nRepCnt, UINT nFlags)
{
    nShiftKeyFlag = FALSE;
    nCtrlKeyFlag = FALSE;

    CWnd::OnKeyUp(nChar, nRepCnt, nFlags);
}
```

７－７．画像のファイル保存、動画保存、印刷

（１）表示画面のファイル保存

景観シミュレータ sim.exe のメニュー[ファイル]の下にある、[JPEG 形式で出力]メニューから起動する、OnMenuImageSave()関数において、画像の保存を行っている。ファイル選択ダイアログを開いて、ユーザーが保存ファイル名を指定し OK で終了した場合に、拡張子に従い、以下の処理を行っている。拡張子はデフォルトで「.jpg」であるが、ユーザーにより、保存ファイル名の拡張子に「.sgi」が指定された場合には SGI 形式での保存を行う。

①JPEG 形式の場合、画面から OpenGL 描画領域を取り出し、ピクセル情報を配列にコピーした上で、JPEG ライブラリ関数を用いて圧縮を行い、ユーザーが指定したファイルに保存する。

②SGI 形式の場合、画面から OpenGL 描画領域を配列に取り出し、アプリケーション・ライブラリ ISAVE.C の SaveImageData 関数を用いてファイルに保存している。

なお、この処理の中には、指定されたファイル名本体部の末尾が「_small」または「_s」が指定された場合に景観データベース見出し用の小さな画像を作成する処理を行っている。

（２）動画のファイル保存

景観シミュレータのメニュー[編集][視点設定][移動経路設定]により起動する、移動経路ダイアログ(CKeiroWnd)においては、ユーザーが設定した経路に従って、視点をユーザーが指定した刻みで移動するたびに、メイン画面の表示を更新することにより、メイン画面で動画として表示する機能を提供している。

この移動経路ダイアログで、設定した移動経路に従って移動（ウォークスルー）が実行できる状態がセットされている状態において、ダイアログ下部にあるスタートボタンの代わりに、メニューの[ファイル][動画保存]を選択して動画を保存する AVI ファイル名を指定すると、下記の処理が実行され、設定されている移動を実行すると同時に動画ファイルを作成する。

具体的には、以下の処理を行っている：

① CKeiroWnd::OnMyAviSave() において、まず保存するファイル名の入力をユーザーに求め、OK で終了すると、動画保存処理を開始する。

②この時、CMyEditAvi クラス(MyAvi.cpp)の変数 avi を作成し、視点移動の開始時点で、指定されたファイル名で avi.Create 関数を実行する。

③視点位置が変更されるステップ毎に、メイン画面の OpenGL 描画領域(CDrawFrm)の表示結果をビットマップとして取得し、これを avi.Append 関数を用いて動画ファイルに追加する。

④ 移動経路の終点に到達すると、avi.Close 関数を実行し終了する。

（３）画像のプリンタ出力

従来の方法(sim.exe Ver.2.07 まで)においては、画像保存は以下の方法に従っている。

① プリンタを、環境設定ファイルに定義された文字列から識別する。

② 表示画像を、スクリーンから取得し、プリンタに送出する。

これを、具体的には以下の処理プログラムで実現している。

メイン画面のメニュー構成において、カラー印刷は、ID_FILE_PRINT で、これが指定されると、以下の関数が起動される。

(ソース：Mainfrm.cpp)

void CMainFrame::mainColorPrintCB()

メイン画面の OpenGL 描画エリアを RECT として取得し、ColorPrintDIB 関数を起動する。

(ソース：B3bitmap.c)

int　ColorPrintDIB(RECT *rect)

引数として渡された画面の部分をコピーし、DIB 構造体を作成する。

これを引数として、PrintDIB 関数を起動する

成功なら TRUE、失敗なら FALSE を返す

(ソース：print.c)

WORD　PrintDIB(HDIB hDib, WORD fPrintOpt, WORD wXScale, WORD wYScale,
LPSTR szJobName)

汎用の開発用キットに含まれるソースコードに追記し、環境設定ファイルに指定されたプリンタ名称から印刷用のデバイス・コンテキストを作成し、ここに引数として渡された DIB 構造体を出力する。この関数は、

成功なら０、失敗ならエラーコードを返す

この処理プロセスは、メニューの[ファイル][画面印刷]の機能として Ver.2.09 に残してある。

一方、最近の各種アプリケーションでは、通常、メイン画面の[ファイル]メニューの下に、

プリンタの設定、印刷プレビュー、印刷の３のメニュー項目がある。これによりユーザーはアプリケーションの中からプリンタを選択・変更できる。これらの機能は、Document-View アーキテクチャの Cview の中に標準で含まれており、プログラマは、アプリケーションの中で定義する派生クラス（景観シミュレータ sim.exe の場合には、CSimView）の中で CView クラスにおいて予め用意されている OnPrint 関数をオーバーライドし、その中で、引数として渡された印刷用のデバイス・コンテキストを利用して、アプリケーション独自の印刷機能をプログラムする。

CSimView クラスは、Document-View アーキテクチャにおける CView の派生クラスとして定義されている。

CView クラスには、印刷に関して、以下の関数が提供されている。

リスト７－２０：印刷のための CView クラスのメンバ関数

OnPrint 印刷を実行する。 DoPrintPreview　印刷プレビューを実行する。 OnFilePrintPreview 印刷プレビューのコールバック OnBeginPrinting OnEndPrinting OnPreparePrinting

プリンタの選択は、CWinApp クラスで提供されているため、派生クラスである CSimAPp (sim.cpp)にオーバーライドされている。

上記の通り、「画面印刷」として残してある従来のカラー印刷機能で、環境設定ファイルにより指定されたプリンタが使用不能である場合には、CPrintDialog を直接開いて、利用可能な印刷機の選択をユーザーに要求するように修正した。

ユーザーが適切な印刷機を選択して OK で終了した場合には、これを環境変数(E3_COLOR_PRINTER)に設定した上で、再度印刷処理を行う（但し、ユーザーが環境編集ダイアログから保存操作を行わない限り、kdbms.set ファイルの修正は行わない)。

メニューで印刷機の選択が行われた場合には、CWinApp::OnFilePrintSetup()が標準で起動される(appprnt.cpp)。実際の印刷の実行は、CSimView::OnPrint 関数の中で実行している。従来のカラー印刷では、画面と同じ解像度の画像をプリンタに出力するが、メニューの印刷が選択された場合には、関数

void CSimView::OnPrint(CDC* pDC, CPrintInfo* pInfo)

により印刷機が提供する高い解像度での印刷を実行する。この関数には、引数としてプリンタのデバイス・コンテキストが渡される。

この関数の中では、印刷用のビットマップを作成して、この上に OpenGL による描画を行い、プリンタに出力する。この時、まず通常の方法で、プリンタの解像度のビットマップを作成する。

リスト７－２１：印刷用のビットマップの生成１

```
CBitmap mBM;
CDC mDC;
```

```
double BX = (double)Prect.right/(double)Srect.right;
double BY = (double)Prect.bottom/(double)Srect.bottom;
if(BX<BY){
        Prect.bottom = (LONG)((double)Srect.bottom * BX);
}else{
        Prect.right = (LONG)((double)Srect.right * BY);
}
hasil = mBM.CreateCompatibleBitmap(pDC,Prect.Width(),Prect.Height());
        if(hasil) mDC.SelectObject(mBM);
```

しかし、上記の mBM.CreateCompatibleBitmap 関数は、利用可能なメモリの限界よりも手前で、メモリ不足エラーを返す場合がある。そこで、エラーとなった場合には、以下のより原始的な方法でリトライしている。

リスト７－２２：印刷用のビットマップの生成２

```
BITMAPINFO bmpinfo;
HBITMAP hNewBmp;
HGDIOBJ hG;
ZeroMemory( &bmpinfo, sizeof(bmpinfo));
bmpinfo.bmiHeader.biSize = sizeof(BITMAPINFOHEADER);
bmpinfo.bmiHeader.biWidth = Prect.right;
bmpinfo.bmiHeader.biHeight = Prect.bottom;
bmpinfo.bmiHeader.biPlanes = 1;
bmpinfo.bmiHeader.biBitCount = 32;
bmpinfo.bmiHeader.biCompression = BI_RGB;
hNewBmp=CreateDIBSection(NULL,&bmpinfo,DIB_RGB_COLORS,NULL,NULL,0);
hG = SelectObject( mDC.m_hDC, hNewBmp);
hasil = (int) hG;
```

この方法により、メモリの限界まで解像度を上げて、印刷を行うことができる。

この二番目の方法でも、メモリ不足エラーを起こす場合には、画面の解像度で印刷を行う。

リスト７－２３：印刷用のビットマップの生成３

```
Prect = Srect;
hasil = mBM.CreateCompatibleBitmap(pDC,Prect.Width(),Prect.Height());
mDC.SelectObject(mBM);
```

なお、CSimView::OnPrint 関数は、印刷プレビューと、印刷の両方から共用されており、印刷プレビューの場合には、

CView::DoPrintPreview→・・・→CView::OnPaint →CView::OnDraw

→CSimView::Onprint

の経路で、また印刷の場合には、

CView::OnFilePrint→CSimView::OnPrint

の経路で呼び出される。

８．ファイル・コンバータ

８－１．概要

景観シミュレータの開発に着手した1993年当時に、外部ファイル形式として、モデルを記述するLSS-G、シーンを記述するLSS-S形式を、当時のSilicon Graphics社製のグラフィック・ワークステーション上で三次元CGを扱うOpenGLライブラリをターゲットとして定めた。その時点で、Open Inventorという三次元グラフィックスのプログラミングのためのツールキットが提供されており、この中のコマンドを参考とする形で開発が進められた。但し、各現場や設計担当者における導入に際して、ライセンス上の制約や余分なコスト負担が生じることを避けるために、当時の建築研究所・土木研究所における実際のシステム開発に際しては、リリースするソフトウェアには、ライセンス上の制約のあるライブラリ等を使用せず、全てゼロベースからの構築を行い、外部ファイル形式も独自のものとした。

当時、CADデータの交換のために、DXF形式が用いられていたが、テクスチャ・マッピングや、面から構成される完結した立体を記述するためには十分ではなく、また単純な形状を記述するために厖大なステップ数を必要とする形式であった。しかし、CAD入力成果を利用することは重要なニーズであったため、線分の集合体として表現されたDXFファイルから、線分の接続を解析し、面、さらには立体を復元するような処理を組み込んだコンバータを開発した。その後DXF形式は次第にバージョン・アップされ、面やソリッド、パラメトリックな図形や、自由曲面まで表現できる形式に拡張されて現在に至っている。

インターネットを介して三次元データを配信するためにVRML形式が標準化された。これもOpen Inventorをベースとしたものであり、景観シミュレータのためのデータを記録するための形式との間でコンバータを作成するには親和性の良い形式であった。但し、景観シミュレータにおいては、地物固有の形状や表面仕上げを記述するLSS－Gファイルと、表示のための環境条件である光源や時間や視点・注視点などのカメラ情報を記述するLSS-Sファイルを分離し、後者が前者を参照する方法を採用したのに対し、VRMLでは、一つのファイルに一体化されている。

地図・GIS分野では、メッシュに切ったエリアの標高を高さ値で表現するDEM（Digital Elevation Model）の使用が開始された。これを景観シミュレーションに利用するためには、統一されない様々な形式で納品される測量成果のDEMデータを読み込み、これを三角形の集合として表現する地表面に変換するコンバータを用意する必要があった。幸い、1992～93年度に、国土地理院により、建設技術評価制度を適用した、ステレオ空中写真の解析技術の評価が行われ、この時に審査するためのデータ形式が指定された。認定を受けようとする測量各社は、この形式による測量成果の提出を求められたため、各システムの出力コンバータを整備することとなった。このため、景観シミュレーションのためには、この評価用のファイル形式を利用するコンバータさえ用意しておけば、各社各システムの測量結果を利用することができた。その後、国土地理院から数値地図としてDEMの提供が開始さ

れた(1:25,000-1:50,000 に相当する 50m メッシュ)。これにはデータ形式の定義が解説されている。基本的には DEM の一類型として選択することとなった。2004 年には、レーザースキャナを用いた高精度の数値地図が、重要な都市部に関して作成・配布されるようになった（5m メッシュ)。これは従来の数値地図とはかなり異なる形式であったが、やはり DEM の一類型であり、コンバータを拡充することにより対応した。

インフラや自然条件の記述のために、GIS データが建設省－国土交通省の各現場に導入されるようになった。GIS データを交換するための代表的なデータ形式は SHP 形式である。SHP ファイルは、様々の図形要素を記述する、かなり内部形式の異なるサブセットの集合体である。この内、地形を表現する SHP ファイルは、DEM の一種として処理可能な形式であったため、やはり DEM の一類型としてコンバータで対応することとした。併せて、平面図形である等高線で地形を表現している SHP ファイルについても、DEM 型の地形データに変換した上で、三角形の面の集合体としての地形に変換する機能を作成した。

建築確認申請に三次元データが提出されれば、斜線制限などのチェックを自動化することも可能となる。このため、1992 年頃から電子申請に三次元データを添付する方法が導入され、データ形式（.330）が公開された（施行規則第十一条の三、第十四号様式)。残念ながら、この方法は普及していないが、2002 年頃に、試行的に景観検討にも直ちに利用できるようにコンバータを開発した。このデータ形式は、比較的単純なテキスト・ファイルで三次元座標を記述する方法を採っているため、シンプルな変換で足りる(5-1(2)②)。

建築研究所では長年、市街地延焼シミュレーションの開発に取り組んでおり、市街地の建物を立体的に記述するためのデータ形式を定めている。窓などの開口部が延焼に関係することから、建物の三次元形状のみならず、外壁の材料や開口部を記述できる形式となっている。データ形式は比較的単純なテキスト・ファイルで三次元座標を記述しているため、シンプルな変換で足りるが、窓が無い建物は殆ど延焼しないため、窓に関する情報が無い市街地データ（ステレオ空中写真解析結果など）を延焼シミュレータに出力する場合には、標準的な窓を自動的に追加するような機能を付加してある(5-1(2)③)。

現在では、三次元形状を記述するために、様々なデータ形式が用いられている。三次元形状に加えて、産業分野における CAD-CAM 連携など、使用目的に応じて、様々な属性情報（例えばメカニズムの記述）が付加されている。また、レーザー・スキャナなどを用いて人体や地物などの自由形状を計測する技術が発達し、自由曲面などの複雑な形状のコンパクトな記録や、WEB 上での配信のための工夫も行われている。

Ver.2.09 においては、外部関数に加えて、プラグイン DLL の仕組みにより、今後も発展が予想される様々のデータ形式に、基幹部分のソースコードを変更することなく対応することができる。

リスト８－１：現在使用されている三次元データ形式

IGES (.IGS .IGES)
Unigraphics (.PRT)

MasterCAM (.MC8 etc)
STL (.STL)
VDA-FS (.VDA)
CADKEY (.CDL)
NFL (.NFL)
SolidWorks (.SLDPRT)
SolidEdge (.PAR)
Parasolid (.X_T etc.)
STEP (.STP)
AutoCAD (.DXF .DWG)
3DS (.3ds)
3dsMAX (.MAX)
ACIS (.SAT)
CG メタファイル(.CGM)
VRML (.WRL)
VectorWorks (.mcd)
マイホームデザイナー (.m3d)
MAPCUBE (.obj)
FLT
ZMD
OpenFlightFormat
XVL
LandXML
PLM
PRC
PDF
COLLADA

８－２．外部関数による変換

形状を記述する比較的小さなファイルを部品として取り込み、地物に追加するためには、外部関数により記述されたコンバータを使用するのが便利である。変換結果は、一時的なLSS-G ファイルとして生成されるが、変換結果を含む地物をファイル保存する際には、関数名と引数である変換元のファイル名（例えば、XXX.wrl）だけが保存され、変換は読み込む度に実行される。

もし外部関数を用いて LSS-G 形式に展開したファイルが必要であれば、貿易コンバータ（８－３参照）のメニューから上記と同じ外部関数を選択し、これにより変換を行う。

①VRML 形式

外部関数 VRML2LSS.exe により実行する。

リスト８－２：VRML 形式の例

```
#VRML V2.0 utf8

DEF Plane01 Transform {
children [
DEF Plane01-FACES Transform {
children [
Shape {
appearance Appearance {
material Material {
diffuseColor 1.000000 1.000000 1.000000
emissiveColor 1.000000 1.000000 1.000000
specularColor 0.000000 0.000000 0.000000
ambientIntensity 0.000000
shininess 0.145000
}
texture ImageTexture {
url "m_eg_sm1.rgb"
}
textureTransform TextureTransform{
translation 0 0
scale 1 -1
}
}
geometry IndexedFaceSet {
coord Coordinate { point [
-4 8 0,-4 0 0,4 8 0,4 0 0]
}
creaseAngle 0.523599
normal Normal { vector [
0 0 1]
}
texCoord TextureCoordinate { point [
0 -1,0 0,1 -1,1 0]
}
coordIndex [
0, 1, 2, -1,
3, 2, 1, -1
]
normalPerVertex TRUE
normalIndex [
0, 0, 0, -1,
0, 0, 0, -1
]
texCoordIndex [
0, 1, 2, -1,
3, 2, 1, -1
]
}
}
]
}
]
}
```

②建築確認申請形式

外部関数 BS2LSS.exe により実行する。

リスト８－３：建築確認申請形式(.330)の例

```
/* 建物等の配置 */
```

```
#33000
/* 1.敷地境界線、敷地の接する道路の位置等 */
#33010,"-107000","-75500"," 5","-121459","-77528"," 0","-141459"," 64972"," 0",;
"-149187","-81416","-164731"," 61709"," 0"," 0","0","0","0"," 12600"
#33010,"-127000"," 67000"," 2","-129171"," 126961"," 106329"," 135461"," 0"," 2000",;
" 2000"," 60000"
#33010," 108500"," 75500"," 4"," 124109"," 79014"," 0"," 144609","-11986"," 0"," 0",;
" 0","0","0","0"," 15000"
#33010," 129000","-15500"," 4"," 144564","-19211"," 0"," 131564","-73711"," 0"," 0",;
" 0","0","0","0"," 15000"
#33010," 116000","-70000"," 4"," 116632","-95492"," 0","-106368","-100992"," 0",;
" 0"," 0","0","0","0"," 22700"
/* 3.方位 */
#33030," 33600"
/* 4.用途地域 */
#33040,"00932"
/* 5.防火地域 */
#33050,"00601"
/* 6.敷地内における建築物の位置及び形状 */
#33060," 1"," 0"," 0"," 000"," 0"
#33065,"0","1"," 0"," 0"," 500"," 0",;
"-5229"," 59772"," 0"," 5229"," 59772"," 0"," 15529"," 57956"," 0"," 25357",;
" 54378"," 0"," 34415"," 49149"," 0"," 42426"," 42426"," 0"," 49149"," 34415",;
" 0"," 54378"," 25357"," 0"," 57956"," 15529"," 0"," 59772"," 5229"," 0",;
" 56846"," 4973"," 18541"," 55119"," 14769"," 18541"," 51717"," 24116",;
" 18541"," 46744"," 32730"," 18541"," 40350"," 40350"," 18541"," 32730",;
" 46744"," 18541"," 24116"," 51717"," 18541"," 14769"," 55119"," 18541",;
" 4973"," 56846"," 18541","-4973"," 56846"," 18541"
#33065,"0","1"," 0"," 0"," 9500"," 0",;
"-5229"," 59772"," 0"," 5229"," 59772"," 0"," 15529"," 57956"," 0"," 25357",;
" 54378"," 0"," 34415"," 49149"," 0"," 42426"," 42426"," 0"," 49149"," 34415",;
" 0"," 54378"," 25357"," 0"," 57956"," 15529"," 0"," 59772"," 5229"," 0",;
" 56846"," 4973"," 18541"," 55119"," 14769"," 18541"," 51717"," 24116",;
" 18541"," 46744"," 32730"," 18541"," 40350"," 40350"," 18541"," 32730",;
" 46744"," 18541"," 24116"," 51717"," 18541"," 14769"," 55119"," 18541",;
" 4973"," 56846"," 18541","-4973"," 56846"," 18541"
#33065,"0","1"," 0"," 0"," 18500"," 0",;
"-5229"," 59772"," 0"," 5229"," 59772"," 0"," 15529"," 57956"," 0"," 25357",;
" 54378"," 0"," 34415"," 49149"," 0"," 42426"," 42426"," 0"," 49149"," 34415",;
" 0"," 54378"," 25357"," 0"," 57956"," 15529"," 0"," 59772"," 5229"," 0",;
" 56846"," 4973"," 18541"," 55119"," 14769"," 18541"," 51717"," 24116",;
" 18541"," 46744"," 32730"," 18541"," 40350"," 40350"," 18541"," 32730",;
" 46744"," 18541"," 24116"," 51717"," 18541"," 14769"," 55119"," 18541",;
" 4973"," 56846"," 18541","-4973"," 56846"," 18541"
#33065,"0","1"," 0"," 0"," 27500"," 0",;
"-5229"," 59772"," 0"," 5229"," 59772"," 0"," 15529"," 57956"," 0"," 25357",;
" 54378"," 0"," 34415"," 49149"," 0"," 42426"," 42426"," 0"," 49149"," 34415",;
" 0"," 54378"," 25357"," 0"," 57956"," 15529"," 0"," 59772"," 5229"," 0",;
" 56846"," 4973"," 18541"," 55119"," 14769"," 18541"," 51717"," 24116",;
" 18541"," 46744"," 32730"," 18541"," 40350"," 40350"," 18541"," 32730",;
" 46744"," 18541"," 24116"," 51717"," 18541"," 14769"," 55119"," 18541",;
" 4973"," 56846"," 18541","-4973"," 56846"," 18541"
#33065,"0","1"," 0"," 0"," 500"," 18541",;
"-4973"," 56846"," 0"," 4973"," 56846"," 0"," 14769"," 55119"," 0"," 24116",;
" 51717"," 0"," 32730"," 46744"," 0"," 40350"," 40350"," 0"," 46744"," 32730",;
" 0"," 51717"," 24116"," 0"," 55119"," 14769"," 0"," 56846"," 4973"," 0",;
" 48356"," 4231"," 16726"," 46887"," 12563"," 16726"," 43993"," 20514",;
" 16726"," 39762"," 27842"," 16726"," 34324"," 34324"," 16726"," 27842",;
" 39762"," 16726"," 20514"," 43993"," 16726"," 12563"," 46887"," 16726",;
" 4231"," 48356"," 16726","-4231"," 48356"," 16726"
#33065,"0","1"," 0"," 0"," 9500"," 18541",;
"-4973"," 56846"," 0"," 4973"," 56846"," 0"," 14769"," 55119"," 0"," 24116",;
" 51717"," 0"," 32730"," 46744"," 0"," 40350"," 40350"," 0"," 46744"," 32730",;
" 0"," 51717"," 24116"," 0"," 55119"," 14769"," 0"," 56846"," 4973"," 0",;
" 48356"," 4231"," 16726"," 46887"," 12563"," 16726"," 43993"," 20514",;
" 16726"," 39762"," 27842"," 16726"," 34324"," 34324"," 16726"," 27842",;
" 39762"," 16726"," 20514"," 43993"," 16726"," 12563"," 46887"," 16726",;
```

```
" 4231"," 48356"," 16726","-4231"," 48356"," 16726"
#33065,"0","1"," 0"," 0"," 18500"," 18541",;
"-4973"," 56846"," 0"," 4973"," 56846"," 0"," 14769"," 55119"," 0"," 24116",;
" 51717"," 0"," 32730"," 46744"," 0"," 40350"," 40350"," 0"," 46744"," 32730",;
" 0"," 51717"," 24116"," 0"," 55119"," 14769"," 0"," 56846"," 4973"," 0",;
" 48356"," 4231"," 16726"," 46887"," 12563"," 16726"," 43993"," 20514",;
" 16726"," 39762"," 27842"," 16726"," 34324"," 34324"," 16726"," 27842",;
" 39762"," 16726"," 20514"," 43993"," 16726"," 12563"," 46887"," 16726",;
" 4231"," 48356"," 16726","-4231"," 48356"," 16726"
#33065,"0","1"," 0"," 0"," 27500"," 18541",;
"-4973"," 56846"," 0"," 4973"," 56846"," 0"," 14769"," 55119"," 0"," 24116",;
" 51717"," 0"," 32730"," 46744"," 0"," 40350"," 40350"," 0"," 46744"," 32730",;
" 0"," 51717"," 24116"," 0"," 55119"," 14769"," 0"," 56846"," 4973"," 0",;
" 48356"," 4231"," 16726"," 46887"," 12563"," 16726"," 43993"," 20514",;
" 16726"," 39762"," 27842"," 16726"," 34324"," 34324"," 16726"," 27842",;
" 39762"," 16726"," 20514"," 43993"," 16726"," 12563"," 46887"," 16726",;
" 4231"," 48356"," 16726","-4231"," 48356"," 16726"
#33065,"0","1"," 0"," 0"," 500"," 35267",;
"-4231"," 48356"," 0"," 4231"," 48356"," 0"," 12563"," 46887"," 0"," 20514",;
" 43993"," 0"," 27842"," 39762"," 0"," 34324"," 34324"," 0"," 39762"," 27842",;
" 0"," 43993"," 20514"," 0"," 46887"," 12563"," 0"," 48356"," 4231"," 0",;
" 35133"," 3074"," 13274"," 34065"," 9128"," 13274"," 31963"," 14905",;
" 13274"," 28889"," 20228"," 13274"," 24938"," 24938"," 13274"," 20228",;
" 28889"," 13274"," 14905"," 31963"," 13274"," 9128"," 34065"," 13274",;
" 3074"," 35133"," 13274","-3074"," 35133"," 13274"
#33065,"0","1"," 0"," 0"," 9500"," 35267",;
```

③延焼シミュレーション形式

外部関数 FIRE2LSS. exe により実行する。

リスト８－４：延焼シミュレーション形式の例（建築物の記述）

```
838,2
10000,2,0,3,3,1,0,2
0
4
-16087.0619233643,-31910.5970964464,0
-16083.7964626323,-31917.6681184415,0
-16077.0130559097,-31914.6349389426,0
-16079.8947789214,-31907.5167793032,0
4
0,4
1,-16086.4496494771,-31911.9229130705,.8,-16086.0414668855,-31912.8067908199,2
1,-16084.8169191111,-31915.458424068,.8,-16084.4087365195,-31916.3423018174,2
1,-16086.4496494771,-31911.9229130705,3.8,-16086.0414668855,-31912.8067908199,5
1,-16084.8169191111,-31915.458424068,3.8,-16084.4087365195,-31916.3423018174,5
1,4
1,-16082.5245738718,-31917.0993972855,.8,-16081.6766480314,-31916.7202498481,2
1,-16079.1328705105,-31915.582807536,.8,-16078.2849446701,-31915.2036600986,2
1,-16082.5245738718,-31917.0993972855,3.8,-16081.6766480314,-31916.7202498481,5
1,-16079.1328705105,-31915.582807536,3.8,-16078.2849446701,-31915.2036600986,5
2,4
1,-16077.5533789744,-31913.3002840102,.8,-16077.9135943508,-31912.4105140552,2
1,-16078.9942404803,-31909.7412041906,.8,-16079.3544558567,-31908.8514342356,2
1,-16077.5533789744,-31913.3002840102,3.8,-16077.9135943508,-31912.4105140552,5
1,-16078.9942404803,-31909.7412041906,3.8,-16079.3544558567,-31908.8514342356,5
3,4
1,-16081.2386185044,-31908.0943387676,.8,-16082.1345115598,-31908.4793784104,2
1,-16084.8221907259,-31909.6344973392,.8,-16085.7180837813,-31910.019536982,2
1,-16081.2386185044,-31908.0943387676,3.8,-16082.1345115598,-31908.4793784104,5
1,-16084.8221907259,-31909.6344973392,3.8,-16085.7180837813,-31910.019536982,5
10001,3,0,3.5,1,0,0,2
0
4
-16258.6554845246,-31873.4130781604,0
-16257.1961276149,-31876.5058805961,0
-16248.6758832601,-31872.493149151,0
-16250.1180985281,-31869.6439324294,0
```

```
4
0,3
1,-16258.1082256835,-31874.5728790738,.8,-16257.7433864561,-31875.3460796828,2
1,-16258.1082256835,-31874.5728790738,3.8,-16257.7433864561,-31875.3460796828,5
1,-16258.1082256835,-31874.5728790738,6.8,-16257.7433864561,-31875.3460796828,8
1,6
1,-16251.338459621,-31873.7471277276,6.8,-16250.2734290766,-31873.245536297,8
1,-16251.338459621,-31873.7471277276,3.8,-16250.2734290766,-31873.245536297,5
1,-16251.338459621,-31873.7471277276,.8,-16250.2734290766,-31873.245536297,2
1,-16255.5985817984,-31875.7534934501,.8,-16254.533551254,-31875.2519020195,2
1,-16255.5985817984,-31875.7534934501,3.8,-16254.533551254,-31875.2519020195,5
1,-16255.5985817984,-31875.7534934501,6.8,-16254.533551254,-31875.2519020195,8
2,3
1,-16249.2167139856,-31871.4246928804,6.8,-16249.5772678026,-31870.7123887,8
1,-16249.2167139856,-31871.4246928804,3.8,-16249.5772678026,-31870.7123887,5
1,-16249.2167139856,-31871.4246928804,.8,-16249.5772678026,-31870.7123887,2
3,6
```

④電子納品形式

外部関数 SCADEC2.exe により実行する。

リスト８－５：SXF 形式の例

```
ISO-10303-21;
HEADER;
FILE_DESCRIPTION(('SCADEC level2 AP202_mode'),
        '2;1');
FILE_NAME('026 ¥X2¥66AB5B9A6A2A65AD56F3FF08305D306E¥X0¥3¥X2¥FF09¥X0¥_model.p21',
        '2003-6-3T12:47:5',
        (''),
        (''),
        'SCADEC_API_Ver2.03',
        'Autodesk',
        '');
FILE_SCHEMA(('ASSOCIATIVE_DRAUGHTING'));
ENDSEC;
DATA;
#10=(
LENGTH_UNIT()
NAMED_UNIT(*)
SI_UNIT(.MILLI.,.METRE.)
);
#20=DRAUGHTING_PRE_DEFINED_CURVE_FONT('continuous');
#30=CURVE_STYLE_FONT_PATTERN(15.000000,0.750000);
#40=CURVE_STYLE_FONT_PATTERN(0.750000,0.750000);
#50=CURVE_STYLE_FONT('CENTERXA',(#30,#40));
#60=CURVE_STYLE_FONT_PATTERN(2.000000,1.000000);
#70=CURVE_STYLE_FONT('HIDDEN1',(#60));
#80=DRAUGHTING_PRE_DEFINED_CURVE_FONT('dashed spaced');
#90=CURVE_STYLE_FONT_PATTERN(5.000000,1.000000);
#100=CURVE_STYLE_FONT_PATTERN(1.000000,1.000000);
#110=CURVE_STYLE_FONT('CENTER1',(#90,#100));
#120=DRAUGHTING_PRE_DEFINED_CURVE_FONT('chain');
#130=CURVE_STYLE_FONT_PATTERN(3.200000,0.400000);
#140=CURVE_STYLE_FONT('2',(#130));
#150=EXTERNAL_SOURCE(IDENTIFIER('scadec'));
#160=EXTERNALLY_DEFINED_TEXT_FONT(IDENTIFIER('STANDARD'),#150);
#170=EXTERNAL_SOURCE(IDENTIFIER('scadec'));
#180=EXTERNALLY_DEFINED_TEXT_FONT(IDENTIFIER('YOKO'),#170);
#190=DRAUGHTING_PRE_DEFINED_COLOUR('white');
#200=PLANAR_EXTENT(' ',2000.000000,300.000000);
#210=DIRECTION(' ',(1.00000000000000,0.00000000000000));
#220=CARTESIAN_POINT(' ',(469.680086,380.898495));
#230=AXIS2_PLACEMENT_2D(' ',#220,#210);
#240=TEXT_LITERAL_WITH_EXTENT('$$SXF_baseline left','S=1:100',#230,'baseline left',.RIGHT.,#160,#200);
#250=TEXT_STYLE_FOR_DEFINED_FONT(#190);
```

```
#260=(
TEXT_STYLE(' ',#250)
TEXT_STYLE_WITH_BOX_CHARACTERISTICS((BOX_HEIGHT(300.000000),BOX_WIDTH(240.000000),BOX_S
LANT_ANGLE(0.00000000000000),BOX_ROTATE_ANGLE(0.00000000000000)))
TEXT_STYLE_WITH_SPACING(LENGTH_MEASURE(0.000000))
);
#270=PRESENTATION_STYLE_ASSIGNMENT((#260));
#280=(
ANNOTATION_OCCURRENCE()
ANNOTATION_TEXT_OCCURRENCE()
DRAUGHTING_ANNOTATION_OCCURRENCE()
GEOMETRIC_REPRESENTATION_ITEM()
REPRESENTATION_ITEM(' ')
STYLED_ITEM((#270),#240)
);
#290=(
NAMED_UNIT(*)
PLANE_ANGLE_UNIT()
SI_UNIT($,.RADIAN.)
);
#300=(
LENGTH_UNIT()
NAMED_UNIT(*)
SI_UNIT(.MILLI.,.METRE.)
);
#310=(
GEOMETRIC_REPRESENTATION_CONTEXT(2)
GLOBAL_UNIT_ASSIGNED_CONTEXT((#290,#300))
REPRESENTATION_CONTEXT('ID1','2D')
);
#320=CARTESIAN_POINT(' ',(0.000000,0.000000));
#330=AXIS2_PLACEMENT_2D(' ',#320,$);
#340=DRAUGHTING_SUBFIGURE_REPRESENTATION('$$SXF_P_W',(#280,#330),#310);
#350=SYMBOL_REPRESENTATION_MAP(#330,#340);
#360=DRAUGHTING_PRE_DEFINED_COLOUR('blue');
#370=PLANAR_EXTENT(' ',1850.000000,150.000000);
#380=DIRECTION(' ',(1.00000000000000,0.00000000000000));
#390=CARTESIAN_POINT(' ',(0.000000,-536.331200));
#400=AXIS2_PLACEMENT_2D(' ',#390,#380);
#410=TEXT_LITERAL_WITH_EXTENT('$$SXF_middleline centre', 'FEP30(1)50(1)', #400, 'baseline centre',
.RIGHT.,  #160,  #370);
#420=TEXT_STYLE_FOR_DEFINED_FONT(#360);
#430=(
TEXT_STYLE(' ',#420)
TEXT_STYLE_WITH_BOX_CHARACTERISTICS((BOX_HEIGHT(150.000000),BOX_WIDTH(120.000000),BOX_S
LANT_ANGLE(0.00000000000000),BOX_ROTATE_ANGLE(0.00000000000000)))
TEXT_STYLE_WITH_SPACING(LENGTH_MEASURE(0.000000))
);
#440=PRESENTATION_STYLE_ASSIGNMENT((#430));
#450=(
ANNOTATION_OCCURRENCE()
ANNOTATION_TEXT_OCCURRENCE()
DRAUGHTING_ANNOTATION_OCCURRENCE()
GEOMETRIC_REPRESENTATION_ITEM()
REPRESENTATION_ITEM(' ')
STYLED_ITEM((#440),#410)
);
----------------------------------------------------中略----------------------------------------
#2710=CARTESIAN_POINT(' ',(35075.646473,4482.606484));
#2720=CARTESIAN_POINT(' ',(28575.646473,4482.606484));
#2730=DIRECTION(' ',(6500.000000,0.000000));
#2740=VECTOR(' ',#2730,1.000000);
#2750=LINE(' ',#2720,#2740);
#2760=TRIMMED_CURVE(' ',#2750,(#2720),(#2710),.T.,.CARTESIAN.);
#2770=CURVE_STYLE(' ',#20,#550,#190);
#2780=PRESENTATION_STYLE_ASSIGNMENT((#2770));
#2790=(
ANNOTATION_CURVE_OCCURRENCE()
```

```
ANNOTATION_OCCURRENCE()
DRAUGHTING_ANNOTATION_OCCURRENCE()
GEOMETRIC_REPRESENTATION_ITEM()
REPRESENTATION_ITEM(' ')
STYLED_ITEM((#2780),#2760)
);
#2800=PLANAR_EXTENT(' ',1100.000000,300.000000);
#2810=DIRECTION(' ',(1.00000000000000,0.00000000000000));
#2820=CARTESIAN_POINT(' ',(32451.522200,3992.606484));
#2830=AXIS2_PLACEMENT_2D(' ',#2820,#2810);
#2840=TEXT_LITERAL_WITH_EXTENT('$$SXF_baseline left','3000',#2830,'baseline left',.RIGHT.,#160,#2800);
#2850=TEXT_STYLE_FOR_DEFINED_FONT(#190);
#2860=(
TEXT_STYLE(' ',#2850)
TEXT_STYLE_WITH_BOX_CHARACTERISTICS((BOX_HEIGHT(300.000000),BOX_WIDTH(240.000000),BOX_S
LANT_ANGLE(0.00000000000000),BOX_ROTATE_ANGLE(0.00000000000000)))
TEXT_STYLE_WITH_SPACING(LENGTH_MEASURE(0.000000))
);
#2870=PRESENTATION_STYLE_ASSIGNMENT((#2860));
#2880=(
ANNOTATION_OCCURRENCE()
ANNOTATION_TEXT_OCCURRENCE()
DRAUGHTING_ANNOTATION_OCCURRENCE()
GEOMETRIC_REPRESENTATION_ITEM()
REPRESENTATION_ITEM(' ')
STYLED_ITEM((#2870),#2840)
);
```

⑤LandXML

外部関数 LandXML. exe により実行する。

リスト８－６ ：LandXML 形式の例

```
<?xml version="1.0" encoding="utf-8"?>
<Skeleton xmlns="http://www.landxml.org/schema/LandXML-1.1"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.landxml.org/schema/LandXML-1.1 Skeleton-1.02.xsd" date="2006-04-10"
time="20:29:29" version="1.0">
<Units>
<Metric areaUnit="squareMeter" linearUnit="meter" volumeUnit="cubicMeter" temperatureUnit="celsius"
pressureUnit="HPA" angularUnit="decimal degrees" directionUnit="decimal degrees" />
</Units>
<CoordinateSystem name="DesignBasedOn" datum="JGD2000" verticalDatum="TP"
horizontalCoordinateSystemName="5(X,Y)" verticalCoordinateSystemName="H"
fileLocation="http://www.gsi.go.jp/GIS/"
/>
<Specifications>
<FormStandard name="TS 試行基準" appliWork="others" measMethod="measurement using TS"
standardType="position criteria">
<StandardValue objective="difference of CLOffset" lowerLimit="-0.5" lowerLimitation="limit" upperLimit="0.5"
upperLimitation="limit" optimalValue="0" unit="m" />
<StandardValue objective="difference of elevation" lowerLimit="-0.5" lowerLimitation="limit" upperLimit="0.5"
upperLimitation="limit" optimalValue="0" unit="m" />
</FormStandard>
<FormStandard name="TS 試行-土工切土" measMethod="measurement using TS" standardType="acceptance
criteria">
<ScopeGroup name="基準高・法長・幅" scopeType="CutFillComposition">
<StandardItem itemNameRef="基準高" unit="mm" standardValueName="±50">
<StandardTarget targetNameRef="道路中心" />
<StandardTarget targetNameRef="道路端" />
<Conditional operator="any">
<Judgment operator="gt or eq" target="difference" refValue="-50" valueType="numeric" />
<Judgment operator="lt or eq" target="difference" refValue="+50" valueType="numeric" />
</Conditional>
</StandardItem>
<StandardItem itemNameRef="法長" unit="mm" standardValueName="-200 or -4%">
```

<StandardTarget targetNameRef="切土法面" />
<Conditional operator="lt" target="designValue" refValue="5000" valueType="numeric">
<Judgment operator="gt or eq" target="difference" refValue="-200" valueType="numeric" />
</Conditional>
<Conditional operator="gt or eq" target="designValue" refValue="5000" valueType="numeric">
<Judgment operator="gt or eq" target="difference" refValue="-4" valueType="designValue%" />
</Conditional>
</StandardItem>
<StandardItem itemNameRef="幅" unit="mm" standardValueName="-100">
<StandardTarget targetNameRef="道路面" />
<StandardTarget targetNameRef="切土小段" />
<Conditional operator="any">
<Judgment operator="gt or eq" target="difference" refValue="-100" valueType="numeric" />
</Conditional>
</StandardItem>
</ScopeGroup>
</FormStandard>
<FormStandard name="TS 試行-土工盛土" measMethod="measurement using TS" standardType="acceptance criteria">
<ScopeGroup name="基準高・法長・幅" scopeType="CutFillComposition">
<StandardItem itemNameRef="基準高" unit="mm" standardValueName="±50">
<StandardTarget targetNameRef="道路中心" />
<StandardTarget targetNameRef="道路端" />
141
<Conditional operator="any">
<Judgment operator="gt or eq" target="difference" refValue="-50" valueType="numeric" />
<Judgment operator="lt or eq" target="difference" refValue="+50" valueType="numeric" />
</Conditional>
</StandardItem>
<StandardItem itemNameRef="法長" unit="mm" standardValueName="-200">
<StandardTarget targetNameRef="盛土法面" />
<Conditional operator="lt" target="designValue" refValue="5000" valueType="numeric">
<Judgment operator="gt or eq" target="difference" refValue="-100" valueType="numeric" />
</Conditional>
<Conditional operator="gt or eq" target="designValue" refValue="5000" valueType="numeric">
<Judgment operator="gt or eq" target="difference" refValue="-2" valueType="designValue%" />
</Conditional>
</StandardItem>
<StandardItem itemNameRef="幅" unit="mm" standardValueName="-100">
<StandardTarget targetNameRef="道路面" />
<StandardTarget targetNameRef="盛土小段" />
<Conditional operator="any">
<Judgment operator="gt or eq" target="difference" refValue="-100" valueType="numeric" />
</Conditional>
</StandardItem>
</ScopeGroup>
</FormStandard>
<FormStandard name="TS 試行-成績評定基準" appliWork="evaluation" measMethod="measurement using TS" standardType="evaluation criteria">
<ScopeGroup name="道路端部" scopeType="Alignment">
<StandardItem itemNameRef="道路端評価" unit="mm" viewType="出来形帳票図" viewDesc="道路設計幅員">
<StandardTarget targetNameRef="道路端部左側" viewSense="descending" viewMax="120" viewMin="-60" viewDesc="設計との差（ΔW(L)）" />
<StandardTarget targetNameRef="道路端部右側" viewSense="ascending" viewMax="120" viewMin="-60" viewDesc="設計との差（ΔW(R)）" />
<Conditional operator="any" evaluation="A 評価" evaluationPriority="1">
<Judgment operator="gt or eq" target="difference" refValue="-50" valueType="numeric" />
<Judgment operator="lt or eq" target="difference" refValue="50" valueType="numeric" />
</Conditional>
<Conditional operator="any" evaluation="B 評価" evaluationPriority="2">
<Judgment operator="gt or eq" target="difference" refValue="-80" valueType="numeric" />
<Judgment operator="lt or eq" target="difference" refValue="80" valueType="numeric" />
</Conditional>
<Conditional operator="any" evaluation="C 評価" evaluationPriority="3">
<Judgment operator="gt or eq" target="difference" refValue="-100" valueType="numeric" />
</Conditional>

```
</StandardItem>
</ScopeGroup>
</FormStandard>
</Specifications>
<Alignments>
<Alignment name="本線 CL" length="400" staStart="0">
<CoordGeom>
<Line length="400">
<Start>0 0</Start>
<End>0 400</End>
</Line>
</CoordGeom>
<Profile>
<ProfAlign name="本線 CL">
<PVI>0 23</PVI>
<PVI>400 31</PVI>
142
</ProfAlign>
</Profile>
</Alignment>
</Alignments>
<CrossSections>
<CrossSection alignmentRef="本線 CL">
<CrossSlope name="路面横断" slopeType="%" slope="-2" />
<Formation name="路面-01" staStart="100" staEnd="230">
<Side sideofRoadCenter="left">
<CrossSectionElement category="lane" priority="1" width="3.5" slopeRef="路面横断" />
<CrossSectionElement category="shoulder" priority="2" width="1.25" slopeRef="路面横断" />
</Side>
<Side sideofRoadCenter="right">
<CrossSectionElement category="lane" priority="1" width="3.5" slopeRef="路面横断" />
<CrossSectionElement category="shoulder" priority="2" width="1.25" slopeRef="路面横断" />
</Side>
</Formation>
<CutFillComposition name="左-盛土法面-01" side="left" staStart="110" staEnd="158">
<CrossSectionElement category="protection shoulder" priority="1" width="1.5" slopeType="%" slope="0">
<Mark name="道路端点" />
</CrossSectionElement>
<CrossSectionElement category="fill slope" priority="2" height="-5" slopeType="1:X" slope="-1.8" />
<CrossSectionElement category="fill berm" priority="3" width="1.5" height="-0.025" />
<CrossSectionElement category="fill slope" priority="4" height="-5" slopeType="1:X" slope="-1.8" />
</CutFillComposition>
<CutFillComposition name="左-切土法面-01" side="left" staStart="158" staEnd="230">
<CrossSectionElement category="protection shoulder" priority="1" width="1.5" slopeType="%" slope="0">
<Mark name="道路端点" />
</CrossSectionElement>
<CrossSectionElement category="cut slope" priority="2" height="5" slopeType="1:X" slope="1.2" />
<CrossSectionElement category="cut berm" priority="3" width="1.5" height="0.025" />
<CrossSectionElement category="cut slope" priority="4" height="5" slopeType="1:X" slope="1.2" />
<CrossSectionElement category="cut berm" priority="5" width="1.5" height="0.025" />
<CrossSectionElement category="cut slope" priority="6" height="5" slopeType="1:X" slope="1.2" />
</CutFillComposition>
<CutFillComposition name="右-盛土法面-02" side="right" staStart="110" staEnd="230">
<CrossSectionElement category="protection shoulder" priority="1" width="1.5" slopeType="%" slope="0">
<Mark name="道路端点" />
</CrossSectionElement>
<CrossSectionElement category="fill slope" priority="2" height="-5" slopeType="1:X" slope="-1.8" />
<CrossSectionElement category="fill berm" priority="3" width="1.5" height="-0.025" />
<CrossSectionElement category="fill slope" priority="4" height="-5" slopeType="1:X" slope="-1.8" />
</CutFillComposition>
<ControlStas NoName="No" NoInterval="20" startNo="0" startPlus="0">
<ControlSta sta="120" name="No6" formation="路面-01" LCutFill="左-盛土法面-01" RCutFill="右-盛土法面
-02">
<IG side="left" priority="2" width="1.818" height="-1.01" />
<IG side="right" priority="2" width="2.754" height="-1.53" />
</ControlSta>
```

```
<ControlSta sta="140" name="No7" formation="路面-01" LCutFill="左-盛土法面-01" RCutFill="右-盛土法面
-02">
<IG side="left" priority="4" width="4.014" height="-2.23" />
<IG side="right" priority="4" width="4.068" height="-2.26" />
</ControlSta>
----------------------------------------後略----------------------------------------
```

８－３．貿易コンバータによる変換

ファイルを入力し、ファイルを出力するような変換は、貿易コンバータを使用している。DEM、GIS、あるいは CAD 等の大きなデータに関しては、範囲の切り出しやレイヤーの選択などの複雑なパラメータ設定が必要となるため、外部関数の引数だけで変換条件を簡単に記述する方法には限界がある。貿易コンバータでは、複雑な変換条件の設定結果もファイルとして保存することができるため、変換条件を記録して再利用することができる。例えば、DEM データの変換に際しては、切り出し範囲を指定したり、間引き率を設定したり、同じエリアのカラー空中写真と位置合わせして、頂点にカラー情報を付けることができる。

貿易コンバータは、Windows NT 3.5 をターゲットとして C 言語で開発された初期のプログラム構成をそのまま保持している。メッセージマップや、MFC 等の便利な DLL は使用しておらず、Windows アプリケーションとしては非常に初期のプリミティブな構成で、現在では珍しくなり、むしろ歴史的な意味があるかも知れない。WinMain 関数でダイアログを構築したあと、コールバック関数の中で、メッセージとパラメータにより各処理に分岐している。

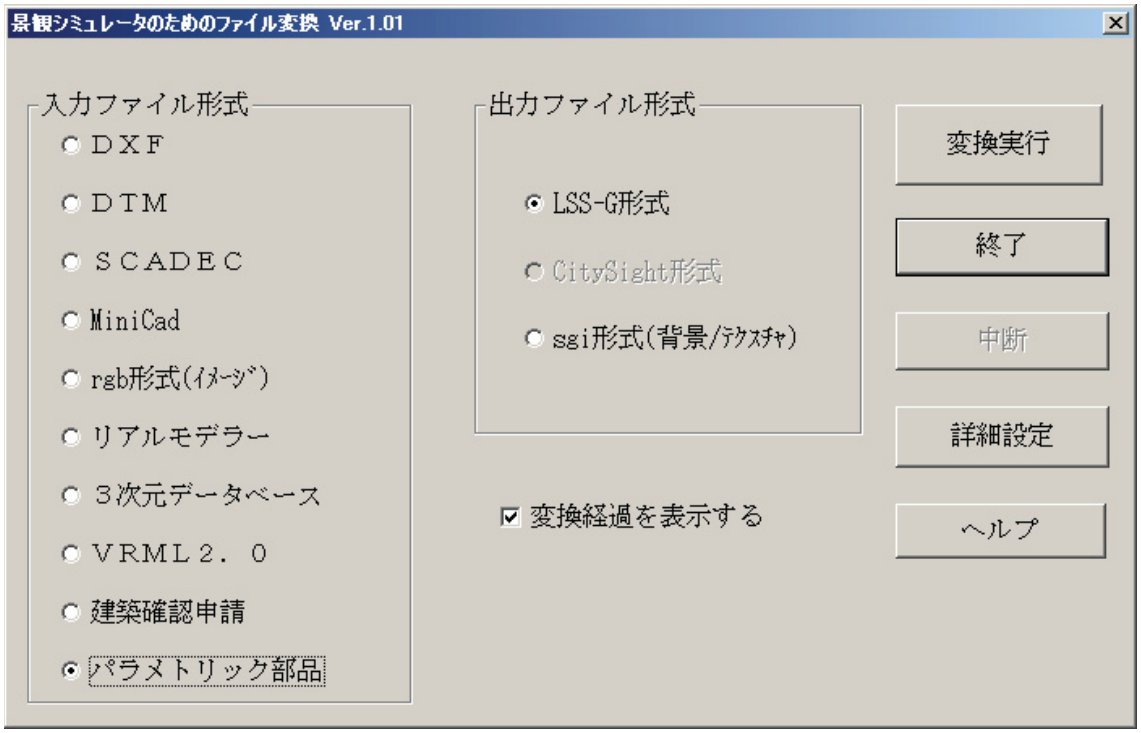


図８－１：貿易コンバータ（変換形式選択画面）

リソース：IDD_DIALOG1　ハンドラ：DIALOG100()（貿易.c）

ヘルプ：helpmain.txt

①DEM 形式

1993 年当時、既に地形を表現する DEM データは使用され始めていたが、その具体的な内容は、システムにより様々であり、データを入手してから、その形式に合わせてコンバータのプログラムを修正して読み込むような状況であった。

X,Y,Z の組を繰り返す形式の他に、長方形エリアのメッシュ点の標高データのみを連続的に出力する形式があり、縦横サイズ、原点位置（国家座標系などによる）、回転角などのデータを添付資料に基づいて補足入力しなければ最終的な座標が確定できないものが多かった。

その後、国土地理院による数値地図の配布開始と並行してメタデータが充実したため、これを併用して位置合わせなどが自動的に行えるようになった。

リスト８－７：数値地図（5m メッシュ標高）の例

```
     1  30  38  40  39  39  36  39  37  34  32 31-9999-9999-9999  17  32  32  33
33   33  33  31  36  34  32  31  32  33  31  23  22  23  22  23  23  22  22  23
33   35  34  33  28  33  33  31  29  33  37  36  37  37  38  36  36  35  37  37
37   38  38  39  37  33  33  33  33  33  31  30  28  30  31  31  32  31  30  29
31   22  22  27  26  28  27  26  27  28  30  30  31  30  29  28  31  33  35  36
35   33  22  21  23  25  26  25  22  25  28  32  31  30  29  32  30  27  26  26
25   26  27  26  24  24  26  25  24  24  24  25  26  25  23  24  33  34  33  34
32   30  28  28  29  30  32  31  27  26  27  25  26  24  24  23  23  23  24  24
24   23  24 24-9999-9999-9999-9999  22  21  23  24  25  25  27  26  27  26  28  26
26   28  27  25  25  26  26  26  27  26  27  28  27  27  29  28  29  27  27  28
26   26  26  21  21  22  24  24  24  24  24  24  22  23  23  22  24  24  24  24
24   23  24  25  26  24  20  19  20  22  22  22  24  24  24  25  23  21  20  21
20   18  20  20  20  20  20  20  21  21  21  21  20  18  17  21  21  22  23  19
19   20  18  18  20  19  19  19  19  19  19  19  19  18  18  23  23  24  25  26
20   20  24  22  23  22  23  24  25  24  23  24  25  26  26  25  25  26  22  24
27   27  26  26  27  26  23  23  23  24  28  31  29  29  29  26  25  25  29  29
27   25  23  26  26  29  28  25  24  22  21  21  24  26  25  23  23  22  21  24
21   21  22  21  21  20  21  21  21  21  22  23  23  22  21  21  21  22  23  23
23   24  25  23  26  27  24  24  21  21  22  22  24  23  24  24  24  23  24  24
23   24  23  22  22  22  24  25  22  23  25  25  23  25  25  17  21  22  23  22
23   26
     2  31  40  40  40  39  43  44  41  36  33 31-9999-9999-9999-9999  31  33  35
31   31  30  29  35  35  34  33  33  33  32  23  22  21  22  23  22  22  22  23
29   35  34  32  29  32  34  32  32  32  38  37  37  37  37  36  36  35  36  37
39   36  37  35  36  34  32  32  32  31  32  33  29  30  31  32  31  30  30  30
29   22  25  28  27  27  29  27  27  28  31  31  31  32  31  29  31  31  36  35
35   36  25  21  23  25  25  25  24  24  26  31  30  27  30  28  24  24  24  24
24   24  24  24  23  24  24  24  23  24  26  28  29  27  27  25  32  34  34  33
33   32  32  29  28  30  30  28  27  26  27  26  26  24  23  24  24  24  24  24
26   25  27 24-9999-9999-9999-9999  21  22  21  25  25  26  28  27  27  26  26  27
27   29  28  28  27  26  25  26  27  27  27  28  28  28  28  27  28  27  27  27
26   27  26  23  22  20  23  22  25  22  24  22  22  23  23  23  25  24  24  22
24   22  25  25  24  22  21  18  20  21  21  21  25  25  25  25  24  22  20  21
20   20  21  20  20  20  20  20  20  20  21  21  21  17  18  22  24  25  25  25
24   23  21  20  19  19  18  19  18  18  19  19  19  17  19  25  25  26  26  24
19   20  20  20  20  20  21  22  23  21  22  24  24  23  23  25  26  26  23  24
26   27  27  26  26  26  23  24  23  24  29  30  30  31  31  26  24  27  30  27
26   24  24  27  26  26  26  25  24  22  21  22  25  26  25  26  26  24  24  27
26   24  23  23  17  17  17  17  21  24  22  26  22  23  24  22  23  21  21  22
22   23  22  23  23  23  22  21  21  21  22  22  23  22  23  24  24  23  24  23
24   23  23  22  22  22  22  23  23  23  24  24  24  26  24  19  22  21  24  25
25   26
```

②SHP ファイル、DBF ファイル

SHP 形式、DBF ファイルは、ArcInfo 系の GIS で用いられているデータ形式である。ヘッ

ダーファイルの後に図形を表すデータの実体が続くが、様々のサブセットがあり、あらゆる形式に対応するコンバータを作成することは効率的ではない。現場のニーズにより、実際に処理する必要が生じた段階で次第に拡充してきたのが実態である。

リスト８－８：等高線を記述する DBF ファイル

（ヘッダー部分）

```
                FNODE_       N
                TNODE_       N
                LPOLY_       N
                RPOLY_       N
                LENGTH       F     ¶|                C311_        N
                C311_ID      N
                ZVALUE       F     ¶└                                        0              0              0
01.9971962853049E+002           1          16.7500000000000E+002          0              0              0
01.1482617999840E+002           2          26.6250000000000E+002          0              0              0
08.6704659759814E+002           3          36.6250000000000E+002          0              0              0
06.3971490054719E+002           4          46.6250000000000E+002          0              0              0
01.0642338773331E+003           5          57.3750000000000E+002          0              0              0
09.1052884729291E+001           6          66.7500000000000E+002          0              0              0
07.5995922494824E+002           7          77.6250000000000E+002          0              0              0
06.4193071235173E+002           8          87.7500000000000E+002          0              0              0
01.5740504400229E+002           9          96.7500000000000E+002          0              0              0
02.6219413461977E+002          10         107.1250000000000E+002          0              0              0
05.5574411531113E+002          11         118.2500000000000E+002          0              0              0
03.0159360370285E+003          12         138.8750000000000E+002          0              0              0
02.3835631279061E+003          13         148.8750000000000E+002          0              0              0
05.6786492920254E+001          14         158.6250000000000E+002          0              0              0
01.5143860910551E+003          15         166.7500000000000E+002          0              0              0
03.3841316777564E+002          16         177.8750000000000E+002          0              0              0
05.3625742128348E+002          17         187.7500000000000E+002          0              0              0
09.3502899539869E+002          18         197.6250000000000E+002          0              0              0
01.2783450584329E+003          19         208.6250000000000E+002          0              0              0
01.1101657547438E+003          20         218.7500000000000E+002          0              0              0
09.7548188098250E+002          21         228.8750000000000E+002          0              0              0
02.4218149178506E+003          22         248.6250000000000E+002          0              0              0
06.0041022028947E+002          23         258.6250000000000E+002          0              0              0
03.2325906666661E+002          24         268.8750000000000E+002          0              0              0
06.6706677314172E+002          25         278.7500000000000E+002          0              0              0
08.5585151318244E+002          26         288.7500000000000E+002          0              0              0
03.9856867755618E+002          27         298.8750000000000E+002          0              0              0
06.3446524586754E+002          28         308.7500000000000E+002          0              0              0
05.6327250240612E+002          29         318.8750000000000E+002          0              0              0
01.2466971804544E+002          30         327.2500000000000E+002          0              0              0
06.4383389489618E+002          31
```

（ヘッダー部分はバイナリである）

リスト８－９：DEM を表現する DBF ファイルの例

（ヘッダー部分）

```
└d
←ン|  。0                    SEQ          N
              XCOD         N  └            YCOD         N  └              ELEV         N  └
              1   -13987.500 -201697.500       156.050              2   -13985.000 -201697.500       156.000
3   -13995.000 -201695.000       156.800              4   -13992.500 -201695.000       156.510              5
-13990.000 -201695.000       156.320              6   -13987.500 -201695.000       156.220              7
-13985.000 -201695.000       156.090              8   -13982.500 -201695.000       156.000              9
-13980.000 -201695.000       156.000             10   -13977.500 -201695.000       156.000             11
-13975.000 -201695.000       156.000             12   -13972.500 -201695.000       156.000             13
-14005.000 -201692.500       157.840             14   -14002.500 -201692.500       157.550             15
-14000.000 -201692.500       157.260             16   -13997.500 -201692.500       156.970             17
-13995.000 -201692.500       156.740             18   -13992.500 -201692.500       156.600             19
-13990.000 -201692.500       156.500             20   -13987.500 -201692.500       156.420             21
-13985.000 -201692.500       156.340             22   -13982.500 -201692.500       156.230             23
-13980.000 -201692.500       156.130             24   -13977.500 -201692.500       156.040             25
-13975.000 -201692.500       156.000             26   -13972.500 -201692.500       156.000             27
```

-13970.000	-201692.500	156.000	28	-13967.500	-201692.500	156.000	29
-13965.000	-201692.500	156.000	30	-13962.500	-201692.500	156.000	31
-13960.000	-201692.500	156.000	32	-14012.500	-201690.000	158.600	33
-14010.000	-201690.000	158.310	34	-14007.500	-201690.000	158.020	35
-14005.000	-201690.000	157.730	36	-14002.500	-201690.000	157.420	37
-14000.000	-201690.000	157.050	38	-13997.500	-201690.000	157.000	39
-13995.000	-201690.000	156.890	40	-13992.500	-201690.000	156.780	41
-13990.000	-201690.000	156.720	42	-13987.500	-201690.000	156.670	43
-13985.000	-201690.000	156.600	44	-13982.500	-201690.000	156.490	45
-13980.000	-201690.000	156.360	46	-13977.500	-201690.000	156.260	47
-13975.000	-201690.000	156.140	48	-13972.500	-201690.000	156.000	49
-13970.000	-201690.000	156.000	50	-13967.500	-201690.000	156.000	51
-13965.000	-201690.000	156.000	52	-13962.500	-201690.000	156.000	53
-13960.000	-201690.000	156.000	54	-13957.500	-201690.000	156.000	55
-13955.000	-201690.000	155.950	56	-14022.500	-201687.500	159.640	57
-14020.000	-201687.500	159.350	58	-14017.500	-201687.500	159.060	59
-14015.000	-201687.500	158.770	60	-14012.500	-201687.500	158.480	61
-14010.000	-201687.500	158.100	62	-14007.500	-201687.500	157.660	63
-14005.000	-201687.500	157.230	64	-14002.500	-201687.500	157.130	65
-14000.000	-201687.500	157.090	66	-13997.500	-201687.500	157.030	67
-13995.000	-201687.500	157.000	68	-13992.500	-201687.500	156.990	69
-13990.000	-201687.500	156.950	70	-13987.500	-201687.500	156.920	71
-13985.000	-201687.500	156.850	72	-13982.500	-201687.500	156.720	73
-13980.000	-201687.500	156.570	74	-13977.500	-201687.500	156.420	75
-13975.000	-201687.500	156.220	76	-13972.500	-201687.500	156.030	77
-13970.000	-201687.500	156.000	78	-13967.500	-201687.500	156.000	79
-13965.000	-201687.500	156.000	80	-13962.500	-201687.500	156.000	81
-13960.000	-201687.500	156.000	82	-13957.500	-201687.500	155.980	83
-13955.000	-201687.500	155.880	84	-13952.500	-201687.500	155.880	85
-13950.000	-201687.500	155.880	86	-14030.000	-201685.000	160.400	87
-14027.500	-201685.000	160.110	88	-14025.000	-201685.000	159.820	89
-14022.500	-201685.000	159.530	90	-14020.000	-201685.000	159.200	91
-14017.500	-201685.000	158.770	92	-14015.000	-201685.000	158.340	93
-14012.500	-201685.000	157.900	94	-14010.000	-201685.000	157.470	95
-14007.500	-201685.000	157.250	96	-14005.000	-201685.000	157.210	97
-14002.500	-201685.000	157.180	98	-14000.000	-201685.000	157.130	99
-13997.500	-201685.000	157.060	100	-13995.000	-201685.000	157.000	101
-13992.500	-201685.000	157.000	102	-13990.000	-201685.000	157.000	103
-13987.500	-201685.000	157.000	104	-13985.000	-201685.000	157.000	105
-13982.500	-201685.000	156.880	106	-13980.000	-201685.000	156.700	107
-13977.500	-201685.000	156.500	108	-13975.000	-201685.000	156.310	109
-13972.500	-201685.000	156.110	110	-13970.000	-201685.000	156.000	111
-13967.500	-201685.000	156.000	112	-13965.000	-201685.000	156.000	113
-13962.500	-201685.000	156.000	114	-13960.000	-201685.000	156.000	115
-13957.500	-201685.000	155.940	116	-13955.000	-201685.000	155.870	117
-13952.500	-201685.000	155.810	118	-13950.000	-201685.000	155.810	119
-13947.500	-201685.000	155.810	120	-13945.000	-201685.000	155.800	121
-14040.000	-201682.500	161.440	122	-14037.500	-201682.500	161.150	123
-14035.000	-201682.500	160.860	124	-14032.500	-201682.500	160.570	125
-14030.000	-201682.500	160.280	126	-14027.500	-201682.500	159.880	127
-14025.000	-201682.500	159.440	128	-14022.500	-201682.500	159.010	129
-14020.000	-201682.500	158.570	130	-14017.500	-201682.500	158.140	131
-14015.000	-201682.500	157.710	132	-14012.500	-201682.500	157.370	133
-14010.000	-201682.500	157.330	134	-14007.500	-201682.500	157.300	135
-14005.000	-201682.500	157.260	136	-14002.500	-201682.500	157.220	137
-14000.000	-201682.500	157.150	138	-13997.500	-201682.500	157.080	139
-13995.000	-201682.500	157.010	140	-13992.500	-201682.500	157.000	141
-13990.000	-201682.500	157.000	142	-13987.500	-201682.500	157.000	143
-13985.000	-201682.500	157.000	144	-13982.500	-201682.500	156.960	145
-13980.000	-201682.500	156.780	146	-13977.500	-201682.500	156.590	147
-13975.000	-201682.500	156.390	148	-13972.500	-201682.500	156.200	149
-13970.000	-201682.500	156.000	150	-13967.500	-201682.500	156.000	151
-13965.000	-201682.500	156.000	152	-13962.500	-201682.500	156.000	153
-13960.000	-201682.500	156.000	154	-13957.500	-201682.500	155.920	155
-13955.000	-201682.500	155.840	156	-13952.500	-201682.500	155.770	157
-13950.000	-						

（ヘッダー部分はバイナリである）

③DXF 形式

テキスト・データであり、解説書によりデータ形式も公開されていたことから、景観シミュレーション・システムの開始以前から、異なるシステム間でのデータ交換に用いられてきた形式である。主に２次元図形のために使用されていたが、データ形式上は初期から Z 座標値も同様に記述できる形式となっていた。上述のように、景観シミュレーションの現場適用を開始した頃の DXF データは、線分の集合であった。しかし、単純に線のデータに変換し、ワイヤーフレームとして表示するだけではリアリティが乏しいため、端の座標値が一致する線分のペアを照合する処理を行い、面→立体を復元するような処理を加えた。その後、DXF 形式の拡張が行われ、面やソリッドなども表現される形式に拡張された。

リスト８－１０：DXF 形式の例（図面を構成する線分の記述）

```
  5
7161
330
1F
100
AcDbEntity
  8
現況
100
AcDbPolyline
 90
        2
 70
     0
 43
0.0
 10
-9321.3590752946
 20
22888.38099993517
 10
-9323.771871896211
 20
22891.86439117691
  0
LWPOLYLINE
  5
7162
```

リスト８－１１：DXF の 3DSOLID 開始部分

```
  0
3DSOLID
  5
8F
330
8B
100
AcDbEntity
  8
Furniture_2
100
AcDbModelerGeometry
 70
     1
  1
koo nnk n o
  1
ni ^ *+0::,4 ^ *+0¥^ [ ng ^ LR nmqoqoqjgmm QK o
```

```
   1
mjqlfffffffffffffff fqfffffffffffffffj:rooh n:rono
   1
=0;& {n {m {rn {rn |
   1
-:9@)+r:&:r>++-6= {rn {rn {rn {o {l {k |
   1
3*2/ {j {rn {i {o |
   1
:&:@-:961:2:1+ {rn j 8-6;    n l +-6 n k ,*-9 o l >;5 o k 8->; o f /0,+<7:<4 o k ,+03 oqkgnmgjjnmkkhljhng k 1+03 lo k ;,63
o g 93>+1:,, o h /6'>-:> o k 72>' o i 8-6;>- o j 28-6; looo j *8-6; o j )8-6; o no :1;@96:3;, |
   1
):-+:'@+:2/3>+: {rn l o n g |
   1
-:9@)+r:&:r>++-6= {rn {rn {rn {m {l {k |
   1
,7:33 {h {rn {rn {g {rn {m |
   1
-:9@)+r:&:r>++-6= {rn {rn {rn {i {l {k |
   1
9><: {f {no {nn {i {rn {nm 90-(>-; ,6183: |
   1
92:,7r:&:r>++-6= {rn {nl {rn {g |
   1
```

（ASCII で表示可能ではあるが、コード変換されており、人には読みづらい）

リスト８－１２：3DSOLID 部分（読みやすくコード変換したもの）

```
-opcode:5 operand:1F7
-opcode:330 operand:1F
-opcode:100 operand:AcDbEntity
-opcode:8 operand:A-Counter_01
-opcode:100 operand:AcDbModelerGeometry
-opcode:70 operand:     1
-opcode:1 operand:400 303 1 0
-opcode:1 operand:16 A utodesk A utoCA D 18 A SM 12.0.0.5822 NT 0
-opcode:1 operand:25.399999999999999 9.9999999999999995e-007 1e-010
-opcode:1 operand:body $1 $2 $-1 $-1 #
-opcode:1 operand:ref_vt-eye-attrib $-1 $-1 $-1 $0 $3 $4 #
-opcode:1 operand:lump $5 $-1 $6 $0 #
-opcode:1 operand:eye_refinement $-1 5 grid 1 3 tri 1 4 surf 0 3 adj 0 4 grad 0 9 postcheck 0 4 stol
2.0501470565795898 4 ntol 30 4 dsil 0 8 flatness 0 7 pixarea 0 4 hmax 0 6 gridar 0 5 mgrid 3000 5 ugrid 0 5 vgrid 0
10 end_fields #
-opcode:1 operand:vertex_template $-1 3 0 1 8 #
-opcode:1 operand:ref_vt-eye-attrib $-1 $-1 $-1 $2 $3 $4 #
-opcode:1 operand:shell $7 $-1 $-1 $8 $-1 $2 #
-opcode:1 operand:ref_vt-eye-attrib $-1 $-1 $-1 $6 $3 $4 #
-opcode:1 operand:face $9 $10 $11 $6 $-1 $12 forward single #
-opcode:1 operand:fmesh-eye-attrib $-1 $13 $-1 $8 #
-opcode:1 operand:face $14 $15 $16 $6 $-1 $17 reversed single #
-opcode:1 operand:loop $-1 $-1 $18 $8 #
-opcode:1 operand:plane-surface $-1 128.44875040196303 534.99999999999989 38.999999999999964 1
4.4583073616319837e-015 0 0 0 -1 forward_v I I I I #
-opcode:1 operand:ref_vt-eye-attrib $-1 $-1 $9 $8 $3 $4 #
-opcode:1 operand:fmesh-eye-attrib $-1 $19 $-1 $10 #
-opcode:1 operand:face $20 $21 $22 $6 $-1 $23 reversed single #
-opcode:1 operand:loop $-1 $-1 $24 $10 #
-opcode:1 operand:plane-surface $-1 63.050873611198085 561.1710926275457 31.499999999999964 0 -1 0 0 0 -1
forward_v I I I I #
-opcode:1 operand:coedge $-1 $25 $26 $27 $28 reversed $11 $-1 #
-opcode:1 operand:ref_vt-eye-attrib $-1 $-1 $14 $10 $3 $4 #
-opcode:1 operand:fmesh-eye-attrib $-1 $29 $-1 $15 #
-opcode:1 operand:face $30 $31 $32 $6 $-1 $33 reversed single #
-opcode:1 operand:loop $-1 $-1 $34 $15 #
-opcode:1 operand:plane-surface $-1 63.050873611198085 604.1710926275457 31.499999999999964 1 0 0 0 0 -1
forward_v I I I I #
----------中略----------
-opcode:1 operand:spline-surface $-1 forward { sweepsur normal
```

-opcode:1 operand: ellipse 405.554560567301 304.88581986905791 63.000000000000171 0.30436101391021336 0.68844839424440707 0.65833364008976836 -0.91460658855749821 0.40434488764817533 0 1 I I
-opcode:1 operand: intcurve forward { lawintcur nubs 3 open 60
-opcode:1 operand: 0 3 0.44178646691106466 1 0.88357293382212931 1 1.3253594007331939 1 1.7671458676442586 1
-opcode:1 operand: 2.2089323345553233 1 2.6507188014663878 1 3.0925052683774523 1 3.5342917352885173 1 3.9760782021995822 1
-opcode:1 operand: 4.4178646691106467 1 4.7123889803846897 1 5.0069132916587327 1 5.3014376029327757 1 5.5959619142068187 1
-opcode:1 operand: 5.8904862254808616 1 6.1850105367549046 1 6.4795348480289476 1 6.7740591593029915 1 7.0685834705770345 1
-opcode:1 operand: 7.3631077818510775 1 7.6576320931251214 1 7.9521564043991644 1 8.2466807156732074 1 8.5412050269472495 1
-opcode:1 operand: 8.8357293382212934 1 9.1302536494953372 1 9.4247779607693793 1 9.7193022720434215 1 10.013826583317465 1
-opcode:1 operand: 10.308350894591509 1 10.602875205865551 1 10.897399517139593 1 11.191923828413637 1 11.486448139687681 1
-opcode:1 operand: 11.780972450961723 1 12.075496762235765 1 12.370021073509809 1 12.664545384783853 1 12.959069696057895 1
-opcode:1 operand: 13.253594007331939 1 13.548118318605983 1 13.842642629880025 1 14.137166941154069 1 14.431691252428113 1
-opcode:1 operand: 14.726215563702155 1 15.020739874976199 1 15.315264186250243 1 15.609788497524285 1 15.904312808798329 1
-opcode:1 operand: 16.198837120072373 1 16.493361431346415 1 16.787885742620457 1 17.082410053894499 1 17.376934365168545 1
-opcode:1 operand: 17.671458676442587 1 17.965982987716629 1 18.260507298990674 1 18.555031610264717 1 18.849555921538759 3
-opcode:1 operand: 405.554560567301 304.88581986905797 63
-opcode:1 operand: 405.65221681736654 305.10671310238553 63.211230714847716
-opcode:1 operand: 405.74797887073635 305.6375143543778 63.633692144543232
-opcode:1 operand: 405.45605280480152 306.52339572031406 64.267384289086323
-opcode:1 operand: 404.69486069646882 307.25895351640924 64.901076433629299
-opcode:1 operand: 403.55969335308237 307.60171125924455 65.534768578171139
-opcode:1 operand: 402.26571770359351 307.37246243697268 66.168460722715025
-opcode:1 operand: 401.1073709962269 306.51155444188146 66.802152867258457
-opcode:1 operand: 400.39295320189916 305.11022584100664 67.435845011801362
-opcode:1 operand: 400.36894815074999 303.40731665059474 68.069537156344381
-opcode:1 operand: 401.15170267992755 301.74921209210254 68.703229300886207
-opcode:1 operand: 402.51364372263788 300.65672676486577 69.266511207147616
-opcode:1 operand: 404.03520727994658 300.19351872432526 69.759382875125695
-opcode:1 operand: 405.45557759916085 300.20023535951486 70.181844304820984
-opcode:1 operand: 406.86551604516796 300.64395054117296 70.604305734516359
-opcode:1 operand: 408.12899878221697 301.51686683017959 71.026767164211662
-opcode:1 operand: 409.11458464124564 302.7688504113006 71.449228593907023
-opcode:1 operand: 409.70842758629652 304.30943123421179 71.871690023601587
-opcode:1 operand: 409.82663979600608 306.01414027437181 72.294151453296735
-opcode:1 operand: 409.42532012284903 307.73451208187697 72.716612882992891
-opcode:1 operand: 408.50724284423239 309.31101574336697 73.13907431268845
-opcode:1 operand: 407.12435522711098 310.58777243993404 73.561535742383739
-opcode:1 operand: 405.37563940633402 311.42771176211107 73.983997172079086
-opcode:1 operand: 403.40032001171011 311.72671137835005 74.406458601774418
-opcode:1 operand: 401.36685373902827 311.42530149287592 74.82892003146975
-opcode:1 operand: 399.45856462391049 310.51669029873131 75.251381461164328
-opcode:1 operand: 397.85714983850573 309.05016920821629 75.673842890860243
-opcode:1 operand: 396.72553731445726 307.12936171619185 76.096304320555831
-opcode:1 operand: 396.19170020964725 304.90525193901954 76.51876575025112
-opcode:1 operand: 396.3350076314486 302.56442505629582 76.941227179946495
-opcode:1 operand: 397.17651343184207 300.31342532643623 77.363688609641798
-opcode:1 operand: 398.67426707515682 298.36054201424952 77.786150039337144
-opcode:1 operand: 400.72429808165299 296.89662855442089 78.208611469031723
-opcode:1 operand: 403.16741561537413 296.07671384695323 78.631072898727638
-opcode:1 operand: 405.80142316557419 296.00415745853752 79.053534328423211
-opcode:1 operand: 408.39782512632348 296.71892803878336 79.475995758118543
-opcode:1 operand: 410.72164735377123 298.19125729094952 79.898457187813875
-opcode:1 operand: 412.55265248138664 300.32146609969209 80.32091861750925
-opcode:1 operand: 413.70603859061049 302.94621354109518 80.743380047204525
-opcode:1 operand: 414.0506895246541 305.85083172786472 81.165841476899118
-opcode:1 operand: 413.52320402650429 308.78683337686164 81.588302906594265

```
-opcode:1 operand: 412.13625976927 311.49316871170731 82.010764336290435
-opcode:1 operand: 409.98034188743304 313.7194133557166 82.433225765985981
-opcode:1 operand: 407.21844426462906 315.24882944936724 82.85568719568127
-opcode:1 operand: 404.07398513451432 315.91918525846916 83.278148625377412
-opcode:1 operand: 400.81280954497709 315.63935477650904 83.700610055072133
-opcode:1 operand: 397.72072156708612 314.40004142577482 84.123071484767223
-opcode:1 operand: 395.07844442081432 312.27745466105318 84.545532914462626
-opcode:1 operand: 393.13620167492445 309.42937517095169 84.967994344157958
-opcode:1 operand: 392.09021562635678 306.08372119097362 85.390455773853319
-opcode:1 operand: 392.06331495390066 302.52041506459977 85.81291720354865
-opcode:1 operand: 393.09153646756272 299.04798296714284 86.235378633243243
-opcode:1 operand: 395.11811768668883 295.97684216960005 86.657840062939158
-opcode:1 operand: 397.99564768870619 293.59158861196937 87.080301492633964
-opcode:1 operand: 401.49642675032339 292.1247557154436 87.502762922328998
-opcode:1 operand: 405.33034372934014 291.73445346365742 87.925224352024458
-opcode:1 operand: 409.16888075462037 292.48801457446581 88.347685781720543
-opcode:1 operand: 412.67326230920833 294.35329189479137 88.770147211416088
-opcode:1 operand: 415.52433681182634 297.19860656794492 89.19260864111142
-opcode:1 operand: 417.45155647489776 300.80159432157461 89.615070070806951
-opcode:1 operand: 417.9894564009906 303.51137308302958 89.896711023936078
-opcode:1 operand: 418.054560567301 304.88581986905797 90.037531500501913
-opcode:1 operand: 0.001
-opcode:1 operand: null_surface
-opcode:1 operand: null_surface
-opcode:1 operand: nullbs
-opcode:1 operand: nullbs
-opcode:1 operand: I I
-opcode:1 operand: 0
-opcode:1 operand: 0
-opcode:1 operand: 0
```

（パラメトリックな原始図形、メッシュ状の曲面等が表現されている）

④MiniCAD 形式、現在の VectorWorks 形式

景観シミュレーションを実際の事業における景観検討に適用し始めた初期には、システム自体のモデリング機能がまだ貧弱であったため、複雑な建物や構造物のデータは、商用 CAD でデータを作成した上で、貿易コンバータで LSS-G 形式に変換し、背景写真との合成や、地形の上への配置などの操作を行っていた。AutoCAD、MicroStation と並んで、三次元 CAD として、MiniCAD もしばしば使用された。

リスト８－１３：MiniCad データ例（部分）

```
Projection(0,0,247.904,-123.952,123.952,123.952,-123.952);
SetView(#0。0' 0",#0。0' 0",#0。0' 0",0,0,0);

{End of Layer Characteristics}

{Object Creation Code}

NameClass('一般');
PenSize(1);
PenPat(2);
ArrowHead(12);
ArrowSize(10);
MoveTo(-13000,6000);
LineTo(-13000,-6000);
MoveTo(-7000,6000);
LineTo(-7000,-6000);
MoveTo(-2800,6000);
LineTo(-2800,-6000);
MoveTo(2200,6000);
LineTo(2200,-6000);
MoveTo(8600,6000);
LineTo(8600,-6000);
MoveTo(13600,6000);
```

```
LineTo(13600,-6000);
MoveTo(-11000,6000);
LineTo(-11000,-6000);
MoveTo(-9000,6000);
LineTo(-9000,-6000);
MoveTo(-5000,6000);
LineTo(-5000,-6000);
MoveTo(-5000,6000);
----------中略----------
Poly(
-3075.0001,-368.5,
-2525,-368.5,
-2525.0001,-1518.4999,
-3075,-1518.5
);
Rect(13250,1675,13950,-1675);
BeginXtrd(0,350);
Arc(9149.9956,751.6639,9153.574,748.0855,#175。59' 6",#183。50' 9");
Arc(8049.9971,1300.0029,9153.5765,196.4235,#0。10' 20",#179。38' 35");
Arc(8049.9971,-196.4235,9153.5765,-1300.0029,#180。11' 5",#179。38' 35");
MoveTo(8050,749.9897);
LineTo(8050,-749.9982);
MoveTo(9153.574,749.8691);
LineTo(9153.574,-749.8719);
EndXtrd;
Rotate3D(#0。0' 0",#0。0' 0",#0。0' 0");
Move3D(0,0,0);
```

（注：1995 年頃のデータである。原型は XY プロッタに対するコマンドのように見える。）

８－４．データ活用による景観シミュレーションの実例

景観法が成立した 2004 年には SXF 形式による電子納品や、５ｍメッシュによる高精度数値地図が利用可能となった。2006 年からは、ステレオ衛星画像が実用的な価格で利用可能になった。そこで、これらを例として、データ活用のための機能開発と、データ作成の事例を具体的に解説する。

（１）高精度数値地図

開発に着手した’93 年当時、国土地理院において建設技術評価制度が行なわれ、ステレオ空中写真自動解析技術の検定のため、メッシュで標高を表現した指定形式での解析結果 DEM データの提出が条件とされた。この形式の地形データは、各社・各システムがコンバータを有していたことから、これをハブとする景観検討用コンバータを作成したことが出発点となった。その後、国土地理院から提供開始された 50ｍメッシュ数値地図や、いくつかの景観検討現場での測量結果を利用するために、適宜拡充を行った。当時、「DEM」と称するデータには、それぞれ微妙な形式の違いが存在していた。

2003 年度には、レーザー・スキャナーの測量結果に基づく、５ｍメッシュの数値地図が、大都市圏を中心に提供開始されたため、福岡地域に関して未公開データの景観検討の応用可能性を試した。

５ｍメッシュの数値地図は、「.1em」の拡張子を持つ、従来の数値地図とは形式が異なるテキストデータで、一つのファイルが、東西 2,000m、南北 1,500m の領域（1:2,500 基本図の範囲）に対応し、横 400 ドット分の標高値が１行で記述され、300 行で一つのファイルと

なっている。

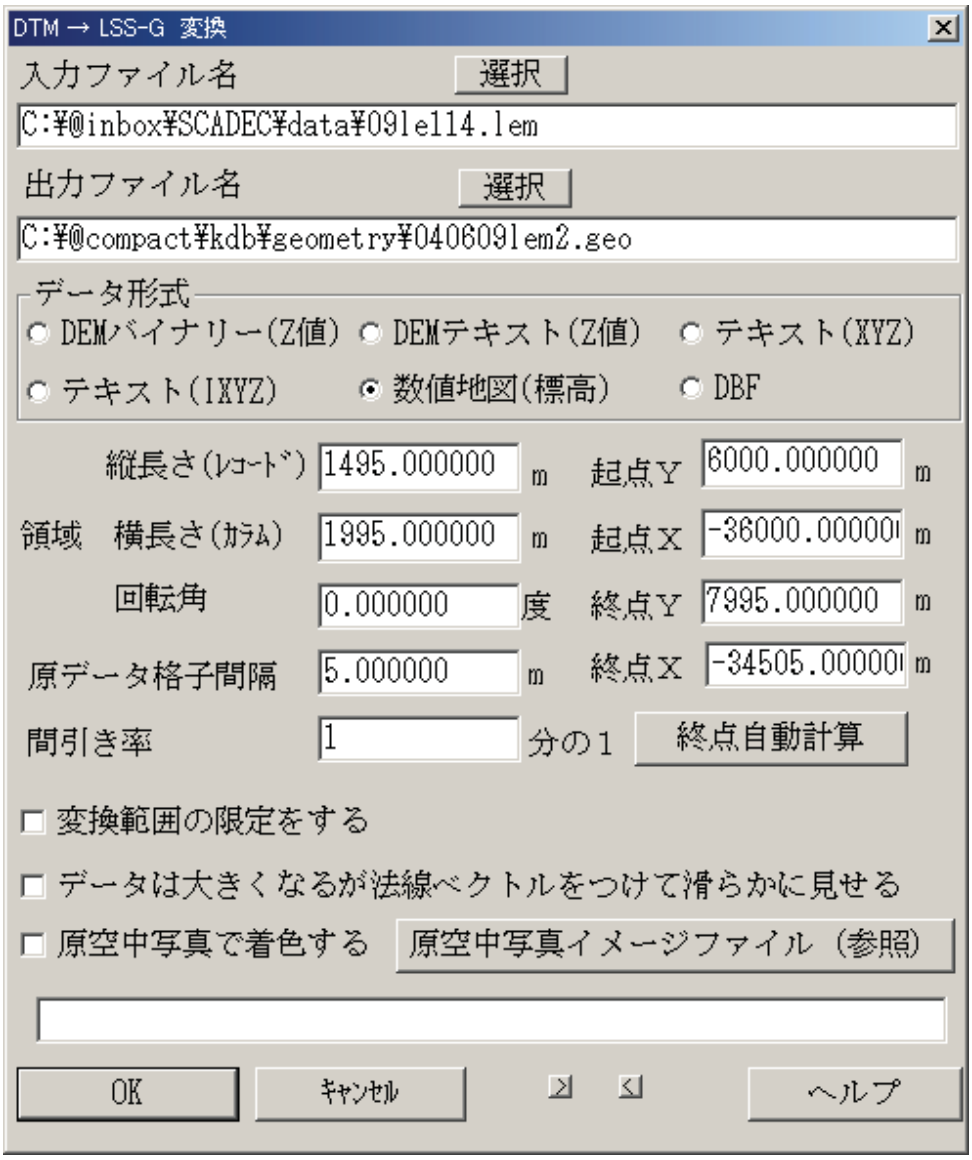


図８－２：数値地図変換のパラメータ設定画面

リソース：IDD_DIALOG21　ハンドラ：Dialog121()（貿易.c）

標高値は、10 倍された整数値となっている。水面等、標高が定まらない領域に関しては、-9999 の数値が記入されている。各図郭ファイルに一つのインデックスファイルが添付され、測量年月日、四隅の緯度経度、国家座標系による範囲その他の諸元が記述されている。

これを三次元に変換するためには、メッシュを構成する頂点から、地形の曲がり方向に応じて、凹凸が少なくなる向きにグリッドを斜め分割した三角形群を生成した。標高不明の点に関しては、当面、標高ゼロとして処理した。

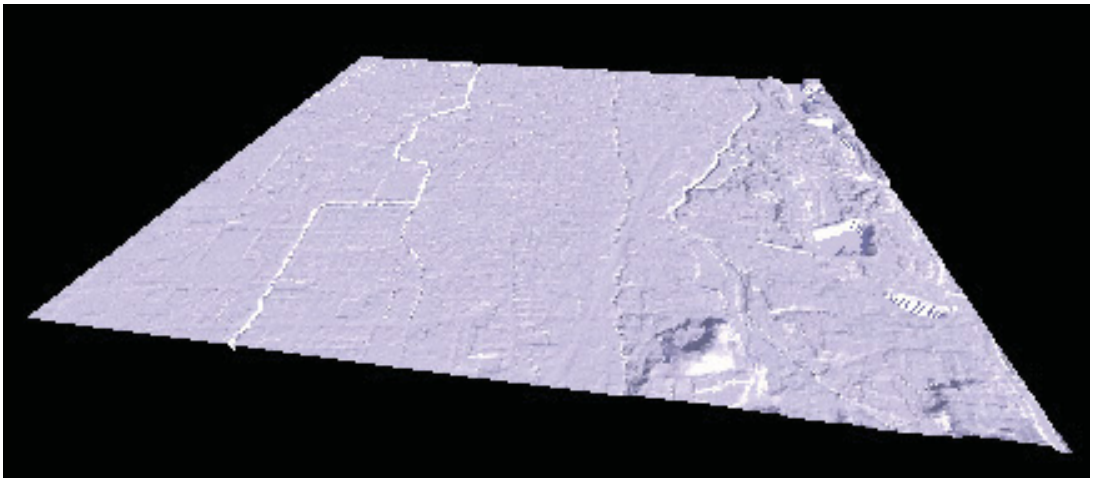


図８－３：高精度数値地図の変換結果（福岡市）

（２）SXF (Standard CAD data Exchange format in japanese Construction field)

電子納品のデータは(SXF)は、JACIC に設置された検討委員会で CALS の一環として検討が進められているものであり、まだ基本的に二次元 CAD の形式であって、2007 年度から三次元の検討結果の一部（道路中心線など）が公開開始されている状況である（文献３７）。

2004 年に道路局の依頼で、利用可能性検討を行なった福岡西道路の場合、電子納品成果は配置図(1:25,000)、平面図(1:500)、横断図(1:100)、縦断図(1:1,000)、交差点図(1:200)、その他各種詳細図の多数ファイル群から構成されていた。三次元形状を生成するために、断面図と中心線の軌跡を抽出し、これらから「掃引体」として路面の立体形を作成した。

① 基本的な変換方法

SXF は基本的にテキスト形式であり、各図面は複数のレイヤーを有する。これらを利用するために、景観シミュレータの新たな「外部関数」として、scadec2.exe を作成した。パラメータ指定部では、ユーザーがファイルを指定した段階でファイルの中に含まれているレイヤー名称を一覧表示し、必要部分のみ選択できるようにした。各ファイルにはオブジェクト毎に記述寸法の単位の定義（多く mm）が行われているため、出力する際には、図面上の図形ではなく、三次元空間に、高さゼロの平面図形を、原寸で生成した。

② 断面形

高架道路等の部分は、断面形と中心線軌跡を求め、掃引することで三次元形状を生成する。

まず断面図には、橋梁部のように構造物の断面が描かれている部分と、橋脚部のように見え掛りが描かれている部分があるが、色や線種から識別することができなかったため、手動で抽出した。橋梁部などの断面に関しては、景観検討に必要な外形を頂点列として拾い出し、道路断面ファイルとして保存する。橋脚部に関しては、外形を抽出した上で、高さ(厚さ)を与えて直ちに三次元形状を生成し、配置するための部品として保存する。

③ 中心線軌跡

中心線高さは縦断図という特殊な形式の図面で記述されている。水平方向は原寸、高さ方向は 10 倍された縮尺で図中の座標値が与えられている。路面に関しては、直線部（勾配

一定の区間）は長い一つの線分として、また勾配の変化点付近は、１ｍ単位の細分された線群で曲線が近似されている。そこで、中心線を記述するレイヤーだけを変換した線分群の接続関係を解析し、一つながりのポリラインに変換するような処理機能を作成した。縦断図は、区間全体をいくつかの図に分割して表現されているために、更に相互接続するために平行移動を行った上で合体し、全体区間の軌跡を作成しファイルとして保存する。

④　道路の形状生成

以上の手順で作成した断面系を中心線軌跡で掃引して、高架部の形状を作成した。

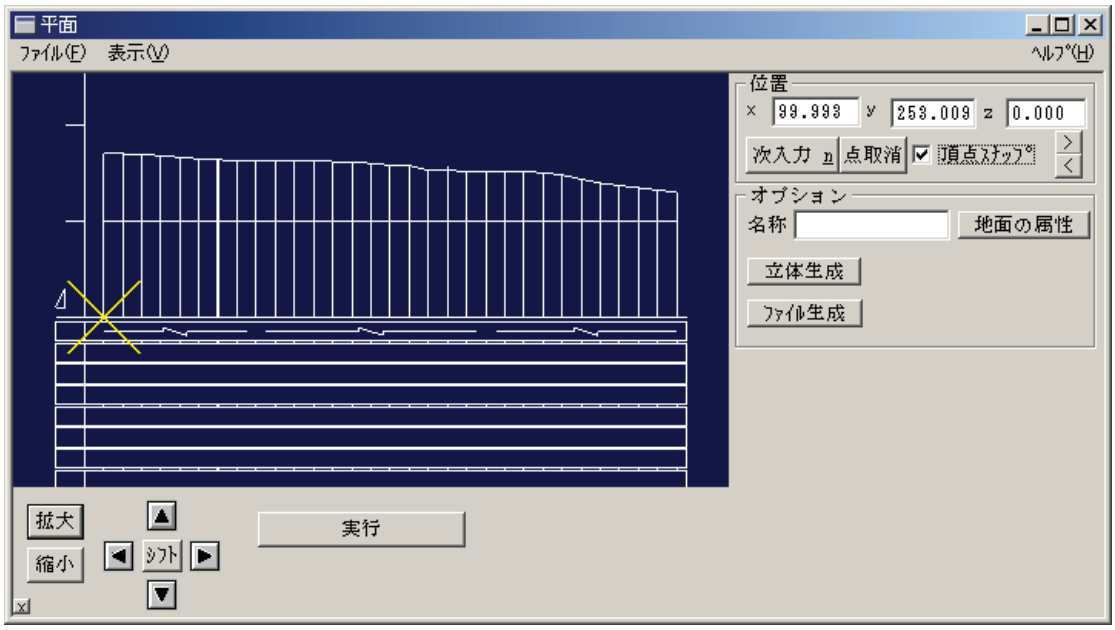


図８－４：縦断図の変換結果（黄×印は手作業による原点の指定）

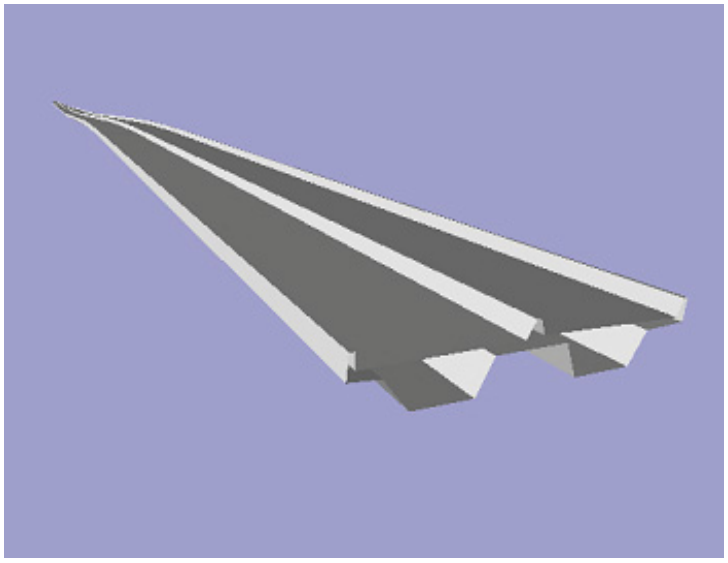


図８－５：断面形と中心線軌跡から道路高架部の形状を掃引し生成

⑤ 地形と構造物の合成

数値地図から作製した地形の上に、道路の高架部と、橋脚を配置し、点景を加えて景観画像とした（図８－６）。全工程を操作手順書にまとめた（文献３０～３３）。

図８－６：地形＋道路＋点景

（３）ステレオ衛星画像

途上国における地球温暖化対策の可能性を検討するために、1999 年からインドネシアをフィールドとして調査研究を進める中、2006 年から、「だいち(ALOS)」衛星(PRISM センサー)が利用可能となった。公開された地形図の精度が低く、空中写真の利用が軍事的理由等により制約されている地域での調査研究や計画策定のためには、衛星画像から得られた地形データは極めて有用である（文献２８、２９）。

バンドン市及びチレボン市の既存の計画的に開発された団地（各５ha 規模）を対象に、衛星画像を用いながら現況調査を行なった上で、代替的な将来像の設計検討を行った。その結果を三次元データとして表現し、建築・都市計画の専門家以外の地元代表、地方政府、学識経験者を招いたワークショップにおいてプレゼンテーションし、質疑応答を行ない、実現可能性等に関して討論を行なう素材とした（図８－７）。

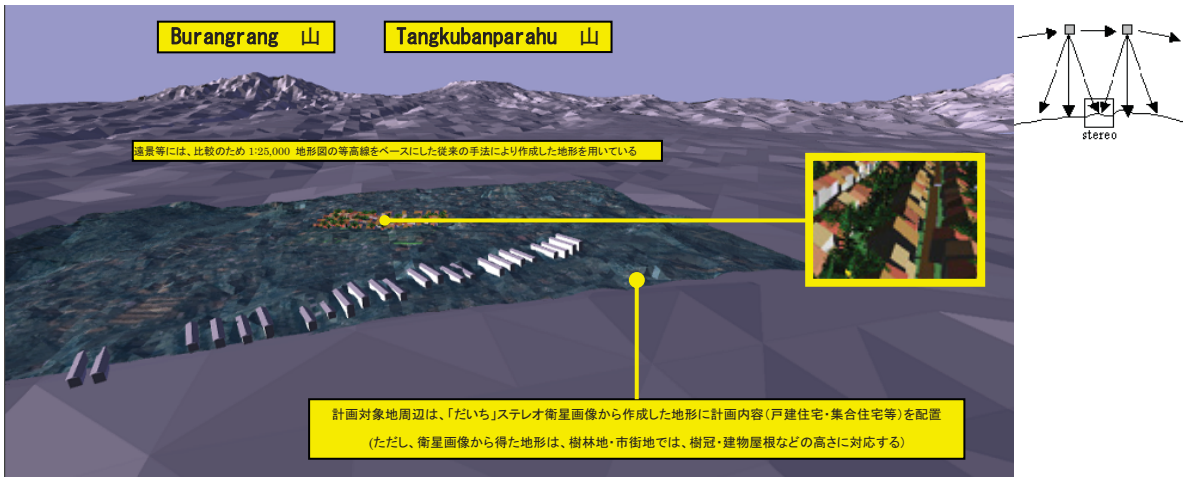


図８－７：バンドン市における衛星画像を用いた計画地周辺の地形データ作成

８－５．各種三次元データ形式による出力

三次元データを各種ファイル形式で出力するためには、メモリ上の全地物データにアク

セスする必要があるため、外部関数として実装することはできない。このため、現在までに対応している形式は、基幹部分のビルドに含まれており、ファイル保存に際して、ファイル拡張子を指定することにより、市街地延焼シミュレーション形式、DXF 形式、及び VRML 形式に保存することを可能としている。これらはアプリケーション・ライブラリ関数により実装しているため、sim.exe 基幹部分のビルドに含まれている。

Ver.2.09 より提供しているプラグイン DLL の拡張方法を用いることにより、内部データ構造にアクセスできるため、これ以外のファイル形式として出力する機能の追加は、基幹部分のバージョンを枝分かれさせることなく実現できる。

８－６．画像データの入出力

テクスチャや背景イメージとして使用する画像ファイルに関しては、冒頭に述べたように、開発に着手した当時の SGI 形式（BMP 形式）から出発したが、現在では流通やデータ交換には殆ど使用されない画像形式となっている。しかし、この形式は単純であり、また JPEG 形式のように圧縮過程で情報量を落とすような処理を行っていないため、保存性・再現性の高いデータ形式である。SGI 形式の入出力は IMGSGI ライブラリにより提供している。

その後、テクスチャ等にデジカメ画像等を使用する必要や、表示画像を資料作成などに用いる必要が生じたため、公開ソースコードを組み込んで入出力機能を提供した。JPEG 形式の入力機能は、JPEG ライブラリにより提供している。但し、前景や、樹木のテクスチャに用いる、透明部分を有するテクスチャを表現することができない。

この他、入力機能として、BMP 形式、TIFF（GeoTiff）形式を読み込むことができる。TIFF 形式は、圧縮方法を含む多様なサブセットを含む形式であるが、実際に使用されているのは仕様全体の一部である。実装しているソースコードは独自に起こしたものであるが、仕様全体には対応していない。しかし、衛星画像などを読み込むような目的には十分と考えている。

移動経路を指定した視点移動の結果を動画として保存する際に、AVI 形式をサポートしている。但し、巨大なファイルが形成される傾向にある。

SGI 形式の画像を、流通性の高い他のファイル形式に変換する機能を、貿易コンバータで提供している。

９．バックアップ・アンドゥ

９－１．概要

多くの処理において、バックアップ・アンドゥの機構を用意した。ユーザーは一つの編集ダイアログを開き、ある操作を行う。その結果が期待しているものとは異なる場合に、データや環境設定を元に戻し、あるいは条件を変えて再び操作を行う。このようなバックアップ・アンドゥの機能が用意されていていなければ、ユーザーは個々の操作に際して、必ず成功しなければならない、というプレッシャーに晒される。実物としての模型製作や実際の建設とは異なって、コンピュータ上のデータとしてシミュレーションを行うことの効用の一つは、やり直しのコストが低いという点にあることは間違いない。

しかし、バックアップ・アンドゥの処理ためには、メモリやファイル等のリソースを必要とする上、バックアップ・アンドゥ系自体が信頼性の低いものである場合（例えばバックアップやアンドゥの処理中にシステムダウンを生じる最悪の場合等）、その存在意義には疑問符が付くこととなる。

これまでに試行され、実装されたバックアップ・アンドゥには、Ver.2.09 で不採用とした方法を含め、以下のものがある。

（1）ヒストリー方式

景観シミュレータの開発初期においては、一括してバックアップ・アンドゥを処理するために、ヒストリー機能が構想され、一定レベルまで開発を進めた。これは、ユーザーの操作をコード化し、一連の操作の記録として小さな外部テキストファイルに保存するという機能である。編集操作中にシステムダウンが生じた場合には、少し時間はかかるが、ヒストリーとして記録されているデータを忠実に再現すれば、システムダウンを生じる直前までの編集操作を再現することができる。

但し、この方法には以下の２点で難点があったため、現在では基幹部分のビルドから外している。

①新たな機能を追加する度に、そのコーディング規則を定め、各処理ダイアログに組み込むと共に、外部の記録ファイルのエンコード・デコード機能を拡充しなければならない。

②ヒストリー機能自体に内在するバグが、システム障害の原因となる場合がある。

理由②は、基本的なライブラリ関数にもまだバグが多く残されていた時期であったことによる。現時点で、ヒストリー機能を再検討すると共に、通信的環境におけるコラボレーションのための同期的・協調的な動作など、新たな使用方法に向けた可能性がある。

編集操作に際しては、ヒストリー機能により、保存する意味のある操作自体がコーディングされ、ヒストリーとして累積的に記録される。ある処理を行っている最終にシステムダウンが生じたような場合には、処理を開始した時点でのシステム状態（あるファイルがロードされた状態）を再現し、そこからヒストリーとして記録された操作を再現することにより、リスクに際限を設けることができる。

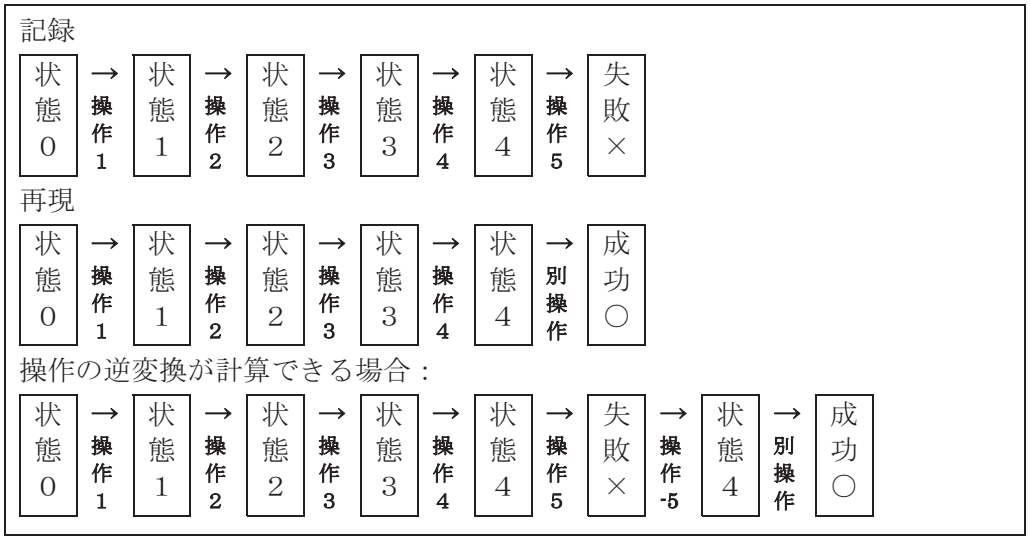


図９－１：ヒストリー方式：ダイアグラム

（２）モデルのファイル保存とアンドゥ

全地物データに大きな改変を加える複雑な処理（地形の自動的な再編集など）に際しては、その前の状態をファイルに保存しておき、結果に不満であれば復元するような仕組みを用意した。

これは、U３ライブラリの機能として実現している。このパターン化した処理においては、バックアップ構造体に、処理前の状態を保存する一時的ファイルの名称を記録する。復元処理においては、この一時的ファイルを復元する。

（３）メモリ上での保存とアンドゥ

バックアップすべき情報の量が、メモリ使用量による制約に影響しない程度に十分小さければ、メモリ上にバックアップ情報を保存しておき、アンドゥに利用することができる。

上記の（２）及び（３）の基本的な処理の流れは下記のようなものとなる。

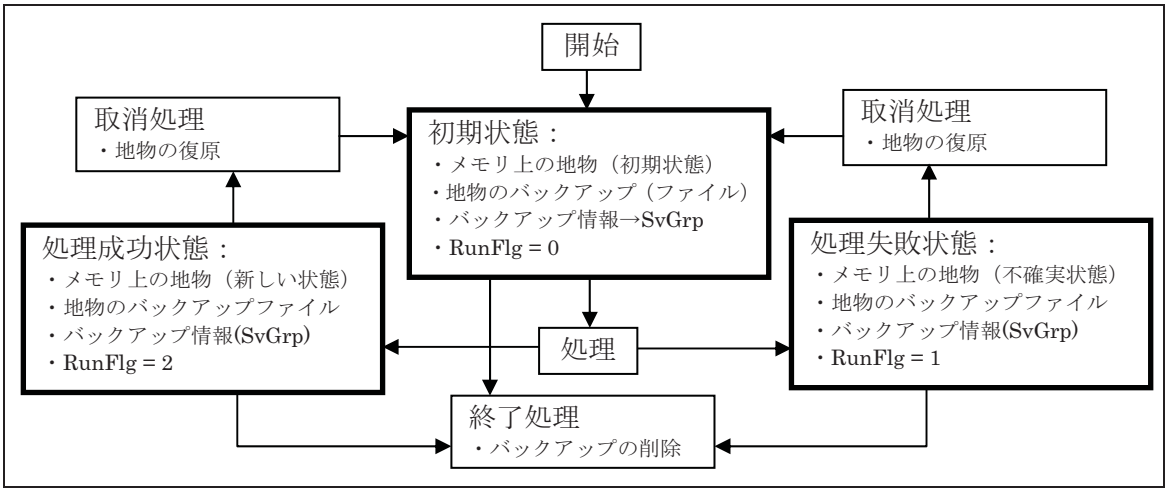


図９－２：地物のバックアップファイルによるアンドゥ処理

９－２．モデル全体のバックアップとアンドゥ

前述（２）で述べたモデルの保存とアンドゥのために、Ｕ３ライブラリの中で関数を用意してある。これは、基幹部分の配置コマンドで用いている方法であるが、同じライブラリ関数をプラグインＤＬＬからも利用可能である。

例えば、園路生成の場合では、RunFlag は、enroOpt.RunFlg として、SvGrp は、enroParam.SvGrp として実装している。このダイアログにおいては、ユーザーが諸条件を設定した後に、地形を大きく改変するような図形演算処理を行っている。このために、編集処理結果（失敗する場合）を見て、やり直すことを想定し、キャンセル・ボタンによって復原・終了するのではなく、復原の後に、再試行することを可能としている。終了処理（出口）は、終了ボタンに統一している。従って、終了ボタンが押されるのは、①何も処理が行われないまま、直ちに終了する場合、②処理が行われ、成功している状態、③処理が行われ、失敗している状態がありうる。

具体的に、以下のように処理している。

① 編集開始時点

CMyOrthoVW::OnPaint() において

giFirst フラグが－１（初回表示）または０（二回目以降、編集機能が起動された）の場合に、編集が開始されたことを認識し、以下のバックアップ処理を実行する。

enroParam.SvGrp = u3SaveGroupFamily(*enroParam.OrgGrp);

これにより、一時的なファイルが作成され、ファイル名等の情報が、SvGrp に記録される。

RunFlg を０にセットする。

② 実行が押された場合

CParkRoadWnd::OnBtnRun()の中で、編集処理が実行される。

RunFlg を調べ、初期状態でない場合には処理を行わない。

enroOpt.RunFlg を１にセットし、実行中であることを示した上で処理を開始する。

処理が正常に終了した場合には、RunFlg を２にセットする。

エラーで終了した場合、地物データは処理前のものとも、正常に処理された場合のものとも異なっている可能性がある。

③ 編集が行われた後、キャンセルが押された場合

1) enroOpt.RunFlg を０にセットする（実行前の状態）。

2) ルート・グループ（オリジナルとは異なる編集されたデータ）を解放する。

3)*enroParam.OrgGrp = u3RestoreGroupFamily(TRUE, enroParam.SvGrp);

この操作で、バックアップファイルは削除され、オリジナルのモデルが復元される。

SvGrp のメモリブロックは解放される。

4) enroParam.SvGrp = u3SaveGroupFamily(*enroParam.OrgGrp);

バックアップファイルが再び作成され、新たに構築された SvGrp に記録される

5) 復元されたデータを、所定の場所にセットする（表示が編集前の状態に戻る）

```
if(GetLssFileType() == K_LSS_S) {//ジオメトリを編集中の場合
        scn = s3GetScene(GetSceneCurrentIndex());
        scn->mdl->root[0] = *enroParam.OrgGrp;
        g3SetRootGroup(scn->mdl->num, enroParam.OrgGrp);
} else if(GetLssFileType() == K_LSS_G) {//シーンを編集中の場合
        g3SetRootGroup(1, enroParam.OrgGrp);
}
```

④ 終了ボタンが押された場合

CParkRoadWnd::OnDestroy()の中で、

enroParam.SvGrp を解放する。

解放処理を個別に記述する代わりに u3RestoreGroupFamily 関数の一番目の引数を NULL、二番目の引数を SvGrp として呼び出すと、SvGrp に記述されたバックアップファイルを削除し、SvGrp 自体も破却する。

ここで、バックアップと復元に関連するデータをリスト９－１に示す。

リスト９－１：バックアップと復元に関連するデータと関数

```
enroParam のメンバ
d3Group**OrgGrp
void *SvGrp

typedef struct{
  char file[I3_MAX_PATHNAME];
  char gname[I3_MAX_PATHNAME];
}u3SaveGroup;

void*u3SaveGroupFamily(d3Group*g);
指定されたグループをファイルに保存する。
グループ名とファイル名を格納した u3SaveGroup 構造体（メモリブロック）を構築し、
そのアドレスを void 型へのポインタとして返す。

void*u3RestoreGroupFamily(int load, void*sginfo)
sginfo をクリアする（ファイルは削除し sginfo 構造体も解放する）。
load が非零なら、sginfo に記録されたファイルを読み込み、グループを返す。
      零なら、NULL を返す。
```

この方法は、プラグイン DLL であるトンネル生成機能(tunnel.dll)、法面生成機能

(nori.dll)、園路生成機能(ParkRoad.dll)、及び地形編集機能(land.dll)に含まれる、標高面作成、地形切断機能、頂点移動機能、地形の細分化と最適化機能で用いている。

９－３．移動・回転・スケール（CEditMove）

内部データとしては、選択したグループの上方リンクに付随するリンク・マトリクスを変更する操作である。編集操作に先立って、対象となるリンク・マトリクスを、BackupInitLayoutMatrix()関数(dataope.c)によりバックアップする。「戻す」ボタン、またはキャンセル終了により、バックアップされたマトリクスを、SetInitLayoutGroupLink()関数により復原する。

但し、編集が行われた後に、選択の変更（上位グループ、下位グループへ）が行われた場合には、新たに選択されたリンクのマトリクスが上書きバックアップされるため、復原することはできない。

９－４．マテリアル

マテリアルの編集は、モデルの編集の１種であるが、以下のような特殊な条件がある。
①ユーザーの便宜のために、編集ダイアログを開いたまま、複数のオブジェクトに対して次々と編集できること。
②オブジェクトの表面光学特性の編集のために用意されている複数のダイアログがあり、相互に呼び出しができること。

a. オリジナルのマテリアル編集画面(４－４(21)参照、CEditMate)
　編集画面ではカラーのみを直接編集する。マテリアル編集(b)、テクスチャ編集(c)のダイアログを下請け的に起動する。BackupGroup, GetBackupGroup, BackupFace, GetBackupFace の各関数を用いる。

b. マテリアル選択画面(４－４(24)参照、CMatList)
　マテリアル一覧をリスト表示し、ここから選択が行われた時点で、マテリアル編集画面(a)の上部の色球にのみ反映する。OK で終了した場合には、メイン画面で選択された対象物に反映する。キャンセルで終了した場合には、メイン画面には影響しない。上記のマテリアル選択操作でマテリアル編集画面に表示するマテリアルが、起動当初とは異なる表示となっていた場合であっても、こちらは復原しない。

c. オリジナルのテクスチャ編集画面(４－４(22)参照、CEditText)
　上記 a から起動するが、独自のバックアップ・復原を行う(BackupGroup, GetBackupGroup, BackupFace, GetBackupFace の各関数を使用)。このダイアログを OK で終了した場合、a の画面に戻るが、ここでキャンセルによる終了が行われてもテクスチャ編集結果を復原しない。

d. グラフィックなマテリアル編集画面（４－４(25)参照、CEditMaterial）
　マテリアルがリストボックスから選択された時点で、ダイアログ上部の色球に反映

する。上記 a からも起動可能であることから、特別なバックアップ処理を行わない。

e. グラフィックなテクスチャ編集画面（４－４(26)参照、CEditTexture）

テクスチャがリストボックスから選択された時点で、ダイアログ上部の図形に反映する。独自のバックアップ・復原処理を行う（BackupGroup, GetBackupGroup, BackupFace, GetBackupFace の各関数を使用)。

f. 様々な表色系によるカラー編集画面（４－４(28)参照、CColorSet）

編集開始に先立って、バックアップを作成する。その際、選択モードがグループであれば、BackupGroup() 関数(dataope.c)によりグループを、また面であれば BackupFace() 関数(dataope.c)により面をバックアップする。キャンセルで終了となった場合には、GetBackupGroup()関数、GetBackupFace()関数により復原を行う。

９－５．光源の編集

光源は、各シーンに対して一つずつ定義される光源グループ（LG、メモリ・ブロック）に、最大８の光源ユニット（メモリ・ブロック）が所属する構成となっている。光源グループは、複数のシーンから共有することができ、また光源ユニットも複数の光源グループ（従ってシーン）から共有できる、という複雑で重層的な構成となっている。

光源の編集においては、ＯＫボタンによる終了と、キャンセルによる終了があり、前者の場合にはそのままの状態で、また後者の場合には、編集前の状態に復原した上で処理を終了している。

編集処理を行うに先立ってバックアップを作成する。処理結果に対して、OK で終了すれば、バックアップを解放する。一方キャンセルで終了すれば、バックアップを復元すると共に、処理結果を解放する。解放に際しては、他のシーンから参照されている光源ユニット、光源グループに関するチェックを行わないと、リンク切れを起こす。例えば、ある光源ユニットの編集に際して、メモリブロックの再構成（realloc）が行われた場合、たとえ新たに取得された光源ユニットが同一名称をもち、編集対象としているシーンに固有の光源グループにリンクされていたとしても、他のシーンが使用している別の光源グループからの、この光源ユニットへのリンクは失われている。

① バックアップ

光源自動計算の場合には、既存の光源グループ LG のバックアップを BU 関数により作成する(gydlg.cpp)。この関数は、LSS-S（シーン）編集モードにおいては、編集対象である仮のシーン TempScene のメンバである光源グループのポインタをコピーする。また、LSS-G（モデル）の編集モードにおいては、唯一の光源グループのコピーを作成する。

② OK 処理

OK で終了する場合には、BB 関数によりバックアップされたデータを処理する。LSS-G モードにおいては、バックアップした光源グループ LG を解放する。LSS-S も

ードにおいては、LG への他のシーンからの参照をチェックし、他からも参照されていれば、そのままとする。唯一の LG であれば、他から参照されていない光源 L のみを解放した上で、LG を解放する。

③ キャンセル処理

キャンセルで終了する場合には、UB 関数により光源グループを復原する。それまでの編集で新たに作成した LG とそれに帰属する L を解放する。

LSS-S（シーン）の編集モードにおいては、光源を編集した結果は、シャッター操作により初めて確定し、シーン配列の中の特定の要素の中に TempScene からコピー・保存され、更に編集終了後のファイル保存に反映される。

LSS-G（モデル）の編集モードにおいては、光源編集は他の編集操作を容易化することだけが目的であり、処理結果をファイルに保存することはない。従って、処理結果は、g3SetLightGroup(&LG)関数で直接表示状態に反映する。この関数は、光源グループのアドレス(&LG)は保持せず、内部でコピーを作成して表示に使用する。このコピーの処理の中で、それまでの LG は解放される。従って、不適切な LG 情報が渡されると、その時点ではなく次の更新の中での解放処理の中で障害を生じる可能性がある点は注意を要する。更に、g3GetLightGroup()関数は、g3 空間にある現在の LG のポインタを返す。このポインタを用いて、不適切な処理（例えば、メモリーブロックの解放）が行われると、やはり以後の更新処理の中でエラーを生じる。従って、光源に関するポインタの処理は特に慎重に行わなければならない。

このような仕組になっている理由は、上述の複雑で重層的な LG 情報の管理方法にある。表示の LG 情報を変更したい時には、作成した LG 情報を g3SetLightGroup(&LG)関数で表示に反映させるが、作成した側で LG 情報を削除することができる。また、シーンの切換に際しては、シーン・リストに保持してあるオリジナル情報を s3SetLightGroup 関数で G3 空間にコピーするだけでよい。仮に G3 空間のコピーが変更（例えば削除）されても、オリジナル情報は影響を受けない。

９－６．効果

効果のデータ構造は、光源の場合に似て重層的である。即ち、一つのシーンに割り当てられた一つの効果グループが、効果ユニットを複数（効果の場合には総数に上限はない）参照できる。また、ひとつの効果ユニットは、複数の効果グループから共用される場合がある。また、光源グループの場合、ひとつのシーンが必ず一つの光源グループを持ち、しかも光源グループには最低一つの光源ユニットが無ければならない、という規則が存在するが、効果の場合にはこのような制約はなく、効果グループを特に持たないシーンや、存在意義は別として、効果ユニットがない効果グループなどが存在しうる。

従って、ひとつのシーンに対する効果の編集のバックアップと、キャンセルで終了した

場合のリカバリーは、全てのシーンに対して行っている。即ち、シーンの数と同数の効果グループの配列をバックアップ用に用意しておき、効果の編集開始時に各シーンの効果グループと効果ユニットのコピーをこれに割り当てる。効果ユニットは、同名で別のメモリブロックが発生すると、上記のデータ構造の一貫性が失われるため、リストとして管理し、既に同名の効果ユニットのコピーが作成されている場合には、新たに作成せず、これを参照する。

正常終了（OK）の場合には、このバックアップを全て解放し終了する。一方、キャンセル終了した場合には、現在の全てのシーンに登録されている効果グループと効果ユニットを全て解放したうえで、バックアップの効果グループと効果のポインタを各シーンにコピーする。

いずれの場合にも、効果グループと効果ユニットの解放に際しては、解放済みの効果ユニットを再度解放しようとして、アクセス違反を犯すことが無いように、「ごみ箱」に相当する効果ユニットの辞書（配列）をまず用意して、この上に解放すべき効果ユニットへのポインタの一覧を作成した上で、最後に各効果ユニットを一度だけ解放している。

９－７．処理方法の使い分け

バックアップ・アンドゥの処理方法には、以上に示したように、いくつかの異なるアルゴリズムがあり、現在のバージョンの中では、必要性や状況により使い分けている。これを大きく分けると、以下のように整理することができる。

①編集対象による違い

編集対象が地形や地物全体などの大きなモデルである場合には、バックアップをメモリ上に構築することはメモリ制約を受けるおそれがあるため、処理時間をある程度犠牲にして、処理前のモデルをファイルに保存しておき、もし処理結果がキャンセルされた場合には、処理結果を廃棄し、ファイルからモデルを再構築している。

編集対象が光源その他環境設定に関するものである場合、保存方法を統一化することは難しい。

②編集方法による違い

たとえば、簡単なダイアログで比較的単純な処理を行う場合には OK ボタンとキャンセル・ボタンを用意し、OK ボタンが押された場合のみ処理を行う。一度行われた処理の取り消しを行うことはできない。キャンセル・ボタンでは、処理を行うことなく終了する。

地形に係る編集などのように、実行ボタン、取り消しボタン、終了ボタンの３種類を有するダイアログでは、実行によりデータが変更される。キャンセル・ボタンで元のデータに復元する。終了ボタンで、処理実行の有無や結果の成否にかかわらず、その状態でダイアログを終了する。

カラー・マテリアルや光源の編集のような場合には、OK とキャンセルのボタン操作ではなく、ダイアログ上のリストからの選択やスライドバー操作により編集を行う。このため、

終了を OK ボタンで行うか、キャンセル・ボタンで行うかで、処理結果の採否を確定している。更に、作業能率のために、ダイアログを開いたまま、メイン画面で選択対象を変えて、それぞれのオブジェクトのカラーやマテリアル設定を「次々と」編集することを可能にしている。このような場合には、キャンセル処理で一つずつ復元することは複雑なデータ構造を必要とするため、キャンセル処理においては、ダイアログが開いた時点に復原している。

１０．ビューワ、ネットワーク機能

１０－１．概要

景観シミュレータ基幹部分 sim.exe を直接起動する場合、コンテンツが無い状態で初期表示を行う。この状態から、メニューのファイルを開くコマンドでファイルを選択して、既存のファイルを開き、表示することができる。

これに対して、ウェブ･ブラウザと連携して、ウェブ上のコンテンツとして提供されているファイルを表示する場合や、景観データベースで選択されたオブジェクトを表示するために、予め表示するファイルを引数として渡す方法で sim.exe を起動する場合には、画面が開いた時点で、指定された地物を直ちに表示する。

１０－２．ドラッグ・アンド・ドロップによる起動

最も基本的な処理として、デスクトップに表示されている sim.exe のアイコンの上に、LSS-G ファイルまたは LSS-S ファイルがドラッグ･アンド･ドロップされた場合に、sim.exe が起動して、これらを表示する。

拡張子.geo, .scn のファイルを開くアプリケーションをエクスプローラの設定で sim.exe に設定してある場合には、これらのファイルをダブルクリックした場合にも同様の表示を行うことができる。

WEB ブラウザから sim.exe がビューワとして起動される場合も、同様の機構である。

これらは、コマンドラインで、リスト１０－１に示したコマンドを入力した場合と同様の起動条件である。

リスト１０－１：WEB ブラウザから起動する場合と等価のコマンド

```
sim.exe [ファイル名]
例：sim.exe sample.geo
```

これらに対応する処理は、CSimApp::InitInstance()関数(sim.cpp)の中で処理している。この関数においては、引数の文字列を検査し、ファイル名が直接指定されている場合には、アプリケーション・ライブラリ関数である SetCalledDBFlag()関数で、表示すべきファイルが指定されていることを示すフラグ「２」を立てた上で、SetLangsung()関数で、そのファイル名を指定する。

引き続き、初期化の過程で、CDrawFrm::OnCreate()関数（メイン画面の初期化を行う。drawfrm.cpp）が、CmainFrame::GetCalledDB()関数を呼び出し、その中でこれらのフラグが検査され、ファイルが指定されている場合にはこれらのロードを行い、初期表示を行う。

なお、引数の無い起動においては、初期化の間、景観シミュレータのロゴを CSplashWnd クラス(Splash.cpp)で表示するが、引数で表示すべきファイルが指定されていて、これが LSS-G 形式（拡張子.geo）の場合には、ロゴの表示を抑制する。

１０－３．景観データベースからの起動

景観データベースで三次元データを確認表示する場合には、景観データベースの側で、CreateProcess 関数により景観シミュレータを起動する。この機構は、１０－２よりも古く、初期のバージョンから実装されている。景観データベースから起動する場合には、ファイル名直接ではなく、[f –db]という引数が sim.exe に渡される。

起動された sim.exe の側では、CSimApp::InitInstance()関数の中で、SetCalledDBFlag 関数を用い、フラグ「１」を立てる。

次に、１０－２と同様 DrawFrm::OnCreate から起動される CMainFrm::GetCalledDB( ) 関数の中で、ファイルのロードを行う。景観データベースからの起動の場合、起動元の景観データベース側で、ファイル名を tmp002.txt というファイルの中で指定する。

このファイルの中には、１行のコマンドの中に、「選択タイプ　ファイルタイプ　ファイル名」の３項目の情報が含まれている。

このファイルは、アプリケーション・ライブラリ関数 ReadFromFile 関数(common.c)で読み出される。読み出された後に、直ちにファイルを読み込んで初期表示を行う。景観データベースからの起動の場合には、ファイルの種類に関わらず、起動時のロゴは表示しない。

なお、上記の一時的ファイルには３種類あり、以下のように使い分けられている。

リスト１０－２：ファイル名伝達のためのテンポラリファイルの用途

tmp001.txt 景観シミュレータの配置ダイアログの操作中、配置オブジェクトを選択するために起動された景観データベース検索機能が、最終的に選択されたオブジェクトを返す。
tmp002.txt 景観データベースで検索した三次元オブジェクトを表示して内容を確認するために起動する景観シミュレータに対して、オブジェクトを伝える。
tmp003.txt 景観シミュレータが終了する際に表示していたファイルを保存する。

１０－４．WEB ブラウザとの連携動作

WEB ブラウザは、リンクを定義するタグ

表示文字列

が定義されている表示文字列がクリックされた時に、「href=」で指定されたファイルをダウンロードし、表示しようとする。その際に、このファイルに関連づけられたアプリケーションが選択される。例えば、「.pdf」が定義されていた場合には、PDF ファイルのビューワを起動し、指定されたファイルを表示する。同様の機構を用いて、sim.exe を用いて、LSS-S ファイルを表示する。

景観シミュレータで表示を行うファイルの場合、最初ロードされるファイルから参照される、様々の部品データや画像のファイルを重層的にロードする方法が一般的である。一方、WEB ブラウザから引数として sim.exe に渡されるファイル名は、通常 WEB 上の URL そのものではなく、WEB ブラウザが既にキャッシュにダウンロードしたローカルなファイ

ル名である。このため、引き続きそこから参照されたファイルにアクセスし、ダウンロードを行うためには、全ての参照をフルパスで記述するか、あるいはサーバーの URL を知っておく必要がある。

このため、WEB サーバー上に置き、リンクを張る根本のファイルとしては LSS-S 形式を用いている。そして、この LSS-S ファイルの中で MODEL コマンドにより定義する LSS-G 形式のファイルを、フルパス(URL)で記述する方法を採っている。

景観シミュレータの側では、最初に引数として渡された LSS-S ファイル（拡張子.scn）を読み込む際に、MODEL コマンドで指定された LSS-G ファイル（拡張子.geo）の所在（サーバーの URL とその下のサブディレクトリ名）を、作業用ディレクトリとして記憶しておく。引き続きそこから参照される部品等のアドレスの前にこの URL をプレフィックスとして加えた、フルパスの URL アドレスを用いて、ダウンロードを行う。LSS-S ファイルを初期ファイルとして指定された条件で起動する場合、ダウンロード中、ロゴの表示を行う。

このロード処理は、以下の系列で行われる。

リスト１０－３：WEB からの LSS データ取得処理

a. CMainFrame::OpenScene(char *file)　(mainfrm.cpp、ダイアログ・ハンドラ)
b. LoadSceneFile(file, width, height)　(common.c、アプリケーション・ライブラリ)
c. dbLoadScene(name,scn)　(dbms.c、DBMS ライブラリ)
要求されたシーンファイルを開き、1 行ずつ読みだして、IP ライブラリの IP_interpret() 関数による解析を実効する。
d. fopen_(filename, “r”)　(fopen_.c、IP ライブラリ)
要求されたファイルがローカルであればこれを開く。ファイル名が http:または ftp:から始まるものである場合には、e.以下を実行し、その結果のファイルを開く。
e. download_file_cashe(urlname)　(fopen_.c、IP ライブラリ)
ファイルに応じてローカルなファイル名を生成し、有無を調べる。存在すればそのファイル名を返す。存在しなければ f.でダウンロードした上で、ローカルなファイル名を返す。
f. geoload.exe の実行によるダウンロード　(bin ディレクトリにセットアップされている)

既にダウンロードしてある LSS-G ファイルは、特別なキャッシュは設けず、通常のディレクトリに置く。このため、上記のダウンロードに際しては、通常のディレクトリに同名のファイルが存在するかどうか検査し、これを使用する。もし存在しなければ、ダウンロードした上でこれを利用する。このことにより、二回目以降のダウンロード時間を省略している。

また、LSS-G ファイルのロード中で、FILE コマンドにより別の LSS-G ファイルを参照する場合には、このローカルなファイル検索に先立って、メモリ上に既にロードされたファイルを検索し、ロード済みであれば再ロードはしない（通常の処理と同様）。

しかしながら、サーバー側で LSS-G ファイルの内容が修正された場合に、ダウンロード済みのファイルが更新されないという問題は残されている（ダウンロードに際して、ネッ

トワークアクセスを必要とするタイムスタンプの比較を行っていない)。

景観データベースで用意してある部品であって、ユーザー側のシステムにセットアップされているものが別ファイルのロード中に参照された場合についても、これが既にローカルに存在していれば、ダウンロードを省略している。即ち、上記のダウンロードに先立つ検索に際して、通常のディレクトリのみならず、景観データベースの LSS-G ファイルを格納するディレクトリにおいても、目的とするファイルを探索し、既に存在する場合にはこれを利用している。

また、外部関数が参照された場合であって、その外部関数がまだインストールされていない場合にも、サーバーから関数をダウンロードした上で形状生成処理を行う。

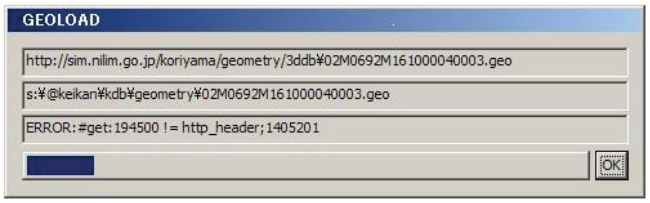


図１０－１　geoload.exe　によるダウンロード過程の表示

これらの各種ファイルを取得する geoload.exe は、サーバーから大きな三次元データをダウンロードする場合に、長い時間を要する場合がある。また、http プロトコルでは、Read 関数が正常終了しても、不完全なファイルが生成されることがある。この不完全なファイルが解析時に文法違反を生じなかった場合には、エラーメッセージなしに地物の一部が欠落した表示が行われる恐れがあった。そこで、ver.2.09 においては以下の改良を行った。

１．一つのファイルのダウンロード中にダイアログを開き、以下の情報を表示する。
(1)ＷＥＢサーバーから取得するリモートファイルのＵＲＬ
(2)ローカルに保存するファイル名(フルパス表示)
(3)処理過程、及びダウンロード終了バイト数
(4)プログレスインジケータ
２．ダウンロードが正常に終了した場合には、直ちにダイアログを閉じる。
３．ダウンロードに失敗した場合には、上記(3)の欄にエラーメッセージを表示して停止し、OK を待つ。
(1)要求したファイルがサーバーに存在しない場合(NotFound)
(2)ファイルの代わりに、サーバーが、"<"で始まる情報を返した場合（エラーメッセージと解釈）
(3)HTTP ヘッダーのコードが 200(OK)以外であった場合
(4)HTTP ヘッダーで示された総バイト数と異なる長さのファイルを取得した場合

エラーで終了した場合、途中まで作成したファイルを削除する。従って、このダイアログ終了後、geoload を起動した sim.exe の側から 「ファイルを開くのに失敗しました」というエラーメッセージが出た後、次のファイル取得処理を続行する。 最後に、ダウンロードできたファイルだけに基づいた表示を行う。

途中でエラーが発生した場合には、sim.exe を閉じた後、再度 WEB ブラウザ上で同じリンクをもう一度クリックする。 すると、再び sim.exe が起動しダウンロードを開始する。この時、既にダウンロードに成功したファイルは、 ローカルに存在しているのでこれを利用しダウンロードをスキップする。前回失敗したファイルのみ、 サーバーからダウンロードを行う。

１１．多言語処理

１１－１．概要

多言語処理は、同一の実行形式において、言語に依存するメニュー、ボタン等のダイアログ表示、エラー・メッセージ、ヘルプを、全て外部のテキストファイルから動的にロードすることにより、言語の切り替えを行う。

このことにより、デバッグや機能改善が行われた場合においても、再度プログラマによるビルド作業を行うことなく、その結果を各国語で利用できるようになる。

また、新たな言語に翻訳して利用しようとする場合においても、プログラム作業を行うことなく、翻訳家によりテキストファイルを用意するだけで対応が可能となった。

シミュレーションの機能を、安定性信頼性を追求した基幹部分と、機能拡張のための外部関数及びプラグイン DLL に分離し、新たなニーズへの対応を基幹部分と分離するとともに、多言語機能により言語毎にソースコードのバージョンが枝分かれした問題を解決するために、基幹部分を共通のソースコードに再統一し、完成度を高めることが可能となった。

多言語処理機能には、sim.exe を構成する各ダイアログのみならず、そこから起動される外部関数やプラグイン DLL をも協調的に動作させる仕組みも提供しており、更に外部関数やプラグイン DLL のコーディングを助けるための共用ライブラリも用意した。

システムが現在使用している言語は、表１１－１に示した ISO 639 のコード表により識別している。これを用いて、sim.exe 内部の各種ダイアログにおける表示言語の選択、セットアップにおける各種ファイルの配置、外部関数やプラグイン DLL における表示言語の協調などの制御を行っている。

表１１－１：ISO 639 の代表的な言語コード

言語	言語コード
アブハジア語	ab
アファル語	aa
アフリカーンス語	af
アルバニア語	sq
アムハラ語	am
アラビア語	ar
アルメニア語	hy
アッサム語	as
アイマラ語	ay
アゼルバイジェン語	az
バシキール語	ba
バスク語	eu
ベンガル語	bn
ブータン語	dz
ビハール語	bh
ビスラマ語	bi
ブルターニュ語	br
ブルガリア語	bg
ビルマ語	my
白ロシア語（ベラルーシ語）	be
カンボジア語	km
カタラン語	ca
中国語（簡体）	zh
中国語（繁体）	zh
コルシカ語	co
クロアチア語	hr
チェコ語	cs
デンマーク語	da
オランダ語	nl
英語	en
エスペラント語	eo
エストニア語	et
フェロー語	fo
ペルシャ語	fa
フィジー語	fj
フィンランド語	fi
フランス語	fr
フリジア語	fy
ガリシア語	gl
ゲーリック語（スコットランド語）	gd
ゲーリック語（マン島語）	gv
ジョージア語	ka
ドイツ語	de
ギリシャ語	el
グリーンランド語	kl

グアラニー語	gn
グジャラト語	gu
ハウサ語	ha
ヘブライ語	he
ヒンディー語	hi
ハンガリー語	hu
アイスランド語	is
インドネシア語	id
インターリングア（国際語）	ia
インターリング語	ie
イヌクティトット語	iu
イヌピア語	ik
アイルランド語	ga
イタリア語	it
日本語	ja
ジャワ語	jv
カンナダ語	kn
カシミール語	ks
カザフ語	kk
キヤーワンダ語（ルアンダ語）	rw
キルギス語	ky
キルンディ語（ルンディ語）	rn
韓国語	ko
クルド語	ku
ラオタ語	lo
ラテン語	la
ラトビア語（レット語）	lv
リンブルク語	li
リンガラ語	ln
リトアニア語	lt
マカドニア語	mk
マダガスカル語	mg
マレー語	ms
マラヤーラム語	ml
マルタ語	mt
マオリ語	mi
マラッタ語	mr
モルダビア語	mo
モンゴル語	mn
ナウル語	na
ネパール語	ne
ノルウェー語	no
オキタン語	oc
オーリア語	or
オロモ語（ガラ語）	om
パシト語（パシュトー語）	ps
ポーランド語	pl
ポルトガル語	pt

パンジャビ語	pa
クエチュア語	qu
レートロマンス語	rm
ルーマニア語	ro
ロシア語	ru
サモア語	sm
サングホ語	sg
サンスクリット語	sa
セルビア語	sr
セルボクロアチア語	sh
セト語	st
セツワナ語	tn
ショナ語	sn
シンド語	sd
シンハラ語	si
シスワティ語	ss
スロヴァキア語	sk
スロヴェニア語	sl
ソマリ語	so
スペイン語	es
スーダン語	su
スワヒリ語（キスワヒリ語）	sw
スウェーデン語	sv
タガログ語	tl
タジク語	tg
タミル語	ta
タタール語	tt
テルグ語	te
タイ語	th
チベット語	bo
チグリニャ語	ti
トンガ語	to
ヅォンガ語	ts
トルコ語	tr
トルクメン語	tk
トウィ語	tw
ウイグル語	ug
ウクライナ語	uk
ウルドゥー語	ur
ウズベク語	uz
ベトナム語	vi
ヴォラピュック語	vo
ウェールズ語	cy
ウォロフ語	wo
コーサ語	xh
イディッシュ語	yi
ヨルバ語	yo
ズールー語	zu

１１－２．言語依存部分の外部テキスト化

（1）ディレクトリ構成

複数言語で切換えながら操作できる環境が実現できるように、セットアップにおいて多言語対応のディレクトリを新たに設けることとした。このディレクトリの位置は自由に設定することができ、環境設定ファイル kdbms.set の LANG_PATH エントリによって定義する。デフォルトでは、ホームディレトリ／ksim／Language である。

各言語に対応した、テキストデータは、この下に、表１１－１で示したコード名のサブ

ディレクトリを設け、そこに格納する。

[例]日本語関連：c:¥@keikan¥ksim¥Language¥ja

（2）リソース

メニュー項目や、操作ボタンの表記等は、従来のシステムにおいては、リソース中に記述され、実行形式の中に固定的に埋め込まれていたが、多言語版においては、テキスト形式外部ファイルとして動的にロードすることとした。

各メニュー項目や、ダイアログのコントロールに表示する、各言語による表記（文字列）は、xml ファイルによって定義している。この xml ファイルは、ksim/bin ディレクトリに格納されたリソースファイルおよびヘッダファイルから、sim.exe が自動的に生成したものを出発点として、翻訳家によるテキスト・ベースの作業を行うことにより完成する。

（3）プログラム中使用する固定的文字列

従来プログラム中に埋め込まれる文字列は、直接プログラム中に記述するのではなく、ソースコードの kj_sjis.h に集約し、他言語に翻訳移植する場合には、このヘッダファイルの翻訳を行った上で、ビルドに使用していた。これは翻訳移植作業を容易化してはいたが、最終的に言語に依存した固定的な文字列は実行形式(sim.exe)の中に埋め込まれるため、修正や他言語への移植に際しては、再ビルドを必要としていた。

各国語専用のプログラム中の文字列はソースコードの kj_sjis.h で定義していたが、これを多言語版においては、LangConv.[言語コード].txt に移し、各言語のディレクトリに置いている。プログラム中で使用される文字列を、システム起動時および言語切換時に動的にロードし、変換テーブル g_TextConvMap により管理し、各種処理に使用する。

ソースコード中においてこの文字列を使用する箇所では、従来の KANJI_XX 等のシンボルは、LangConvert(“KANJI_XX”)という関数に置き換えた。この LangConvert 関数は、上記変換テーブルを用いて、必要な言語による文字列への変換を行う。この変換テーブルは、CMultiLang::InitInfo()関数の中で初期化し構築する。InitInfo 関数は、起動時、及び言語切換時に呼び出される。

（4）ヘルプ・ファイル

ヘルプ・ファイルは、Language/[言語コード]/[ヘルプ名].[言語コード].txt として、用意している。異なる言語のヘルプは、異なるディレクトリに置き、ヘルプが要求された時点で、使用言語に対応したディレクトリ中のヘルプ・ファイルを表示する。

（5）エラー・メッセージ

従来は、ksim/bin/ERR_MSG.txt に格納し、システムの初期化段階でロードしエラーの表示を行っていた。これも、言語毎のディレクトリに格納し、使用言語により選択的にロードすることとした。

（6）その他のテキストファイル

この他に、テクスチャの自動貼り付けのファイルをリストする kdb/texture/sgi/autotex.set、およびプラグイン dll の一覧表である ksim/bin/plugin.tab に

関しても、各言語のディレクトリに翻訳結果が存在すれば、そちらを表示に使用する。いずれも翻訳結果のファイルは、元のディレクトリではなく、Language.[言語コード]のディレクトリに置く。また、ファイル名称も、autotex.[言語コード].set、plugin.[言語コード].set等とし、誤って異なる言語のファイルに上書きしないように配慮している。

なお、この他に、ext.tab (外部ファイル一覧)、roadsec.set、riversec.set（それぞれ道路断面、河川断面を記述したファイル名を格納）等があるが、ファイル名を直接記述し、表示するだけなので、通常は多言語対応する必要はない。もし登録ファイル名に漢字名称等が用いられていて、他国語で表示不能であれば、断面ファイル等の名称を変更した上で、これらのファイルにその名称を直接登録し直す必要がある。

１１－３．表示に使用するフォント

ダイアログの各コントロールに使用されるフォントは、xml ファイルの中で指定することができる。

各ダイアログの定義の中に、例えば

<FONT size=”12” original=”System” replace=”MS UI Gothic” />

という記述で、サイズとフォント名称を指定することができる。

サイズは、xml ファイルを自動生成する際に、リソースファイルで指定されたサイズが初期値として設定されている。

翻訳移植にあたっては、使用する言語が表示できるフォントを指定しなければならない。ダイアログ単位で、サイズと、xml ファイルの中で置換後のフォントを書き換えることにより、特記指定することができるので、ダイアログに使用したリソースの中に表示文字列が納まり切らない場合に、フォントを小さくするという対応も可能である。

実際のフォント生成にあたっては、各ダイアログの初期化の中で、

CMultLang::CMultiLangDialogConv(変換後言語名称, 変換対象ダイアログ)

が起動され、この中でまず、指定されたサイズの論理フォントを作成し、各コントロール（ボタン、固定的テキスト表示、チェックボックス、ラジオボタン等々）の表示書き換えが実行される際に、このフォントを使用する。

フォントの生成には、CFont::CreatePointFont(size, name) 関数を用いて、サイズとフォントの種類だけを指定する方法を採っている。ここで生成したフォントが、不要となるまでの間、固定的なアドレスに保持されていなければ、再描画の際にシステム・フォントが使用されてしまう。そこで、CMultilang::FontManager 関数により、数多くのダイアログに使用される論理フォントを、サイズと名称でソートした上で、スタティックな配列に割り付ける形で管理している。この論理フォントのデータは、言語の切り替えに際して再構築され、システム終了時に除却される。各コントロールへのフォントと表示文字の設定を行う場所は、ダイアログにより異なるが、各クラスの構築の中で行っている場合、OnInitDialog 関数、OnCreate 関数の中で実行している場合などがある。

フォントを管理するためのメモリ・ブロックを解放するためには、各ダイアログの終了時点で、ハンドラのクラスのデストラクタ関数等において、

FontManager(0,"");

を実行する。FontManager 関数は、この引数を見て、メモリを解放する。

１１－４．ダイアログのレイアウト

使用する言語によって、ダイアログを構成するボタンやエディットボックスのそれぞれについて納まりの良いサイズやレイアウトは変わる。そこで、現在は、リソースの中に、日中韓台などの２バイト系文字によるレイアウト（横幅小に対応した IDD_DIALOGXXX と、アルファベット等の１バイト系文字によるレイアウト（横幅大）に対応した IDD_DIALOGXXX_0 の２種類を用意し、指定した言語が１バイト系か２バイト系かで、使用するリソースを使い分けている。

各リソースの中の、言語に依存する文字列に関しては、１バイト系しか表示することのできない OS 環境における環境設定（適切にインストールされていない場合の表示など）に資するため、全て１バイト系のコードが用いられている。実行形式が起動してから、必要とする言語の文字列に動的に置き換えられる。

ダイアログテンプレートを選択するためには、各ダイアログのハンドラ・クラスのコンストラクタ関数の中で、

m_lpszTemplateName = MAKEINTRESOURCE("ダイアログ ID");

を実行すれば良い。

別の方法として、ダイアログを別のクラスからモードレス・ダイアログとして開く場合には、Create 関数の引数として、明示的にダイアログテンプレートを引数として指定する方法がある。

モーダル・ダイアログの場合には、DoModal で起動する際には、Create 関数が自動的には呼び出されないため、前者の方法が有効である。

１１－５．言語の切換操作

以上により、一つのインストール中に複数言語を共存させ、必要な言語に切り替えることが可能となった。

言語の切り替えは、新たに追加したメニューの[ファイル][言語選択]により起動するダイアログで行う。このメニューに一覧表示される言語は、Language ディレクトリにある言語毎のサブディレクトリ名である。

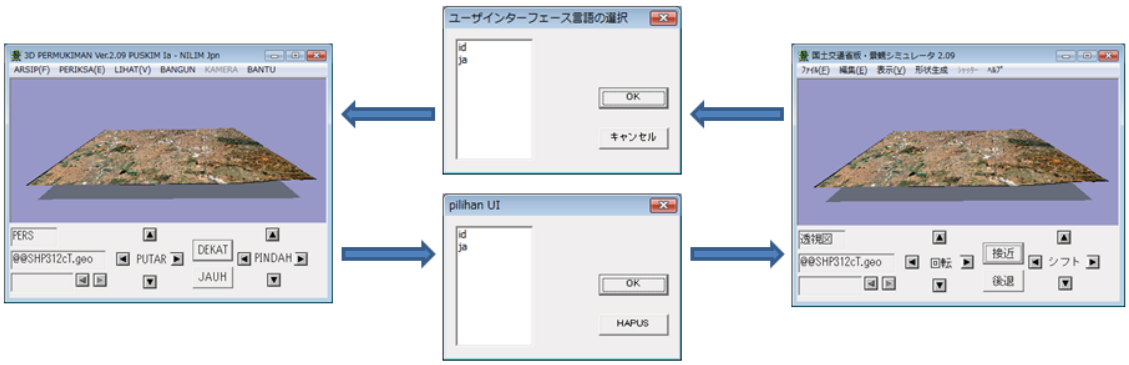


図１１－１：言語切換ダイアログ

１１－６．動作の詳細と、処理プログラム

新たに追加されたソースコードは、language.cpp の中に集約されているが、これに付随して、UNICODE 変換機能や、XML ファイル解析機能を担う補助的な以下のソースコードも追加されている。

表１１－２：多言語機能のための関数を含むソースコード

１．主要部分 multilang.cpp, multilang.h ２．XML ファイル解析機能（オープン・ソースを利用） tinystr.cpp, tinystr.h tinyxml.cpp, tinyxml.h tinyxmlerror.cpp tinyxmlparser.cpp ３．SJIS, utf-8 及び utf-16 の間の２バイト文字コード変換 UnicodeF.cpp, UnicodeF.h ４．外部関数およびプラグイン DLL における多言語対応機能 LocalLang.cpp, LocalLang.h これは、LocalLang.lib として、外部関数、プラグイン DLL から参照する。

さらに、各ダイアログをコントロールする処理の中にも、言語切り替えの処理が追加されている。多言語化のために追加された部分に関しては、表１１－３のようにプリプロセッサで囲ってあるため、多言語版のビルド環境においては、プリプロセッサ MULTI_LANG を定義しておく必要がある。

表１１－３：多言語化のために追加した部分の表記

```
#ifdef MULTI_LANG
  (追加されたソースコード)
#endif
```

（1）初期化動作

言語が設定された時点で、実行形式と同じksim¥binディレクトリにlang.ctlファイル が作成される。これには言語の種類と、言語関連のテキストファイルを格納する Languageディレクトリのパスが記録されている。起動した時点で、lang.ctl が既に存在していれば、初期表示言語は、lang.ctl ファイルに記録されている言語に設定される。動作中に言語が変更された場合には、選択された言語をもってlang.ctl ファイルを上書きする。

Sim.exe が起動した時点で、CSimApp::CSimApp() から、CMultiLang::CMultiLang()が呼び出され、この中で、Lang.ctl に記憶された前回使用言語に基づく初期化動作を行う。Lang.ctl ファイルが存在しなかった場合には、OS が日本語を表示可能である場合には、日本語、そうでない場合には、インドネシア語（アルファベット）による初期表示を行う。初期化時点での言語は、m_Lang に２バイトの文字列として格納され、以後の処理に用いられる。言語が変更された時点で、使用されていた言語がLang.ctl に記録される。

lang.ctl ファイルに記録する、言語を示すコードは、表１１－１に示した通り、ISO 639 に従い、id（インドネシア語）, ja（日本語） などのアルファベット２文字である。

多言語版においては、ksim¥bin ディレクトリの中に、従来からの実行形式本体であるsim.exe に加えて、開発環境からコピーした、リソースの文字情報を含む sim.rc およびresource.h（ビルドの中に含まれるリソース・ファイル）の二つのファイルを追加で置く。起動した sim.exe は、設定された言語を取得した後に、sim.rc と、これを展開し言語依存部分のテキストを格納した、SimRc.[言語コード].xml というｘｍｌ形式で訳語を格納したファイルの有無の確認と、タイムスタンプを比較する（図１１－２）。

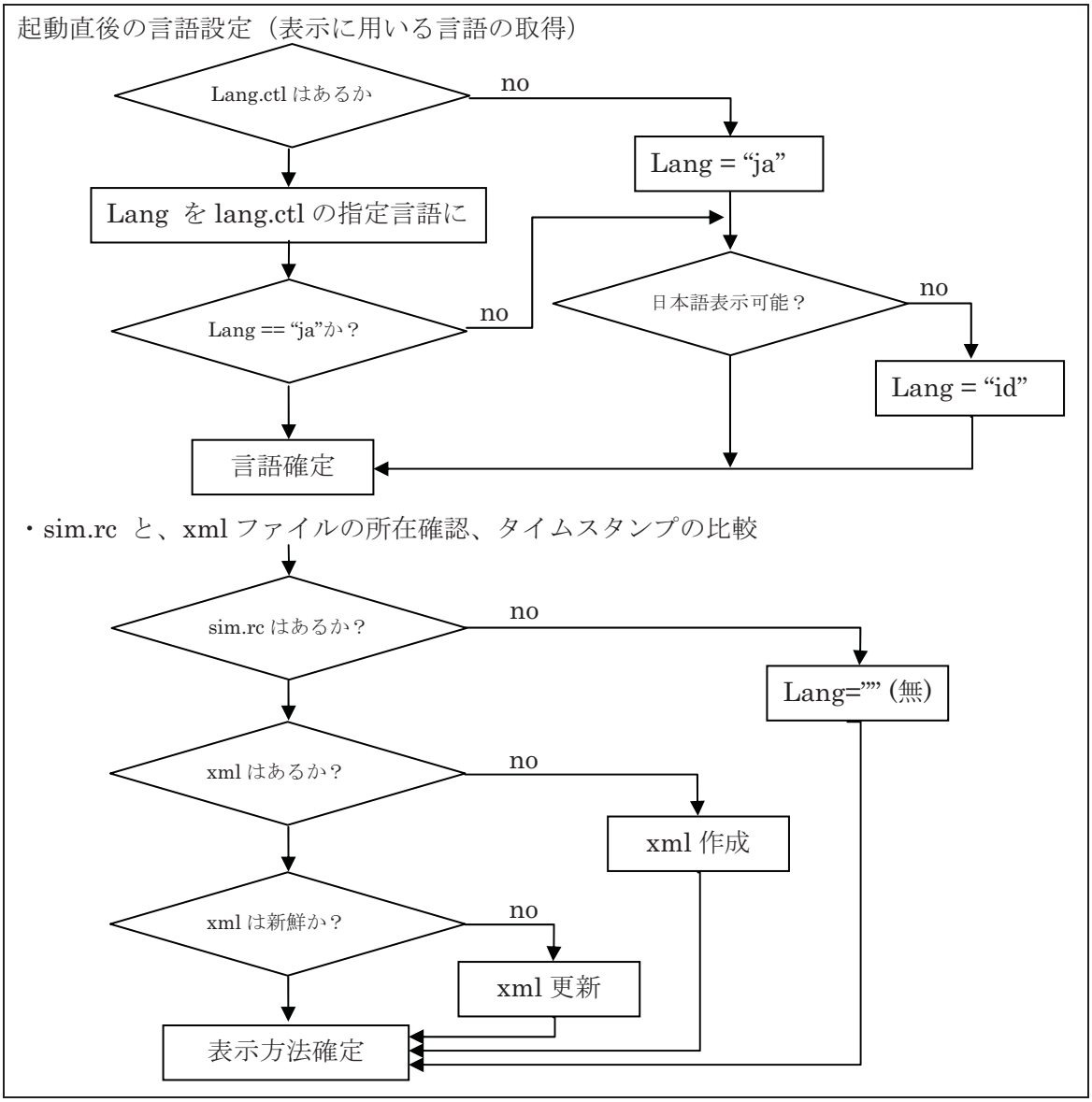


図１１－２：sim.exe の、多言語に対応した初期化フロー図

もし xml ファイルが存在し、かつ sim.rc よりも新しければ、これをそのまま使用して、各ダイアログのための表記文字をロードする。また、もし xml ファイルが存在しなければ、新たに訳語が未記入の新しい xml ファイルを作成し、これを表示に用いる。

一方、既存の xml のタイムスタンプの方が、sim.rc よりも古ければ、この xml が作成された後に sim.exe のバージョンの変更（デバッグ、機能追加等）が行われ、xml ファイルは既に陳腐化していると判定し、新たな xml ファイルを作成する。その際に、旧版の xml ファイル中の訳語が入った項目を保ちながら、新たに追加されたメニュー項目があれば、対応するエントリーだけを作成し、訳語は空欄として、翻訳者による補完に備える。

実行段階では、ロードした xml ファイルの中に、表示に必要な訳語が存在しない場合（空白文字列：訳語未記入）には、リソースファイルの ID コード（“IDD_DLG_XXX” 等)に対

応する、xml で「id=”文字列”」で定義された文字列をアルファベットで表示に用いることで、翻訳が必要な項目が存在することをユーザーに知らしめる。

xml ファイルが使用できない場合（Lang=””（言語無指定））には、ビルド時にリソースで定義された表示文字(もし xml ファイルがあれば、original=”文字列”で定義された文字列に対応する)で表示を行う。

（２）各ダイアログの初期化

各ダイアログのタイトル、メニュー、ボタン等を表記する言語は、各ダイアログの初期化段階で設定される。比較的単純な経年変化ダイアログの場合で見ると、

CDispkeinen::CDispkeinen(CWnd: pParent) 関数の中で、使用する言語が２バイト系か１バイト系かを判定した上で、

Create(IDD_DLG_KEINEN_HENKA, NULL) あるいは

Create(IDD_DLG_KEINEN_HENKA_0, NULL) を実行し、さらに

theApp.m_pLang->MultiLangDialogConv(“IDD_DLG_KEINEN_HENKA”,this)又は

theApp.m_pLang->MultiLangDialogConv(“IDD_DLG_KEINEN_HENKA_0”,this)を実行する。後者の処理の中で、現在選択されている言語による表記が各要素に対して設定される。

DoModal 型のダイアログの場合、Create 関数が呼び出されずにダイアログが開く。また、予め Create を行っていると、DoModal の時点でエラーを生じる。そこで、構築の中で、使用するダイアログテンプレートを示す CDialog のメンバ変数を書き換えておく。

m_lpszTemplateName = MAKEINTRESOURCE(IDD_DLGXXX_0)

この変数は、通常は、ヘッダーの中で、ダイアログのメンバ変数 IDD = IDD_DLGXXX（漢字系）として初期化されている。この方法は、Create 関数を用いる場合にも適用することができる。

メイン画面からユーザーにより起動される各ダイアログが既に開いている状態において言語の切り替えはできない。

唯一、メイン画面のダイアログに関しては、言語の切り替えのみならず、言語が切り替えられた時点で、動的にダイアログを作り直す処理を行っている。

（３）言語切り替え動作

言語切り替えは、メインの[ファイル][言語の選択]により、選択可能な言語の一覧を表示し、そこからユーザーが選択することにより実行する。

CDlgSelLang という新たに追加されたクラスが処理を行う。この中で選択可能な言語がリストボックスで表示される。言語のリストは、Language ディレクトリの中にある、サブディレクトリ名の一覧となっている。この中から言語が選択され、OK により、言語切り替え処理に進む。

切換処理の中では、メイン画面以外のすべてのダイアログを閉じる。メモリ上にロードされているテキスト関連のデータ（ダイアログ、固定的文字列その他）を解放し、新たに

選択された言語のデータに入れ替える。次にこれを用いて、メイン画面のメニューと、下部の操作ボタン等の表示を、新たな言語に切り替える。この時、ロードされていたデータはそのまま継承され、グラフィックな表示は変わらない。

各編集等のダイアログは、次に構築・表示される際に、新たに設定された言語で表示される。

（4）ヘルプ処理

ヘルプは、テキストを notepad.exe または textviewer.exe によって表示するという方法を採用した。考え方として、ユーザーが発見したことをメモとして追加できることを目指した。表示に用いられるテキスト・エディタまたはテキスト・ビューワに、引数としてテキストファイル名を添えて、プロセスを起動することにより実現している。

ファイル名称は、概ね、各種ダイアログのプログラム上の名称に対応した名前に、「.txt」を拡張子として有している。従来のヘルプ・ファイルは、環境設定ファイルの、HELP_PATH エントリで定義されたディレクトリ（デフォルト設定は、ksim/help）に置かれている。多言語対応版においては、Language ディレクトリの下の言語別のディレクトリに置かれる。

更に、多言語化したために、ヘルプ・ファイルは言語毎に別のファイルとして作成される。言語を識別するために、「[固有名称].[言語コード].txt」という名称としている。日本語が選択されている場合に、言語別のディレクトリに、多言語版のヘルプ・ファイルが存在していない場合には、従来のヘルプ・ファイルを表示する。

引数として渡されるヘルプ・ファイルを、使用言語に対応して切り替えるために、Ver.2.07 までは、Pembantu()関数を使用して、CreateProcess 関数で notepad.exe を起動していた。多言語化を実現するために、これに代わって新たに PembantuX()関数を用意した。この関数は、現在選択されている言語を用いて、ヘルプ・ファイルが存在するディレクトリと、ヘルプ・ファイルの名称を作成した上で、コンテンツの表示を行う。ヘルプ・ファイルが存在しない場合には、新たに作成する。

最後に探しに行ったディレクトリにファイルが存在しなかった場合には、Pembantu 関数は、空白の状態で notepad.exe を起動する。ここでユーザーが適切な入力を行い、ファイル保存すれば、以後同じ状況でこのファイルがヘルプ・ファイルとして使用される。既存のファイルが表示された場合も、ユーザーが内容を追記して上書き保存した上で終了すれば、修正されたファイルが以後のヘルプ表示に使用される。

なお、Pembantu 関数は、ファイル名がフルパスで指定されていた場合には、ヘルプ・ファイルが格納されているディレクトリ以外の場所にあるテキストファイルを表示する。ヘルプ以外の局面においても使用される場合がある。例えば、報告書作成機能においては、現在編集中の地物のグループ総数、表示面総数、参照外部ファイル、適用しているマテリアルやテクスチャ等を集計した上で TEMP ライブラリにテキストファイルとして保存した上で、Pembantu 関数を用いて表示を行う。また、平面編集機能においては、現在編集中の面の頂点座標を出力し、Pembantu 関数で表示する機能を用いている。

現時点で実装されているヘルプ・ファイルの一覧を、リスト１１－４に示す。

リスト１１－１：ヘルプ・ファイル一覧

```
1997/09/01  14:55               419 Antiarea.txt
1997/09/01  14:55               292 dispkein.txt
1998/05/16  10:13             1,396 Editligh.txt
1997/10/15  00:27               775 Editmate.txt
1998/11/13  09:10             1,054 Editmove.txt
2003/05/31  08:53             1,314 Editshit.txt
1998/05/09  08:34             4,234 Edittext.txt
1997/09/01  14:54               163 Elembrid.txt
1997/09/01  14:54                75 Elemgras.txt
1997/09/01  14:54               213 Elemstee.txt
1997/09/01  14:53               859 Extwnd.txt
1998/04/21  18:11               283 Grid.txt
2003/10/02  09:44             3,107 Gydlg.txt
1997/09/01  14:53               358 Haichdt.txt
1999/01/09  11:38             5,853 Haichi.txt
1997/09/01  14:52               391 Haichipr.txt
2008/09/27  23:14               783 hyoukou.txt
1997/09/01  14:52                61 Imadlg.txt
2003/10/26  23:54                82 IpErrLog.txt
2008/09/27  18:50               623 kabe.txt
1997/09/01  14:52               577 Kashiwnd.txt
1997/09/01  14:52                91 Keirownd.txt
1997/09/01  14:52               338 Kougen.txt
2003/11/07  17:03             7,180 Koumain.txt
2008/09/28  05:47               132 land.txt
1999/04/20  10:42             5,913 mainfrm.txt
1997/09/01  14:50               118 Mojidlg.txt
1997/09/01  14:50               114 Noridlg.txt
1997/09/01  14:49               498 Noriwnd.txt
2005/10/26  04:32               404 opedlg.txt
1998/04/22  19:01             1,531 Planewnd.txt
2008/09/28  06:12               392 pmove.txt
1998/12/22  14:25               348 pnts.dat
1997/09/01  14:49                87 Primcily.txt
```

```
1997/09/01   14:49              454 Primcone.txt
1997/09/01   14:49              449 Primcube.txt
1997/09/01   14:49              281 Primflco.txt
1997/09/01   14:48              137 Primflcy.txt
1997/10/15   01:01              433 Primline.txt
1997/09/01   14:48              162 Primsphe.txt
1997/09/01   14:48              889 Roadwnd.txt
1997/09/01   14:48              150 Savedlg.txt
1997/09/01   14:47              708 Shitenwn.txt
1997/09/01   14:47              331 Shutterd.txt
1997/09/10   14:35              613 sweep1.txt
1997/09/17   08:23              829 sweep2.txt
2008/09/27   23:23              228 topocut.txt
2008/09/27   23:40              772 topoedit.txt
2008/09/30   07:22            1,374 tunnel.txt
1997/09/01   14:47              341 what.txt
2003/08/28   16:28            6,949 YuuMain.txt
2003/08/28   17:41            5,598 Zaimain.txt
```

（5）プログラム中の文字列定数

景観シミュレータのプログラム中に用いる文字列定数は、初期の段階では、グラフィック・ワークステーション(EUC 系コード)と PC（Shift-JIS 系コード）のマルチ・プラットフォームの要請と、外国語に翻訳する必要性から、ソースコード中の文字列定数を、KANJI.h というヘッダファイルで定義し、その中で

```
#define KANJIxx  文字列
```

という形式でラベルに変換し、このラベルをプログラム中で使用してきた。しかし、この方法では、動的に言語を変換することができないため、多言語に対応した Ver.2.09 においては、LangConvert()関数に引数として KANJIxx を渡すことで、動的に、すなわちその時点で選択されている言語における KANJIxx に対応する文字列を取得している。

この文字列データを格納するファイルは、LangConv.yy.txt というファイル名で、言語毎のディレクトリに置く。言語毎のファイルは、

```
KANJI_xx  “文字列”
```

という形式で変換テーブルを定義する。

１１－７．外部関数及びプラグイン DLL の表示言語の協調動作

ユーザーが、外部関数、及びプラグイン DLL を多言語対応で起動する場合には、以下の

条件が整っていれば、基幹部分で選択された言語を外部関数及びプラグイン DLL で協調的に利用できる。以下、外部関数またはプラグインの名称を XXX、言語コードを YY、ヘルプを必要とするダイアログ名称を ZZZ とした時に、

①外部関数またはプラグイン DLL のビルドに使用されたリソースとヘッダーのコピーが、それぞれ XXX.rc、XXX.rc.h という名称で、ksim/bin ディレクトリに存在する。

②リソースに基づいて構築され、翻訳結果を追記した Language/YY/XXX.YY.xml が、言語毎のディレクトリに作成されている。

③上記②の xml ファイルの末尾に Kanji エントリーが作成されている（プログラム中で使用する言語依存の文字列)。

④外部関数の場合、ヘルプを記述した Language/YY/XXX.YY.txt　が作成されている。

ただし、旧バージョンのインストール環境との互換のため、ヘルプの ja（日本語） に限って、従来の Ksim/Help にあっても正常に動作する。プラグイン DLL の場合には、基幹部分のヘルプ表示機能が利用できるため、ファイル名は、任意に設定することができ、通常、基幹部分と同様 Language/YY/ZZZ.YY.txt という名称となる。

外部関数、及びプラグイン DLL は、起動時点で bin/Lang.ctl を開き、基幹部分が現在使用している言語の種類を知る。この内、①から②を作成する機能に関しては、プログラミングを容易化するために、共通の機能として、sim.exe 本体の側で、外部関数及びプラグイン DLL を起動する時点で、xml ファイルを外部関数またはプラグイン DLL のリソースから新規作成または更新（タイムスタンプ比較による）を行っている。従って、外部関数またはプラグイン DLL の側でリソースを確認し xml ファイルを作成する必要はない。

外部関数側で共通の②③④のファイルを利用する機能に関しては、LocalLang.cpp でコーディングし、LocalLang.lib というスタティック・ライブラリに集約してあるため、開発にあたっては、これをリンクに加えると共に、必要な部分でライブラリ関数を実行することにより、多言語に対応した外部関数、プラグイン DLL を開発することができる。

１１－８．外部関数及びプラグイン DLL における多言語機能の実装方法

（1）多言語版の外部関数、及びプラグイン DLL の作成方法

外部関数を多言語で使用するためには、メニュー項目、ダイアログの各コントロール（ボタン、チェックボックス、固定的文字表示など)、ヘルプ、メッセージ等の表示を、複数言語の中から現在選択されている言語に切り替えて表示できる条件を実現する必要がある。更に、言語により必要なボタンサイズの横幅が異なるため、漢字系とアルファベット系の二つのレイアウトを用意する必要がある。

実行段階では、メイン側で、外部関数のリソースファイルを解析し、外部関数が言語の切り替えに必要とする情報を格納した言語別の XML ファイルを構築してから外部関数を起動する。従って、外部関数の側では、本体から指定された言語の XML ファイルを用いて、

ダイアログを構築する処理だけを行えば良い。

外部関数を開発するプログラマにより、少なくとも漢字系とアルファベット系の２種類の表示レイアウト（リソースファイルで定義されたダイアログテンプレート）が実現されていれば、後はプログラミング作業なしに、翻訳者によるテキストファイルの作成のみで、第三の新しい言語への対応が可能となる。

（２）実行環境

要求された言語での表示を行うためには、以下のファイルが、必要なディレクトリに存在することが条件

・ksim/bin ディレクトリに、[関数名]_D.rc というリソースファイルがコピーされていること。

・ksim/bin ディレクトリに、[関数名]_D.rc.h という名前に修正した、リソースのヘッダファイルが存在していること。

なお、開発環境が標準的に生成するリソースのヘッダファイルの名称は一律に、「resource.h」であるため、基幹部分のリソースや他の外部関数、プラグイン DLL 等と区別することができない。そこで、各機能の開発においては、上記の名称に変更した上で、このヘッダファイルを参照する部分にも対応する修正を施した上でビルドを行う。

（３）起動前の準備処理

外部関数を起動する sim.exe の側では、プロセス起動に先立って、以下の準備処理を行う。

① 要求された言語に対応する xml ファイルが存在しない場合には、新たに上記のリソースとヘッダーから作成する。xml ファイルの所在と名称は以下の通り：

[LangPath]¥[言語名]¥[関数名].[言語名].xml

例： ksim¥MultiLang¥ja¥sample.ja.xml

② 存在する場合には、リソースファイルと xml ファイルのタイムスタンプを比較し、リソースのほうが新しければ更新をかける。その際に、新たなダイアログやコントロールが追加されていれば、これに対応する新たなエントリーを作成し、翻訳結果を空白文字列とする。削除されたダイアログやコントロールについてはそのまま残す。

③ ksim/bin ディレクトリに選択された言語を記録した lang.ctl ファイルを作成（更新）する。

このファイルは、sim.exe が次に起動した際の初期表示言語の選択と、外部関数やプラグイン DLL の起動時における表示言語の選択に使用される。

lang.ctl ファイルには、以下の情報が含まれている：

第１行　言語コード　　例：ja

第２行　言語ディレクトリ　　例：c:¥@keikan¥ksim¥Language

（４）外部関数・プラグイン DLL の側でのダイアログ構築処理

起動する言語の側では、以下の処理を行う。

① 起動された実行形式自身が存在するディレクトリに lang.ctl ファイルが存在すれば

これを用いて表示すべき言語を認識し、必要な xml ファイルをロードする。存在しなければ、日本語として処理する。

② xml ファイルの定義に従って、各ダイアログのメニュー、コントロールの表示を切り替える。その際に、言語が２バイト系か１バイト系かを識別して、ダイアログの配置に使用するリソースを選択する。

リソースにおけるダイアログは、２バイト系に関して［ダイアログ名］、１バイト系に関して［ダイアログ名］_0 の名称を付して識別する。

リスト１１－２：プラグイン DLL における多言語化のための追加部分

関連するソースの冒頭部分

```
#ifdef MULTI_LANG
#include "LocalLang.h"
extern CLandApp theApp;
#endif
```

OnCreate（ダイアログのリソースの選択とメニューの設定）

```
#ifdef MULTI_LANG
        if(IsKanjiDlg("land.dll")){
                if(!m_Prm_Bar.Create(this, IDD_LAND_DLG_HYOUKOU,

        CBRS_ALIGN_RIGHT | CBRS_TOOLTIPS | CBRS_FLYBY,

        IDD_LAND_DLG_HYOUKOU)) {
                        TRACE0("m_Prm_haichi_Bar create faild¥n");
                        return -1;
                }
                MultiLangDialogConv("IDD_LAND_DLG_HYOUKOU", (CDialog*)&m_Prm_Bar);
        }
        else{
                if(!m_Prm_Bar.Create(this, IDD_LAND_DLG_HYOUKOU_0,

        CBRS_ALIGN_RIGHT | CBRS_TOOLTIPS | CBRS_FLYBY,

        IDD_LAND_DLG_HYOUKOU_0)) {
                        TRACE0("m_Prm_haichi_Bar create faild¥n");
                        return -1;
                }
                MultiLangDialogConv("IDD_LAND_DLG_HYOUKOU_0", (CDialog*)&m_Prm_Bar);
        }

        if(IsMultLangOK()){
                CMenu* pMenu = GetMenu();
                MultiLangMenuConv(pMenu,"IDR_MENU_LAND_HAICHI", 0, 0);
        }
#endif
```

OnInitDialog（ダイアログの表記文字の張替を行う）

```
BOOL CLandMenu::OnInitDialog(){
        CDialog::OnInitDialog();
        // 初期化の補足処理
        SetIcon(AfxGetApp()->LoadIcon(MAKEINTRESOURCE(IDI_SIM)),1);
#ifdef MULTI_LANG
        if(IsKanjiDlg("land.dll")){
                MultiLangDialogConv("IDD_LAND_DLG",(CDialog*)this);
        }else{
                MultiLangDialogConv("IDD_LAND_DLG_0",(CDialog*)this);
        }
#endif
        return TRUE;
}
```

終了処理

```
#ifdef MULTI_LANG
        MultiLangEnd();
#endif
```

（５）外部関数及びプラグイン DLL における文字列定数

選択的にセットアップされるプラグイン DLL と外部関数共に、翻訳を必要とする文字列は、本体が用いているデータに加えることができないため、独自のデータとして用意する必要がある。そこで、言語別 xml ファイルのルート・ノードとして、<Kanji>タグを使用し、このタグの中に、個々のプラグイン DLL および外部関数が使用する文字列定数を格納する方法を LocalLang ライブラリが提供している。

LocalLang.lib の中の、Kanji(“KANJIxx”)関数を用いて、現在選択されている言語での文字列を取り出すことができる。

ComboBox に初期条件としてセットする文字列は、単一言語であれば、リソースファイルの中に格納することも可能であるが、選択された文字列をマッチングすることも考えると、文字列を外部化し、明示的な初期化ルーチンの中で、文字列の初期値をセットする方法が便利である。その際に使用する初期化用、及び検証用の文字列を、文字列定数として、xml ファイルの<Kanji>タグの中で定義することができる。

エラー・メッセージの処理は、外部関数とプラグイン DLL で異なっている。プラグイン DLL の場合には、本体(sim.exe)からエクスポートされるｚ３ライブラリ関数を利用することができる。従って、本体側で用意しているエラー・メッセージ集で提供されているメッセージであれば、これを利用することができる。それ以外の文字列定数は独自に用意しなければならない。

外部関数の場合には、本体とは独立したビルドとなるため、多言語で表示するエラー・メッセージを全て独自にプログラムしなければならない。

そこで、外部関数及びプラグイン DLL の多言語処理のために用意した LocalLang.lib ライブラリでは、ダイアログやメニューを XML から構築する機能を中心とする諸関数に加えて、文字列定数を言語別に利用するための Kanji()関数が用意してある。本体側が自動生成し、これに翻訳家が訳を追記する言語別の XML ファイルの末尾(<SimResource>タグの内側)に、<Kanji>・・・<Kanji/>タグを設け、この中に、

リスト１１－３：Kanji()関数のためのデータの形式

```
<Kanji>
  <KANJI_1 replace=”訳語１”/>
  <KANJI_2 replace=”訳語２”/>
     ・・・・・・・・・・
</Kanji>
```

という形で文字列定数を定義しておくことにより、MessageBox（）関数等を用いて表示する文字列を、言語別に定義することができる。LocalLang.lib の Kanji()関数の引数として上記の「KANJI_1」などのタグを文字列として渡すことにより、現在選択されている言語の XML ファイルにおけるこのタグ中の「replace=」で定義された文字列が返される。これを用いることにより、エラー・メッセージや、コンボボックスの初期化、文字列比較などの処理を行うことができる。

（６）外部関数及びプラグイン DLL におけるヘルプの作成方法の実際

プラグイン DLL においては、基幹部分の関数をインポートすることにより利用できるため、基幹部分と同様の方法で、ヘルプを表示することができる。

外部関数の場合には、本体側の Pembantu、PembantuX 関数が使用できないため、ヘルプ表示を行う機能を独自に作りこまなければならない。特に、ヘルプ・ファイルを格納するディレクトリを環境変数から取得するためには、ＥＮＶライブラリを使用する必要があり、これをリンクすると、小さい単純なダイアログであっても不必要に大きくなる。そこで、本体側から使用言語に関する情報を提供する Lang.ctl ファイルの２行目に、翻訳結果を格納した xml ファイルの所在をフルパスで記述している。この文字列から、ヘルプ・ファイル等の所在を得る方法が、簡便である。このための関数を、LocalLang.lib ライブラリの中に用意した。即ち、LocalLang ライブラリが提供している PembantuX(テキストファイル名)関数を用いることにより、言語別のディレクトリに格納してあるヘルプ表示用のテキストファイルを選択的に表示することができる。

例えば、日本語が選択された状態で PembantuX(“yyy.txt”)を起動すると、言語別のディレクトリ ja にある、yyy.ja.txt というテキストファイルを表示する。

また、より簡便に、引数なしで Bantu()関数を起動することにより、外部関数名またはプラグイン DLL 名と同じ名称を有する言語別のテキストファイルを表示することができる。

例えば、Cube_D.exe 関数の中で韓国語が選択された状態で Bantu()関数(引数なし)を起動すると、言語別のディレクトリ ko の下にある、Cube.ko.txt というファイルを表示する。

（7）外部関数・プラグイン DLL の多言語対応におけるプログラマの仕事

開発する新しい外部関数の例として[例] newfunc のダイアログ部を newfunc_D.exe とする。ダイアログは、外部関数の引数パラメータを入力し、これを実際に形状生成する newfunc.exe に渡すための小さなファイルを作成することが仕事である。

プラグイン DLL の場合には、[例] newdialog.dll とする。プラグイン DLL は、ダイアログから形状生成、表示を含む、全ての機能を実装することができる。ダイアログも複数もつことができる。

当初から多言語版として外部関数を開発する場合には、以下の手順に従う。

①プログラミングに先立って、リソースのためのヘッダーを、最初にデフォルトの resource.h から変更して、newfunc_D.rc.h としておくと便利である。 リソースファイルと、ダイアログ・ハンドラの冒頭部分の#include も、これに合わせて修正する。

②２バイト系のダイアログを日本語表示でデザイン・レイアウトする。

プログラムをコーディングし（ダイアログが表示できれば可）、実行形式 newfunc_D.exe を作る。日本語だけを用いて表示テストを行う場合には、プリプロセッサ MULTI_LANG を定義しない状態でビルドを通す。

③EXT.TAB 関数に、newfunc を登録する。

④Newfunc_D.exe およびリソースとヘッダーを、ksim/bin ディレクトリにコピーし、これらを用いて、日本語表示モードで関数を sim.exe から起動する。この段階で、関数の本来の機能をデバッグする。

⑤この時 sim.exe が、Language/jp ディレクトリに生成する newfunc.ja.xml ファイルを、別フォルダまたは別名称で保存しておく。日本語表記部分を、翻訳結果としてあとで利用する。

⑥リソース・エディタで、２バイト系のダイアログの表記を日本語からローマ字に修正する（この時、ボタン等からはみ出しても気にしない）。

２バイト系のリソースを日本語としたままの場合、日本語が表示できない環境での言語切り替え処理に支障が生じる可能性がある（文字列比較などでエラーを生じ当該言語での翻訳結果に正常に切り替わらない）。また、日本語を出発点として翻訳できない場合に、翻訳家の作業が困難となる。

⑦ newfunc_D.rc を、テキスト・エディタで開き、各ダイアログをコピーし、コピーしたダイアログの名称に「_0」を追加して、１バイト系の表示用ダイアログとする。

リスト１１－４：リソースの中のダイアログのデザイン定義の編集例

編集例：

リソース・エディタで作成したダイアログの配置と構成

```
IDD_DIALOG_SEL_LANG DIALOGEX 0, 0, 167, 94
STYLE DS_SETFONT | DS_MODALFRAME | DS_FIXEDSYS | WS_POPUP | WS_CAPTION | WS_SYSMENU
CAPTION "USER INTERFACE GENGO SENTAKU"
FONT 8, "MS Shell Dlg", 400, 0, 0x1
BEGIN
```

```
    DEFPUSHBUTTON    "OK",IDOK,110,39,50,14
    PUSHBUTTON       "CANCEL",IDCANCEL,110,65,50,14
    LISTBOX                  IDC_LIST_LANG,7,7,56,80,LBS_SORT  |  LBS_NOINTEGRALHEIGHT  |
WS_VSCROLL | WS_TABSTOP
END
```

テキスト・エディタで、リソースファイル中に作成したコピー（１バイト系言語用）

```
IDD_DIALOG_SEL_LANG_0 DIALOGEX 0, 0, 177, 94
STYLE  DS_SETFONT  |  DS_MODALFRAME  |  DS_FIXEDSYS  |  WS_POPUP  |  WS_CAPTION  |
WS_SYSMENU
CAPTION "USER INTERFACE GENGO SENTAKU"
FONT 8, "MS Shell Dlg", 400, 0, 0x1
BEGIN
    DEFPUSHBUTTON    "OK",IDOK,110,39,60,14
    PUSHBUTTON       "CANCEL",IDCANCEL,110,65,60,14
    LISTBOX                  IDC_LIST_LANG,7,7,56,80,LBS_SORT  |  LBS_NOINTEGRALHEIGHT  |
WS_VSCROLL | WS_TABSTOP
END
```

（ダイアログの ID と、配置、コントロールのサイズを示す画面座標値以外は同じである）

⑧ newfunc_D.rc.h をテキスト・エディタで開き、上記のコピーしたダイアログ名称に、他と重複しない ID 数値を定義する。

⑨グラフィックなリソースエディタ（開発環境）で newfunc_D.rc を開き、１バイト系のダイアログ_0 をローマ字表示でレイアウトする。多くの場合、アルファベット表記の方が横長となるため、ボタン等を横に長くデザインする必要がある。

⑩すでに④⑤で作成してある、newfunc.ja.xml のコピーと、ローマ字表記の newfunc_D.rc を、二つのテキスト・エディタで開き、newfunc.ja.xml の original = "日本語表記" replace = "" というペアを、original ="ローマ字表記" replace = "日本語表記"　となるように修正する。

長い文字列等の場合、replace="" の側にローマ字表記を入力した上で、original= を noriginal= と簡単に修正する。更に、既にある original="日本語" を noriginal="日本語" と簡単に修正しておく。最後に、全文対象置換をかけて noriginal → replace、nreplace → original 等と一括処理すると能率的である。

⑪リソースファイル newfunc_D.rc と、ヘッダファイルを newfunc_D.rc.h として、再度 ksim/bin ディレクトリにコピーする。MULTI_LANG プリプロセッサを定義した状態でビルドした、newfunc_D.exe に更新する。

修正（⑩）した日本語表記の newfunc_D.ja.xml を、Language/ja にコピーする。

⑫sim.exe から、この外部関数を、開発に使用した１バイト系言語と、２バイト系言語で起動する。

⑬日本語が正常に変換表示されることを確認する。

⑭新たに生成された、Language/id/newfunc_D.id.xml に訳語を加える（この例ではインドネシア語）。

⑮生成した二つの xml ファイルの各メニュー項目、ダイアログ、コントロールのエントリーに、最終的に表示したい文字列を入力する。なお、リソース定義と同一の表示で良い場合には省略が可能である。

⑯確認と検査を行う。

以上の作業の後、リソース、ヘッダー、二つの言語の XML ファイルをバックアップした上で、関数を二つの言語で実行し、翻訳結果が正確に表示されることを確認する。

次に、二つの xml ファイルをテキスト・エディタで開く。各ダイアログの最後に、新たなエントリーが追加され、訳語が空欄(replace="")となっていないか確認する。もし存在する場合、翻訳作業漏れ、original="ローマ字表記"のミスタイプ（大文字小文字、余白の数等に注意）等が疑われるため、対応する項目を修正した上で、システムが自動的に追加したエントリーを削除する。修正後、再度検査を行い、XML ファイルに変化が無いことをもって終了とする。リソースのミスタイプを修正する場合には、開発環境にあるリソースと、ksim/bin にコピーしたリソースの両方を更新し、同一とする必要がある。

日本語で既に開発されている外部関数の多言語化を行う場合には、言語切り替えの関数を追加することで多言語化ができる。

①各ダイアログの構築部、InitDialog 関数に修正を加え、言語別にダイアログテンプレート、メニュー、及びダイアログの文字列表示を行うようにする。

②リソース・エディタで各ダイアログのコピーを作成し、ダイアログ名_0 という名称とする。これを、１バイト系（欧文）の文字列にふさわしい寸法に調整する。

③リソースとヘッダーの名称を、多言語対応に修正する。

外部関数名_D.exe の場合には、外部関数名.rc および 外部関数名.rc.h

プラグイン名.dll の場合には、プラグイン名.dll.rc 及びプラグイン名.dll.rc.h

とする。

以後は、新たに多言語用として開発する場合の④以下と同様である。

１バイト系外国語で既に開発されている外部関数を多言語化する場合も、本質的には同様である。日本語の訳語をｘｍｌファイルに追記するためには、日本語が入力・表示できる OS 環境が必要であるが、日本語が表示できない環境であっても、日本語を入力する前段の、訳語がまだ空欄のままのｘｍｌファイルを作成するところまでは実行可能である。

（８）外部関数・プラグイン DLL に関する翻訳家の仕事

例えば、newfunc_D.exe というダイアログの、新たな言語 yy で使用する場合、

①出発点として使いやすい XML ファイルをコピーして newfunc.[新たな言語コード].xml (newfunc.yy.xml)というファイルを作成する。

②各項目について翻訳作業を行い、結果を各エントリーの replace="" の部分に、replace="[言語 yy による翻訳結果]" という形で入力する。

③Langconv.yy.txt、ERR_MSG.yy.txt についても、既存の言語のファイルのコピーを基

に翻訳を行う。

④ヘルプファイル（多数）の翻訳を行い、ファイル名を修正する。

以上の翻訳結果を、対応する言語ディレクトリにコピーする。

プログラムのバージョンアップ、デバッグ等に伴い、新たなコントロールが追加されたり、オリジナルの表示文字列が修正されたりした場合にも、翻訳家による訳語の追加補正作業が必要となる。

オリジナルの表示文字列が修正された場合には、以前の翻訳結果が残っているので、replace="" の形で空欄となっている項目のみ訳語を補えば良い。

（9）外部関数及びプラグイン DLL のインストールの方法

①外部関数

EXT.TAB への登録。これはアルファベットで記述された外部関数名称だけを使用しているため、多言語対応は必要ではない。

Ksim/bin ディレクトリに実行形式と XXX.rc、XXX.rc.h を置く。

Language/yy ディレクトリに、XXX.yy.xml を置く

②プラグイン DLL

PLUGIN.YY.TAB という名称で多言語化したファイルに、各言語に翻訳した機能名称と、実行形式の名称を登録する。このファイルは、メニューに追加する各国語での機能名称と、実行形式の名称を登録する必要があるため、使用する言語に対応する Language/yy ディレクトリに置く。

同じ Language/yy ディレクトリに、XXX.yy.xml を置く。

DLL の本体と、リソース、およびヘッダーを ksim/bin ディレクトリに置く。

１２．環境設定

はじめに

景観シミュレータは、マルチ・プラットフォームを前提として開発されたため、Windows アプリケーションで一般的に行われている、レジストリによる実行環境設定は行わず、セットアップに含める環境設定ファイルにより動作環境を定義している。このため、アン・インストールは、ディレクトリごと、関連するファイル等を削除するだけでクリアすることができるため、地元説明準備作業やイベント開催中などに短期間集中的に利用し、終了後はアン・インストールするような用途にも適している。

環境設定に係る諸機能は、ＥＮＶライブラリ（４－２参照）に集約されている。システム起動時、環境設定の初期化は最優先されるべき事柄である。特に、エラーメッセージは、環境設定の初期化が行われた後でないと、メッセージ・ストリングを正しく取得することはできない。従って、環境設定の初期化で生じるエラーは、各言語で表示することのできない深刻なエラーとなる。このため、この初期化プロセスは、エラーを発して処理を断念する前に、いくつかの可能なセットアップ状況を仮定した、環境設定初期化を成功させるような試行錯誤的処理も行っている。

１2－１．環境設定ファイル

景観シミュレータ、景観データベースの動作環境は、環境設定ファイル（デフォルト名称 KDBMS.SET）で定義する。使用する環境設定ファイルの所在と名称は、システムの環境変数 KSIM_ENV で、フルパスで指定する。従って、異なる環境設定を定義した KDBMS_2.SET といった、異なる名称のファイルを別途用意しておき、環境変数を変更することにより、異なるユーザーやプロジェクトのために使い分けることもできる。

通常は、セットアップ時点で、ユーザーが指定したインストール先に対応して、インストーラが自動的に設定するため、ユーザーが手動で設定する必要はないが、ログインしたユーザーにより異なる環境を設定する必要がある場合などは、手動で設定することができる。インストーラが設定する環境変数は、各種実行形式(.exe, .dll 等)が置かれる BIN ディレクトリの KDBMS.SET ファイルである。

景観シミュレータ、景観データベース等が起動されたときに、初期化の過程で、まず環境変数 KSIM_ENV が調べられ、設定されている場合には、ENV ライブラリの機能により、環境設定ファイルが読み込まれ、これに従って以後の処理が行われる。

環境変数が定義されていなかった場合には、景観シミュレータなどの実行形式が起動したディレクトリにある、KDBMS.SET ファイルを使用する。このファイルも開くことができなかった場合であっても、動作環境にはそれぞれデフォルト値が設定される。

KDBMS.SET の各行は、

シンボル名＝設定値；

の形式で記述する。最初のカラムが＃から始まる行は、コメント行として無視されるた

め、一時的に環境を変更する場合などは、行頭に＃を付けて、修正前の環境のメモを残しておくことができる。

各エントリー名、それぞれのエントリーに定義されるべきデータの形式を、ヘッダーファイル e3env.h の中で定義している。データ型の種類をリスト１に示す。

リスト１２－１：各エントリーに定義されるデータ型の一覧(e3env.h の一部抜粋)

```
#define E3_TYPE_HOME_PATH          0    インストール先のディレクトリのフルパス
#define E3_TYPE_FILE_PATH          1    ディレクトリのフルパス表記またはホームパスからの相対表記
#define E3_TYPE_FILE_NAME          2    ファイル名
#define E3_TYPE_SELECT             3    複数の文字列選択肢の内の一つ
#define E3_TYPE_LONG_1             4    長い整数一つ
#define E3_TYPE_LONG_2             5    長い整数二つ
#define E3_TYPE_LONG_3             6    長い整数三つ
#define E3_TYPE_LONG_4             7    長い整数四つ
#define E3_TYPE_FLOAT_1            8    浮動小数一つ
#define E3_TYPE_FLOAT_2            9    浮動小数二つ
#define E3_TYPE_FLOAT_3           10    浮動小数三つ
#define E3_TYPE_FLOAT_4           11    浮動小数四つ
#define E3_TYPE_STRING            12    文字列
```

次に、各エントリーと、対応するデータ型を定義している部分を、e3env.h から抜粋したものをリスト２に示す。プログラムで使用可能なプロムプト用のエントリー名称（説明）もここで定義されているので、説明的である。2009 年当初時点でのエントリー数は、61 となっている。このエントリーで定義されていないキーワードが KDBMS.SET の中で使用されていた場合、警告メッセージが表示され、処理は続行される。エントリーで定義されているキーワードが含まれていなかった場合には、デフォルト値が用いられる。同一のキーワードが複数回用いられていた場合には、エラー表示等は行わず、最後の定義を用いる。

リスト１２－２：各エントリーの定義(e3env.h の一部抜粋)

```
static e3MngTbl e3mng[E3_PARAM_COUNT] = {
{E3_TYPE_HOME_PATH,   "HOME_PATH", "ホームディレクトリ", "",NULL},
{E3_TYPE_FILE_PATH,   "BIN_PATH", "実行ディレクトリ", "",NULL},
{E3_TYPE_FILE_PATH,   "TEMP_PATH", "テンポラリファイルのディレクトリ", "",NULL},

#ifdef MULTI_LANG
{E3_TYPE_FILE_PATH,   "E3_LANG_PATH", "テンポラリファイルのディレクトリ", "",NULL},
#endif

{E3_TYPE_STRING,      "UNIT", "単位", "",NULL},
{E3_TYPE_FLOAT_3,     "CLEAR_COLOR", "クリアカラー", "",NULL},
{E3_TYPE_FLOAT_1,     "GRID_SIZE", "グリッドサイズ", "",NULL},
{E3_TYPE_FLOAT_3,     "GRID_COLOR", "グリッドカラー", "",NULL},
{E3_TYPE_LONG_4,      "WINDOW", "画面サイズ", "",NULL},
{E3_TYPE_SELECT,      "DOUBLE_BUFFER", "ダブルバッファー", "ON OFF",NULL},

{E3_TYPE_STRING,      "COLOR_PRINT", "カラー印刷コマンド", "",NULL},
{E3_TYPE_STRING,      "COLOR_PRINTER", "カラープリンタ名", "",NULL},
{E3_TYPE_STRING,      "SCANNER", "スキャナ入力コマンド", "",NULL},

{E3_TYPE_LONG_2,      "SPHERE", "球コマンドの分割数", "",NULL},
{E3_TYPE_LONG_1,      "SEGS", "角柱、円柱コマンドの角数", "",NULL},

{E3_TYPE_SELECT,      "EMPHASIS_INDICATION_TYPE", "拡張表示タイプ", "OUTLINE1 OUTLINE2 OUTLINE3 OUTLINE4
OUTLINE5",NULL},
```

```
{E3_TYPE_FLOAT_4,     "EMPHASIS_INDICATION_COLOR", "拡張表示色", "",NULL},

{E3_TYPE_FILE_PATH,   "FILE_PATH_SNAPSHOT", "スナップデータのディレクトリ", "",NULL},
{E3_TYPE_FILE_PATH,   "FILE_PATH_DXF", "ＤＸＦデータのディレクトリ", "",NULL},

{E3_TYPE_FILE_PATH,   "FILE_PATH_LATEX", "ＴｅＸソースのディレクトリ", "",NULL},
{E3_TYPE_FILE_PATH,   "FILE_PATH_LATEX_BIN", "ＴｅＸの実行ディレクトリ", "",NULL},
{E3_TYPE_FILE_PATH,   "FILE_PATH_LATEX_STYLE", "ＴｅＸスタイルのディレクトリ", "",NULL},

{E3_TYPE_SELECT,      "SEARCH_MODE", "データベース検索モード", "AND OR CLEAR ALL",NULL},
{E3_TYPE_SELECT,      "JIREI_HIYOU", "優良景観事例検索画面の費用メニュー基準値", "10 100 1000 10000",NULL},
{E3_TYPE_SELECT,      "YOUSO_KAKAKU", "景観構成要素検索画面の価格メニュー基準値", "10 100 1000 10000",NULL},

{E3_TYPE_FILE_PATH,   "FILE_PATH_MASTER_DB", "マスターデータのディレクトリ", "",NULL},
{E3_TYPE_FILE_PATH,   "FILE_PATH_MASTER_SCENE", "マスターシーンデータのディレクトリ", "",NULL},
{E3_TYPE_FILE_PATH,   "FILE_PATH_MASTER_GEOMETRY", "マスタージオメトリデータのディレクトリ", "",NULL},
{E3_TYPE_FILE_PATH,   "FILE_PATH_MASTER_IMAGE", "マスターイメージデータのディレクトリ", "",NULL},
{E3_TYPE_FILE_PATH,   "FILE_PATH_MASTER_MATERIAL", "マスターマテリアルデータのディレクトリ", "",NULL},
{E3_TYPE_FILE_PATH,   "FILE_PATH_MASTER_TEXTURE", "マスターテクスチャデータのディレクトリ", "",NULL},

{E3_TYPE_FILE_PATH,   "FILE_PATH_JIREI_DB", "優良景観事例データのディレクトリ", "",NULL},
{E3_TYPE_FILE_PATH,   "FILE_PATH_JIREI_SCENE", "優良景観事例シーンデータのディレクトリ", "",NULL},
{E3_TYPE_FILE_PATH,   "FILE_PATH_JIREI_GEOMETRY", "優良景観事例ジオメトリデータのディレクトリ", "",NULL},
{E3_TYPE_FILE_PATH,   "FILE_PATH_JIREI_IMAGE", "優良景観事例イメージデータのディレクトリ", "",NULL},
{E3_TYPE_FILE_PATH,   "FILE_PATH_JIREI_MATERIAL", "優良景観事例マテリアルデータのディレクトリ", "",NULL},
{E3_TYPE_FILE_PATH,   "FILE_PATH_JIREI_TEXTURE", "優良景観事例テクスチャデータのディレクトリ", "",NULL},

{E3_TYPE_FILE_PATH,   "FILE_PATH_YOUSO_DB", "景観構成要素データのディレクトリ", "",NULL},
{E3_TYPE_FILE_PATH,   "FILE_PATH_YOUSO_SCENE", "景観構成要素シーンデータのディレクトリ", "",NULL},
{E3_TYPE_FILE_PATH,   "FILE_PATH_YOUSO_GEOMETRY", "景観構成要素ジオメトリデータのディレクトリ", "",NULL},
{E3_TYPE_FILE_PATH,   "FILE_PATH_YOUSO_IMAGE", "景観構成要素イメージデータのディレクトリ", "",NULL},
{E3_TYPE_FILE_PATH,   "FILE_PATH_YOUSO_MATERIAL", "景観構成要素マテリアルデータのディレクトリ", "",NULL},
{E3_TYPE_FILE_PATH,   "FILE_PATH_YOUSO_TEXTURE", "景観構成要素テクスチャデータのディレクトリ", "",NULL},

{E3_TYPE_FILE_PATH,   "FILE_PATH_ZAIRYO_DB", "景観材料データのディレクトリ", "",NULL},
{E3_TYPE_FILE_PATH,   "FILE_PATH_ZAIRYO_SCENE", "景観材料シーンデータのディレクトリ", "",NULL},
{E3_TYPE_FILE_PATH,   "FILE_PATH_ZAIRYO_GEOMETRY", "景観材料ジオメトリデータのディレクトリ", "",NULL},
{E3_TYPE_FILE_PATH,   "FILE_PATH_ZAIRYO_IMAGE", "景観材料イメージデータのディレクトリ", "",NULL},
{E3_TYPE_FILE_PATH,   "FILE_PATH_ZAIRYO_MATERIAL", "景観材料マテリアルデータのディレクトリ", "",NULL},
{E3_TYPE_FILE_PATH,   "FILE_PATH_ZAIRYO_TEXTURE", "マスターテクスチャデータのディレクトリ", "",NULL},

{E3_TYPE_FILE_PATH,   "FILE_PATH_TXT_DB", "テキストデータのディレクトリ", "",NULL},

{E3_TYPE_FILE_NAME,   "CLASS_FILE_JIREI1", "優良景観事例の構成種別定義ファイル名", "",NULL},
{E3_TYPE_FILE_NAME,   "CLASS_FILE_JIREI2", "優良景観事例の事業種別定義ファイル名", "",NULL},
{E3_TYPE_FILE_NAME,   "CLASS_FILE_JIREI3", "優良景観事例の建設種別定義ファイル名", "",NULL},

{E3_TYPE_FILE_NAME,   "CLASS_FILE_YOUSO1", "景観構成要素の構成種別定義ファイル名", "",NULL},
{E3_TYPE_FILE_NAME,   "CLASS_FILE_YOUSO2", "景観構成要素の種別Ａ","",NULL}, /*010916 DR.H.K.*/
{E3_TYPE_FILE_NAME,   "CLASS_FILE_YOUSO3", "景観構成要素の種別Ｂ","",NULL}, /*010916 DR.H.K.*/

{E3_TYPE_FILE_NAME,   "CLASS_FILE_ZAIRYO1", "景観材料の材質分類定義ファイル名", "",NULL},
{E3_TYPE_FILE_NAME,   "CLASS_FILE_ZAIRYO2", "景観材料の用途分類定義ファイル名", "",NULL},
{E3_TYPE_FILE_NAME,   "CLASS_FILE_ZAIRYO3", "景観材料の品種分類定義ファイル名", "",NULL},

{E3_TYPE_FILE_PATH,   "FILE_PATH_HELP", "ヘルプ・ファイルのディレクトリ", "",NULL},
{E3_TYPE_FILE_PATH,   "FILE_PATH_TIMER", "成熟都市シミュレータとのインターフェース", "",NULL},


{E3_TYPE_FILE_PATH,   "EXTERNAL_PATH", "外部関数", "",NULL} /*010916 DR.H.K.*/
};
```

リスト２に示した一覧において、各エントリーの最初の項はデータ型、二番目は、ラベ

ル文字列、三番目の項目は説明用文字列、四番目は、選択肢から一つを選ぶデータ型の場合に選択肢となる文字列（選択肢ではないデータ型の場合は空白文字列)、五番目は設定値を示す e3Param 構造体（メモリブロック）へのポインタ（初期値 NULL）である。

セットアップがインストールする環境設定ファイルの初期内容をリスト３に示す。

リスト１２－３：環境設定ファイルの初期設定

```
#####################################################################
## 景観シミュレータのための動作環境記述ファイル
## kdbms.set --- Initialize Setting File
## 020827 インストール用オリジナル
#####################################################################
HOME_PATH = C:¥@keikan;
#各ディレクトリは、フルパスで記述することも、HOME_PATH からの相対アドレスとすることもできる
#####################################################################
#ネットワーク環境で外部関数を探しに行く URL
#EXTERNAL_PATH = http://sim.nilim.go.jp/ksim/bin;
EXTERNAL_PATH = ksim/bin; URL で指定した場合に、セットアップされていない新しい外部関数を取得する
TEMP_PATH = ksim/temp;    一時的ファイルの作成場所
BIN_PATH = ksim/bin;      実行形式の格納場所

UNIT = m;

#CLEAR_COLOR        = 0. 0.2 0.2;
CLEAR_COLOR         = 0.6 0.6 0.8;
#CLEAR_COLOR        = 1.0 1.0 1.0;

GRID_SIZE  = 1.0 1.0 1.0;
GRID_COLOR          = 1.0 1.0 0.0;

TEX_BASE_STYLE      = 0;

WINDOW              = 70 70 450 450; /*初期画面サイズ  WinX WinY DrawAreaWidth DrawAreaHeight */
#DOUBLE_BUFFER      = on;
DOUBLE_BUFFER       = off;

#############################################################################
FILE_PATH_KDB                   = kdb;    データのセットアップ先
FILE_PATH_KSIM                  = ksim/source/sim;    Windows 版では使用していない
FILE_PATH_RESOURCE  = ksim/source/resource;    Windows 版では使用していない
FILE_PATH_SIM_RES   = ksim/main/srcsgi/sim;    Windows 版では使用していない
FILE_PATH_JIREI_RES = ksim/main/srcsgi/jirei;    Windows 版では使用していない
FILE_PATH_YOUSO_RES = ksim/main/srcsgi/youso;    Windows 版では使用していない
FILE_PATH_ZAIRYO_RES= ksim/main/srcsgi/zairyo;    Windows 版では使用していない
FILE_PATH_TEX                   = C:WIN32SAP¥WINTEX¥BIN¥WP;    Windows 版では使用していない
#############################################################################

# <============================== (1)
FILE_PATH_SCENE     = kdb/scene;      シーンファイル（LSS-S、拡張子.scn）の格納場所
# <============================== (2)
FILE_PATH_MASTER_IMAGE          = kdb/image;  画像ファイル（背景など）の格納場所
FILE_PATH_JIREI_IMAGE= kdb/jireiimg;    景観事例データベースが使用する画像の格納場所
FILE_PATH_YOUSO_IMAGE           = kdb/yousoimg;    景観構成要素データベースが使用する画像の格納場所
FILE_PATH_ZAIRYO_IMAGE          = kdb/zaiimg;    景観材料データベースが使用する画像の格納場所
# <============================== (3)
FILE_PATH_GEOMETRY = kdb/geometry;    ジオメトリファイル（LSSG、拡張子.geo）の格納場所
FILE_PATH_MASTER_GEOMETRY   = kdb/geometry;    ユーザーが作成するジオメトリの格納場所「
FILE_PATH_JIREI_GEOMETRY        = kdb/jireigeo;    景観事例データベースのジオメトリの格納場所
FILE_PATH_YOUSO_GEOMETRY    = kdb/yousogeo;    景観構成要素データベースのジオメトリの格納場所
```

```
FILE_PATH_ZAIRYO_GEOMETRY       = kdb/zaigeo;     景観材料データベースのジオメトリの格納場所
FILE_PATH_MATERIAL    = kdb/material;     マテリアル・ファイル（拡張子.mgl）の格納場所
MATERIAL_FILES  = default.mtl att2lss.mtl temp.mtl houkadai.mtl nittoko.mtl;     現在は無視している
# <============================== (4)
FILE_PATH_TEXTURE = kdb/texture;     テクスチャ（画像）の格納場所
FILE_PATH_MASTER_TEXTURE        = kdb/texture;     検討中の景観に関するテクスチャの格納場所
#FILE_PATH_JIREI_TEXTURE         = kdb/jireitxt;
#FILE_PATH_YOUSO_TEXTURE         = kdb/yousotxt;
#FILE_PATH_ZAIRYO_TEXTURE        = kdb/zaitxt;
FILE_PATH_JIREI_TEXTURE         = kdb/jtexture;        景観事例データベースのテクスチャ格納場所
FILE_PATH_YOUSO_TEXTURE         = kdb/jtexture;      景観構成要素データベースのテクスチャ格納場所
FILE_PATH_ZAIRYO_TEXTURE        = kdb/jtexture;      景観材料データベースのテクスチャ格納場所
################################################################
#FILE_PATH_SNAPSHOT = kdb/snapshot;
COLOR_PRINT             = lp;     プリンタのデバイス名
COLOR_PRINTER           = MJ800;     プリンタの機種
!COLOR_PRINT            = cap;
SCANNER                 = D:¥EPSCAN32¥EPSCAN.EXE;  スキャナのドライバ

EMPHASIS_INDICATION_TYPE    = outline;     選択した対象物の強調表示方法（輪郭線強調）
EMPHASIS_INDICATION_COLOR = 0.9 0.6 0.6;     強調表示の色
################################################################
FILE_PATH_JIREI         = kdb/jireitxt;     事例データベースのデータ(com.txt)格納場所
FILE_PATH_YOUSO         = kdb/yousotxt;     構成要素データベース(com.txt)格納場所
ILE_PATH_ZAIRYO         = kdb/zaitxt;     景観材料データベース(comltxt)の格納場所
################################################################
CLASS_PATH_JIREI1    = kdb/jireicls/jirei_c1.cls;     事例データベースのメニュー1項目の定義ファイル
CLASS_PATH_JIREI2    = kdb/jireicls/jirei_c2.cls;     事例データベースのメニュー2項目の定義ファイル
CLASS_PATH_JIREI3    = kdb/jireicls/jirei_c3.cls;       事例データベースのメニュー3項目の定義ファイル
CLASS_PATH_YOUSO1    = kdb/yousocls/youso_c1.cls;       要素データベースのメニュー1項目の定義ファイル
CLASS_PATH_ZAIRYO1 = kdb/zaicls/zai_c1.cls;       要素データベースのメニュー1項目の定義ファイル
CLASS_PATH_ZAIRYO2 = kdb/zaicls/zai_c2.cls;       要素データベースのメニュー2項目の定義ファイル
CLASS_PATH_ZAIRYO3 = kdb/zaicls/zai_c3.cls;       要素データベースのメニュー3項目の定義ファイル
################################################################
JIREI_HIYOU             = 1000;     事例データベースの費用の単位
YOUSO_KAKAKU            = 100;     要素データベースの価格の単位

SEARCH_MODE             = AND;       /* AND/OR/CLEAR/ALL */     データベース検索方法

SPHERE     = 16 20;      /* */     球面の分割数
SEGS       = 16;                    /* CYRINDER/CONE */   円柱、円錐の分割数

DEBUG = on;     デバッグ（使用していない）

################################################################
# 970723
################################################################
#ネットワーク環境では URL を参照する
FILE_PATH_TXT_DB = kdb;
FILE_PATH_JIREI_DB = kdb;
FILE_PATH_YOUSO_DB = kdb;
FILE_PATH_ZAIRYO_DB = kdb;

CLASS_FILE_JIREI1    = jireicls/jirei_c1.cls;
CLASS_FILE_JIREI2    = jireicls/jirei_c2.cls;
CLASS_FILE_JIREI3    = jireicls/jirei_c3.cls;
CLASS_FILE_YOUSO1    = yousocls/youso_c1.cls;
CLASS_FILE_ZAIRYO1 = zaicls/zai_c1.cls;
CLASS_FILE_ZAIRYO2 = zaicls/zai_c2.cls;
CLASS_FILE_ZAIRYO3 = zaicls/zai_c3.cls;

TIMER_PATH = timer;     成熟都市シミュレータとのインターフェース設定
HELP_PATH = ksim/help;     ヘルプ・ファイルの格納場所
```

```
LANG_PATH = ksim/Language     多言語処理に関するファイルのディレクトリ
#################################################################
#ネットワーク環境でファイルを探しに行くURL
#外部関数
#EXTERNAL_PATH = http://sim.nilim.go.jp/ksim/bin;
#FILE_PATH_YOUSO_DB                    = http://sim.nilim.go.jp/kdb;
#FILE_PATH_YOUSO_SCENE                  = http://sim.nilim.go.jp/kdb/scene;
#FILE_PATH_YOUSO_GEOMETRY                = http://sim.nilim.go.jp/kdb/yousogeo;
#FILE_PATH_YOUSO_IMAGE                 = http://sim.nilim.go.jp/kdb/yousoimg;
#FILE_PATH_YOUSO_TEXTURE                = http://sim.nilim.go.jp/kdb/jtexture;
#FILE_PATH_ZAIRYO_DB                  = http://sim.nilim.go.jp/kdb;
#FILE_PATH_ZAIRYO_SCENE                = http://sim.nilim.go.jp/kdb/scene;
#FILE_PATH_ZAIRYO_GEOMETRY               = http://sim.nilim.go.jp/kdb/zaigeo;
#FILE_PATH_ZAIRYO_IMAGE                = http://sim.nilim.go.jp/kdb/zaiimg;
#FILE_PATH_ZAIRYO_TEXTURE                = http://sim.nilim.go.jp/kdb/jtexture;
#FILE_PATH_TXT_DB                     = http://sim.nilim.go.jp/kdb;
#CLASS_FILE_YOUSO1                     = http://sim.nilim.go.jp/kdb/yousocls/youso_c1.cls;
#CLASS_FILE_ZAIRYO1                    = http://sim.nilim.go.jp/kdb/zaicls/zai_c1.cls;
#CLASS_FILE_ZAIRYO2                    = http://sim.nilim.go.jp/kdb/zaicls/zai_c2.cls;
#CLASS_FILE_ZAIRYO3                    = http://sim.nilim.go.jp/kdb/zaicls/zai_c3.cls;
```

なお、環境設定ファイルにおいて、＃で始まる行はコメント行として無視される。一時的に定義を変えておき、後で元に戻したい場合などは、もとの定義をコメントとして残しておくのが便利である。

システム起動時点における環境設定ファイルの読み込みと解析は、e3SetKeyword 関数により１行単位で実行され、その結果は、e3mng[i]構造体（配列）に格納され、以後システムにより利用される。

この解析に先立って、各エントリーには、リスト４に示した初期化を行っており、解析した KDBMS.SET に定義が欠如していた場合には、これがデフォルト値として用いられる。

リスト１２－４：環境変数のデフォルト設定値

```
void e3SetKeywordDefault(void)
{
1
            dbSetString(&e3mng[E3_HOME_PATH].param.path,"../..");  /*990215 DR.H.K.  menambah*/
            dbSetString(&e3mng[E3_BIN_PATH].param.path,  "ksim/bin");
            dbSetString(&e3mng[E3_TEMP_PATH].param.path,  "ksim/temp");

            dbSetString(&e3mng[E3_UNIT].param.path,  "m");
            e3mng[E3_CLEAR_COLOR].param.flt[0] = (float)0.0;
            e3mng[E3_CLEAR_COLOR].param.flt[1] = (float)0.0;
            e3mng[E3_CLEAR_COLOR].param.flt[2] = (float)0.0;
            e3mng[E3_GRID_SIZE].param.flt[0] = (float)1.0;
            e3mng[E3_GRID_COLOR].param.flt[0] = (float)1.0;
            e3mng[E3_GRID_COLOR].param.flt[1] = (float)1.0;
            e3mng[E3_GRID_COLOR].param.flt[2] = (float)1.0;
            e3mng[E3_GRID_COLOR].param.flt[3] = (float)1.0;
            e3mng[E3_WINDOW].param.lng[0] = 50;
            e3mng[E3_WINDOW].param.lng[1] = 50;
            e3mng[E3_WINDOW].param.lng[2] = 500;
            e3mng[E3_WINDOW].param.lng[3] = 500;
            dbSetString(&e3mng[E3_DOUBLE_BUFFER].param.path,  "ON");

            dbSetString(&e3mng[E3_COLOR_PRINT].param.path,  "lp");

            e3mng[E3_SPHERE].param.lng[0] = 8;
```

```
	e3mng[E3_SPHERE].param.lng[1] = 16;
	e3mng[E3_SEGS].param.lng[0] = 16;

	dbSetString(&e3mng[E3_EMPHASIS_INDICATION_TYPE].param.path, "OUTLINE3");
	e3mng[E3_EMPHASIS_INDICATION_COLOR].param.flt[0] = (float)1.0;
	e3mng[E3_EMPHASIS_INDICATION_COLOR].param.flt[1] = (float)0.0;
	e3mng[E3_EMPHASIS_INDICATION_COLOR].param.flt[2] = (float)0.0;
	e3mng[E3_EMPHASIS_INDICATION_COLOR].param.flt[3] = (float)1.0;

	dbSetString(&e3mng[E3_FILE_PATH_SNAPSHOT].param.path, "ksim/temp");
	dbSetString(&e3mng[E3_FILE_PATH_DXF].param.path, "ksim/temp");

	dbSetString(&e3mng[E3_FILE_PATH_LATEX].param.path, "ksim/latex");
	dbSetString(&e3mng[E3_FILE_PATH_LATEX_BIN].param.path, "ksim/latex/bin");
	dbSetString(&e3mng[E3_FILE_PATH_LATEX_STYLE].param.path, "ksim/latex/style");

	e3mng[E3_SEARCH_MODE].param.lng[0] = DB_SR_AND;
	e3mng[E3_JIREI_HIYOU].param.lng[0] = 1000;
	e3mng[E3_YOUSO_KAKAKU].param.lng[0] = 100;

	dbSetString(&e3mng[E3_FILE_PATH_MASTER_DB].param.path, "kdb");
#if 0
	dbSetString(&e3mng[E3_FILE_PATH_MASTER_SCENE].param.path, "kdb/scene");
	dbSetString(&e3mng[E3_FILE_PATH_MASTER_GEOMETRY].param.path, "kdb/geometry");
	dbSetString(&e3mng[E3_FILE_PATH_MASTER_IMAGE].param.path, "kdb/image");
	dbSetString(&e3mng[E3_FILE_PATH_MASTER_MATERIAL].param.path, "kdb/material");
	dbSetString(&e3mng[E3_FILE_PATH_MASTER_TEXTURE].param.path, "kdb/texture");
#endif

	dbSetString(&e3mng[E3_FILE_PATH_JIREI_DB].param.path, "jireidb");
#if 0
	dbSetString(&e3mng[E3_FILE_PATH_JIREI_SCENE].param.path, "jireidb/texture");
	dbSetString(&e3mng[E3_FILE_PATH_JIREI_GEOMETRY].param.path, "jireidb/geometry");
	dbSetString(&e3mng[E3_FILE_PATH_JIREI_IMAGE].param.path, "jireidb/image");
	dbSetString(&e3mng[E3_FILE_PATH_JIREI_MATERIAL].param.path, "jireidb/material");
	dbSetString(&e3mng[E3_FILE_PATH_JIREI_TEXTURE].param.path, "jireidb/texture");
#endif

	dbSetString(&e3mng[E3_FILE_PATH_YOUSO_DB].param.path, "yousodb");
#if 0
	dbSetString(&e3mng[E3_FILE_PATH_YOUSO_SCENE].param.path, "yousodb/scene");
	dbSetString(&e3mng[E3_FILE_PATH_YOUSO_GEOMETRY].param.path, "yousodb/geometry");
	dbSetString(&e3mng[E3_FILE_PATH_YOUSO_IMAGE].param.path, "yousodb/image");
	dbSetString(&e3mng[E3_FILE_PATH_YOUSO_MATERIAL].param.path, "yousodb/material");
	dbSetString(&e3mng[E3_FILE_PATH_YOUSO_TEXTURE].param.path, "yousodb/texture");
#endif

	dbSetString(&e3mng[E3_FILE_PATH_ZAIRYO_DB].param.path, "zaidb");
#if 0
	dbSetString(&e3mng[E3_FILE_PATH_ZAIRYO_SCENE].param.path, "zaidb/scene");
	dbSetString(&e3mng[E3_FILE_PATH_ZAIRYO_GEOMETRY].param.path, "zaidb/geometry");
	dbSetString(&e3mng[E3_FILE_PATH_ZAIRYO_IMAGE].param.path, "zaidb/image");
	dbSetString(&e3mng[E3_FILE_PATH_ZAIRYO_MATERIAL].param.path, "zaidb/material");
	dbSetString(&e3mng[E3_FILE_PATH_ZAIRYO_TEXTURE].param.path, "zaidb/texture");
#endif

	dbSetString(&e3mng[E3_FILE_PATH_TXT_DB].param.path, "txtdb");

	dbSetString(&e3mng[E3_CLASS_FILE_JIREI1].param.path, "jireicls/jirei_c1.cls");
	dbSetString(&e3mng[E3_CLASS_FILE_JIREI2].param.path, "jireicls/jirei_c2.cls");
	dbSetString(&e3mng[E3_CLASS_FILE_JIREI3].param.path, "jireicls/jirei_c3.cls");

	dbSetString(&e3mng[E3_CLASS_FILE_YOUSO1].param.path, "yousocls/youso_c1.cls");
```

```
        dbSetString(&e3mng[E3_CLASS_FILE_YOUSO2].param.path, "yousocls/youso_c2.cls");/*000114 DR.H.K.*/
        dbSetString(&e3mng[E3_CLASS_FILE_YOUSO3].param.path, "yousocls/youso_c3.cls");/*000114 DR.H.K.*/

        dbSetString(&e3mng[E3_CLASS_FILE_ZAIRYO1].param.path, "zaicls/zai_c1.cls");
        dbSetString(&e3mng[E3_CLASS_FILE_ZAIRYO2].param.path, "zaicls/zai_c2.cls");
        dbSetString(&e3mng[E3_CLASS_FILE_ZAIRYO3].param.path, "zaicls/zai_c3.cls");

        /*970728 DR.H.K. {*/
        dbSetString(&e3mng[E3_FILE_PATH_HELP].param.path, "ksim/help");
        dbSetString(&e3mng[E3_FILE_PATH_TIMER].param.path, "ksim/timer");
        /*010916 DR.H.K. */
        dbSetString(&e3mng[E3_EXT_PATH].param.path, "APAINI");
}
```

また、旧バージョンとの互換性を保つために、e3SetKeyword においては、リスト５のようなキーワードの無視（エラーメッセージの省略）と、読み換え（最新のエントリーへの変換）が行われている

リスト１２－５：環境設定ファイルにおけるキーワードの読み替えと無視

```
void e3SetKeyword( char *buf )
{
        int     i, j;
        char    *p, keywd[256], pa[256];

        if (!(p = (char *)strchr(buf, '=')))
                return;
        memset(keywd, 0x00, 256);
        memset(pa, 0x00, 256);
        for (i = 0; buf[i] == ' ' || buf[i] == '¥t'; i++);
        for (j = 0; buf[i] != ' ' && buf[i] != '¥t' &&
                                buf[i] != '='; keywd[j] = buf[i], i++, j++); /*970201 ada masalah jika huruf
kosong dlm nama seperti program files*/
        for (p++; *p == ' ' || *p == '¥t'; p++);
        for (j = 0; *p != ';' && *p != '¥0' && *p != '¥n'; pa[j] = *p, j++, p++);
/*●以下、読み替え*/
          /* 970201 ; upper compatibility */
          if (!strcmp(keywd, "FILE_PATH_SCENE")) strcpy(keywd, "FILE_PATH_MASTER_SCENE");
          if (!strcmp(keywd, "FILE_PATH_GEOMETRY")) strcpy(keywd, "FILE_PATH_MASTER_GEOMETRY");
                  /* 990311 DR.H.K. additional compatibility */
          if (!strcmp(keywd, "FILE_PATH_IMAGE")) strcpy(keywd,"FILE_PATH_MASTER_IMAGE");
          if (!strcmp(keywd, "FILE_PATH_MATERIAL")) strcpy(keywd,"FILE_PATH_MASTER_MATERIAL");
          if (!strcmp(keywd, "TIMER_PATH")) strcpy(keywd,"FILE_PATH_TIMER");
          if (!strcmp(keywd, "HELP_PATH")) strcpy(keywd,"FILE_PATH_HELP");
/*●以下無視するキーワード*/
          /*if (!strcmp(keywd, "EXTERNAL_PATH")) return; 010916 DR.H.K. ＵＲＬアクセス関数のため復活する*/
          if (!strcmp(keywd, "TEX_BASE_STYLE")) return;
          if (!strcmp(keywd, "FILE_PATH_KDB")) return;
          if (!strcmp(keywd, "FILE_PATH_KSIM")) return;
          if (!strcmp(keywd, "FILE_PATH_RESOURCE")) return;
          if (!strcmp(keywd, "FILE_PATH_SIM_RES")) return;
          if (!strcmp(keywd, "FILE_PATH_JIREI_RES")) return;
          if (!strcmp(keywd, "FILE_PATH_YOUSO_RES")) return;
          if (!strcmp(keywd, "FILE_PATH_ZAIRYO_RES")) return;
          if (!strcmp(keywd, "FILE_PATH_TEX")) return;
          if (!strcmp(keywd, "MATERIAL_FILES")) return;
          if (!strcmp(keywd, "FILE_PATH_TEXTURE")) return;
          if (!strcmp(keywd, "!COLOR_PRINT")) return;
          if (!strcmp(keywd, "FILE_PATH_JIREI")) return;
          if (!strcmp(keywd, "FILE_PATH_YOUSO")) return;
          if (!strcmp(keywd, "FILE_PATH_ZAIRYO")) return;
```

```
            if (!strcmp(keywd, "CLASS_PATH_JIREI1")) return;/*CLASS_FILE_JIREI1 etc.*/
            if (!strcmp(keywd, "CLASS_PATH_JIREI2")) return;
            if (!strcmp(keywd, "CLASS_PATH_JIREI3")) return;
            if (!strcmp(keywd, "CLASS_PATH_YOUSO1")) return;
            if (!strcmp(keywd, "CLASS_PATH_YOUSO2")) return;/*000114 DR.H.K.*/
            if (!strcmp(keywd, "CLASS_PATH_YOUSO3")) return;/*000114 DR.H.K.*/
            if (!strcmp(keywd, "CLASS_PATH_ZAIRYO1")) return;
            if (!strcmp(keywd, "CLASS_PATH_ZAIRYO2")) return;
            if (!strcmp(keywd, "CLASS_PATH_ZAIRYO3")) return;
            if (!strcmp(keywd, "DEBUG")) return;
/*●以下、ヘッダーe3env.hで定義されたキーワードとの照合による解析*/
                        /* 96.08.19 */
          for (i = 0; i < E3_PARAM_COUNT; i++) {
            if ( e3mng[i].name == NULL || strcmp(keywd, e3mng[i].name) != 0)
                        continue;

            switch (e3mng[i].type) {
（以下省略）
```

１２－２．セットアップ・ディレクトリ構成

景観シミュレータを運用するに伴い、各種のファイルが作成される。その中には、三次元形状、背景・前景画像、テクスチャ画像、マテリアル、シーンなどが含まれる。これらの種類の異なるファイルを、通常は、異なるディレクトリに格納する。それぞれの種類のファイルをどのディレクトリに格納するかを環境設定ファイルの中で定義している。必要に応じて、全ての種類のファイルを同一のディレクトリに格納する設定も、全ての種類を細かく分類して別ディレクトリに保存する設定も可能である。

環境設定ファイルは、以下の種類のファイルと格納ディレクトリを定義している：

（1）ホームディレクトリ（HOME_PATH）

セットアップ先を示す。デフォルトでは整理のために、この下に ksim ディレクトリ（プログラム関係）と kdb ディレクトリ（データ関係）を作成し、それぞれの下に環境設定で定義される各種のディレクトリを置いており、対応するエントリーとして FILE_PATH_KSIM と FILE_PATH_KDB を用意しているが、Windows 系のセットアップでは、これらを使用せず、各種ディレクトリは、ホームディレクトリからの相対パスとして定義している。

（2）プログラム関連のディレクトリ

① BIN_PATH

実行形式を格納する。

② EXTERNAL_PATH

セットアップされていない外部関数等が要求された時に、URL 参照で自動的にダウンロードするように設定する場合には、”http://...” から始まるサーバーのアドレス（URL）を記述する。

③ TEMP_PATH

プログラムが自動的に生成する一時的なファイルを格納するディレクトリを指定する。

④ HELP_PATH

ヘルプ・ファイルを格納するディレクトリを示す。なお、多言語対応版では、ヘルプ・ファイルは言語別のディレクトリに置くため、このディレクトリは、言語別ディレクトリに必要とするファイルが見つからない場合にのみ参照される（未翻訳の場合など)。その場合、日本語のヘルプ等が使用されることとなる。

⑤ LANG_PATH

言語別のサブディレクトリを置く親ディレクトリである。この下に、アルファベット２文字から成る言語コードを名前とするサブディレクトリを作成し、その下に各言語に対応する各種ファイルを置く。

（３）データ関連のディレクトリ（KDB)

(3-1)FILE_PATH_MASTER_DB マスターデータ

ファイルを開くといった操作において、最初に表示されるディレクトリを定義している。また、ファイル解析中に参照ファイルがフルパスではない形で指定された場合に、最初に探しに行くディレクトリである。

① FILE_PATH_MASTER_SCENE シーンファイル

開く LSS-S で最初に表示するディレクトリである。

② FILE_PATH_MASTER_GEOMETRY ジオメトリファイル

メニューの[ファイル][開く LSS-G]で最初に表示するディレクトリである。

③ FILE_PATH_MASTER_IMAGE イメージファイル

シーンに前景・背景を付けようとした場合に、最初に表示するディレクトリである。

④ FILE_PATH_MASTER_MATERIAL マテリアル・ファイル

オブジェクトにマテリアルを設定しようとした場合に、このディレクトリにあるマテリアル・ファイルの一覧が選択肢として表示される。現在は、全てのマテリアル・ファイルをこのディレクトリに置き、必要に応じてマテリアル・ファイルを探索する。

⑤ FILE_PATH_MASTER_TEXTURE テクスチャファイル

オブジェクトにテクスチャを設定しようとした場合に、このディレクトリにあるテクスチャファイルの一覧が選択肢として表示される。

(3-2)FILE_PATH_JIREI_DB 事例データ

① FILE_PATH_JIREI_SCENE

② FILE_PATH_JIREI_GEOMETRY

初期には景観検討事例のジオメトリを格納するためのものであったが、その後事例が蓄積され、各事例のデータは通常多数の LSS-G ファイルを含み、また異なるプロジェクトで同一名称をもつ異なる内容のファイル等も存在するため、ひとつのディレクトリに集めることは実用的でなくなった。現在では事例単位にディレクトリで管理しているため、使用していない。

③ FILE_PATH_JIREI_IMAGE

景観検討事例データベースをローカルにセットアップした場合に、検索画像を格納する。

④ FILE_PATH_JIREI_MATERIAL

上記理由(3-1④)から、使用していない。

⑤ FILE_PATH_JIREI_TEXTURE

上記と同じ理由から、現在では使用していない。

⑥ FILE_PATH_JIREI 事例検索データ格納ディレクトリ

事例データのサマリーをテキストデータとして格納した com.txt を格納する。また、検索ヒストリー等のデータや、追加登録等の編集に際してのバックアップファイル等を置く。

⑦ CLASS_FILE_JIREI1、2、3

検索項目１、２、３の各メニュー構成を示すファイルをフルパスで指定する。

(3-3)FILE_PATH_YOUSO_DB 景観構成要素データ

汎用性の高い景観構成要素を登録したものである。自然物や、公的標準仕様に基づく形状を有する地物（標識など)、メーカー等を特定しない汎用のオブジェクト（ベンチや公園遊具など）を格納している。

① FILE_PATH_YOUSO_SCENE

部品をシーンデータとして利用することが無いため、使用されていない。

② FILE_PATH_YOUSO_GEOMETRY

景観構成要素の形状を LSS-G 形式で格納している。多くのファイルが登録されている。

③ FILE_PATH_YOUSO_IMAGE

景観構成要素データベースをローカルにセットアップした場合に、検索画像を格納する。

④ FILE_PATH_YOUSO_MATERIAL

上記理由(3-1④)から、現在使用していない。

⑤ FILE_PATH_YOUSO_TEXTURE

景観構成要素の形状を記述する LSS-G ファイルから参照されるテクスチャを格納する。

⑥ FILE_PATH_YOUSO 景観構成要素検索データ格納ディレクトリ

景観構成要素データのサマリーをテキストデータとして格納した com.txt を格納する。また、検索ヒストリー等のデータや、追加登録等の編集に際してのバックアップファイル等を置く。

⑦ CLASS_FILE_YOUSO1、2、3

検索項目１、２、３の各メニュー構成を示すファイルをフルパスで指定する。

(3-4)FILE_PATH_ZAIRYO_DB 景観材料データ

当初(1993 年頃)は、バブル経済崩壊の後公共事業による経済下支えのニーズが高まる中、高速道路の遮音壁や、自然木・石を擬したコンクリート二次製品など、各種景観材料が盛んに製品化され興隆を見せていた。これに対応して土木研究所において検討委員会を設置

し、製品化された景観材料を登録して、景観検討作業に活用する構想が存在していたが、その後インターネットが発達し、各社とも WEB サイトから情報発信する方向となった。

景観材料データベースは、価格等の製品情報も格納する設計となっていたが、現時点で改めて意義を見出すとすれば、コンクリートや材木・鋼材など、素材・材質感だけが固有で、形状は自由に加工できるような物品を登録するのが実用的と考えられる。現在は、上記の景観構成要素データベースにこのような物品も登録されているが、形状が固定的であるため、あまり実用的ではない。寧ろ、選択可能な色彩一覧をデータ化したマテリアル・ファイルや、貼り方の規則を付したテクスチャデータが登録されたデータベースが便利である。

① FILE_PATH_ZAIRYO_SCENE

部品をシーンデータとして利用することが無いため、使用されていない。

② FILE_PATH_ZAIRYO_GEOMETRY

景観構成要素の形状を LSS-G 形式で格納している。多くのファイルが登録されている。

③ FILE_PATH_ZAIRYO_IMAGE

景観材料データベースをローカルにセットアップした場合に、検索画像を格納する。

④ FILE_PATH_ZAIRYO_MATERIAL

上記理由(3-1④)から、今後活用可能となる可能性がある。

⑤ FILE_PATH_ZAIRYO_TEXTURE

景観材料の形状を記述する LSS-G ファイルから参照されるテクスチャを格納する。

⑥ FILE_PATH_ZAIRYO 景観構成要素検索データ格納ディレクトリ

景観材料データのサマリーをテキストデータとして格納した com.txt を格納する。また、検索ヒストリー等のデータや、追加登録等の編集に際してのバックアップファイル等を置く。

⑦ CLASS_FILE_ZAIRYO1、2、3

検索項目１、２、３の各メニュー構成を示すファイルをフルパスで指定する。

（4）いくつかの典型的なセットアップ・ディレクトリ構成

景観シミュレータでは、ひとつの実行形式を中心に、単純なビューワから、本格的なデータベースを連動させたデータ構築作業まで行うため、様々な環境設定がありうる。いくつかの典型的なディレクトリ構成を解説する。

①スタンドアロンの環境で、フルセットでセットアップする場合

データ種類毎に格納ディレクトリ構成を細かく仕訳けると整理が良い。このため、特に KDB の下に、データベースの種類毎にディレクトリを切り分ける構成とする。

②ネットワーク環境でビューワとして利用する場合

一時的にダウンロードした三次元データを見るだけであれば、データ関連のディレクトリを一つだけとし、ここに形状、イメージ、テクスチャ、マテリアルを含む全てのファイ

ルを置く方法で十分である。この場合、環境設定ファイルの多くの項目は同じ定義内容となる。

１２－３．作業用ディレクトリ

環境設定ファイルで定義する各種ディレクトリとは別に、ある特定の景観検討業務等に関連した様々のファイルが、同一ディレクトリに集約されていた方が便利な場合がある。そこで、e3Kerja()に、あるディレクトリを記憶しておき、様々のファイル関連操作において、このディレクトリに探索に行くように環境設定することができるようにしてある。

ここに登録したディレクトリは、ユーザーの操作によって、ファイルを開く場合に最初に表示されるディレクトリ、およびインタープリタ等の内部処理の途上で、ファイル参照が必要となった場合に調べに行くパスの一つとして反映される。

作業が終了し、関連するファイルを整理保存する場合に対応するため、「報告書作成」「文書整理」の機能を用意してある。報告書作成においては、現在表示されている景観の中で参照されている全てのファイルがリストされ、文書整理においては、それらのファイルを指定したディレクトリにコピーする操作を記述したバッチファイル(.bat)を作成する。

各種のファイルを部品として参照している、つまりは複数のファイルからなるオブジェクトを保存する操作に先立って、隠し機能の中で、「direct」コマンドを実施すると、保存する際に、全ての参照部品を、メインのオブジェクトの中に直接組み込んだ大きな一本のファイルとして保存を行う。なお、このファイルの中には、元の部品ファイルに由来する部分が記録として残されており、再度部品として分割された形で保存することができる（⇒その方法)。

頻繁に参照される部品は、景観構成要素データベースに登録することができる（⇒その方法)。

１２－４．http プロトコルによるファイル取得のためのネットワーク環境

ネットワークで動作する景観シミュレータ(Ver.2.07 以降)においては、未実装の外部関数が参照された場合には、http プロトコルにより、ダウンロード用サイトから実行形式をダウンロードする機能を有している。景観構成要素や景観材料のデータベースの検索先を URL 上のサーバーに設定した場合、LSS-G 形式のデータファイルを WEB 上からダウンロードする。

環境設定においては、前者に関して EXT_DIR、後者に関して FILE_PATH_YOUSO_DB および FILE_PATH_ZAIRYO_DB 以下のエントリーを WEB サーバーの URL に設定する。この場合、通常の BIN_DIR および FILE_PATH_MASTER_DB 以下はキャッシュとして利用される。即ち、求めるファイルがキャッシュに存在すればこれを直ちに利用し、もし存

在しなければ URL からキャッシュにダウンロードした上で、これを利用する。

また、WEB ブラウザで、LSS-S ファイルへのリンクが選択された時には、このファイルをダウンロードした上で、必要な参照ファイル（LSS-G など）をダウンロードする。その場合には、LSS-S ファイルの中に記述された LSS-G ファイルが、URL 表記で定義されている。

リスト１２－６：モデルをＵＲＬで定義する LSS-S ファイルの例

```
# (ip ver.)国土交通省版・景観シミュレータ Ver. 2.05n β-010924
        T1 = TIME(0);
        CAM2 = CAMERA(-0.19282, -0.7, 1.4798
                ,0.5, 0.5, 0.5
                ,0, 0, 1
                ,64.0108, 1, 0.169706, 169.706);
        L4 = LIGHT(0, 4387.73, -2.55242e+006, 1.57643e+006, 0
                ,1, 1, 1);
        Ld = LIGHT(0, 4387.73, 2.55242e+006, 0, 0
                ,0.2, 0.2, 0.4);
        Ll = LIGHT(0, -300000, 0, 100000, 0
                ,0.2, 0.2, 0.4);
        Lr = LIGHT(0, 300000, 0, 100000, 0
                ,0.2, 0.2, 0.4);
        LG2 = LIGHTGROUP(L4, Ld, Ll, Lr);
        MDL1 = MODEL("http://sim.nilim.go.jp/nowhere/Geometry/UC.geo");
        SCN1 = SCENE(0, , , MDL1, LG2, , CAM2, T1);
```

この場合、sim.exe の側では、作業環境を、URL 上のディレクトリに設定する（上記の例の場合では、http://sim.nilim.go.jp/nowhere/Geometry）。以後、モデルとして指定された LSS-G ファイルから参照されたファイル（子グループの LSS-G、マテリアル、テクスチャ等）については、まずローカルなディレクトリで探し、もし存在しなければ、この URL 上のディレクトリからローカルなディレクトリにダウンロードを行った上で、ローカルなディレクトリから開く。

この機構により、部品ファイル等は、URL からの提供が可能となり、また一度ダウンロードしたファイルを再度開こうとした場合には、通常のローカルな部品として速やかに開くことができる。

但し、タイムスタンプの比較を行っていないため、公開されたファイルが、同じ名称のままで修正されているような場合、あるいはユーザー側で既にローカルに保有している LSS-G ファイルと同じ名称で別の部品が公開された場合に、ユーザー側の表示に反映されない恐れがある点は注意を要する。個別のまちづくりプロジェクトに固有の地物データに

関しては、一般的なファイル名（例えば、road.geo など）ではなく、ユニークな名称としておくことが望ましい。

なお、景観シミュレータを WEB ブラウザと組み合わせて運用した場合に、WEB ブラウザから起動される sim.exe に引数として渡されるファイル・アドレスは、WEB ブラウザ上で選択されたリンクではなく、WEB ブラウザが独自に管理しているキャッシュに既にダウンロードしたファイルへのローカルなアドレスである。従って、この引数から URL アドレスを取得して関連ファイルを辿ることはできない。

１２－５．背景色（CLEAR_COLOR）

メイン画面でオブジェクトの無い余白部分の色を RGB 値で定義する。各値は、0.0～1.0 の間の小数で定義する。

Ver.2.09 においては、カラー編集画面(CEditMate)で、設定を変更したり、環境設定値に戻したりすることができる。

光源計算ダイアログ(CGyDlg)で、太陽位置が水平線よりも下になった場合に、背景を黒とし、そうでない場合には、環境設定値としている。

霧の表示（光源計算ダイアログの中に選択がある）では、オブジェクトの色を、距離と霧の濃度で計算した物体色と背景色との中間色としている。

１２－６．曲面をもつ原始図形等の分割数(SPHERE, SEGS)

球（sphere.exe）、円柱(cylinder.exe)、円錐・円錐台(cone.exe)は、形状生成計算において、位置や径などのパラメータとして設定される値に加えて、環境設定値として指定された分割数を用いて実際の形状を生成している。

SPHERE = 16 20;

球の経度の分割 20、緯度の分割 16

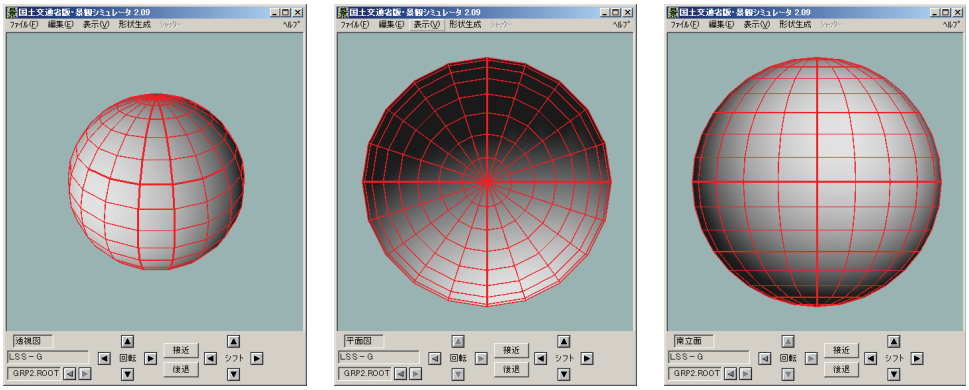


図１２－１：球の分割状況

SEGS = 16;

円柱、円錐・円錐台の経度の分割 16

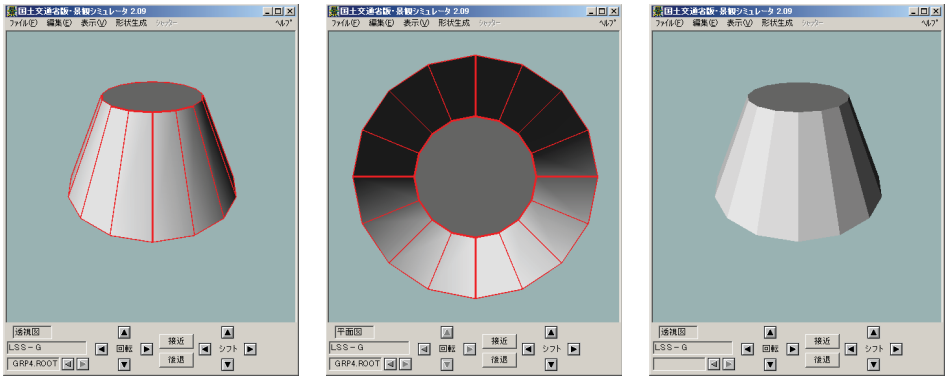


図１２－２：円錐台の分割状況

図１２－２で、比較のために円錐台と１６角錐台を右に示した。基本的な形状は同じであるが、円錐台の側面の各頂点に、隣接する側面の同一頂点と同一の法線ベクトルを定義することにより、連続的な陰影をかけ、滑らかな表現を実現している様子がわかる。

処理しようとする地物のデータ全体が小さい場合や、ハードの性能が十分大きい場合には、これらの環境設定値を大きくすることにより、近似の精度を上げ、表面積や体積を理想的な球や円錐に近づけることができる。

１２－７．グリッドの表現(GRID_SIZE、GRID_COLOR)

オルソ系画面では、編集作業を補助するためにグリッドを表示することができる。その格子間隔や表示色のデフォルト値（初期値）を規定している。

１２－８．強調表示の表現(EMPHASIS_INDICATION_TYPE、COLOR)

編集対象となるオブジェクトをメイン画面でクリックした場合に、選択・強調表示を行う。この場合の強調表示の方法と、強調表示を行う色彩を規定している。

１２－９．その他

①DOUBLE_BUFFER

OpenGL の表示におけるダブルバッファーの使用の有無を指定する。

②COLOR_PRINT

印刷する場合のポートを指定する。

③COLOR_PRINTER

印刷する場合のプリンタ名を指定する。

④SCANNER

スキャナから画像を取り込む場合のコマンド。

⑤FILE_PATH_SNAPSHOT

現在は使用していない。

⑥FILE_PATH_DXF

現在は使用していない。

⑦FILE_PATH_LATEX、LATEX_BIN、LATEX_STYLE

現在は使用していない。かつて通信実験に使用した。

⑧SEARCH_MODE

データベースの検索モード(AND, OR, CLEAR ALL)を示すためのものであったが、現在は使用していない。

⑨JIREI_HIYOU

景観事例の事業費メニュー基準値。

⑩YOUSO_KAKAKU

景観構成要素検索画面の価格メニュー基準値。

⑪FILE_PATH_TIMER

景観シミュレータを出力装置として、別システムからアニメーション等を表示する場合にデータを送り込むポートを定義する。

１３．OS と開発環境の更新へ対応

１３－１．概要

第 1 章ヒストリーでも触れたように、開発着手後１５年以上にわたって改良・メンテナンスを行ってきた本システムは、数度にわたるＯＳの更新及び開発環境の更新を経験してきた。OS を更新することにより、従来問題なく動作していた機能に障害が生じることは、現状維持を目的とする限り、大きなオーバーヘッドとなる。何も新しい機能を付け加えようとしないにもかかわらず、新たな OS の上で正常に動作させるために、多くのマンパワーを投入しなければならない。とりわけ、グラフィックス処理の規約が変更されたような場合には、同じ機能を維持するために、修正が必要となったことがある（１３－３）。

しかしながら、長期的に見ると、OS や開発環境の更新に伴い、潜在バグが具体的障害化する場合がある。Windows98 から ME に更新した際には、多くの障害が生じた。しかしながら、原因を特定すると、結果的に潜在バグが顕在化したことによる障害がほとんどであった。従って、長期的に見ると、OS の変更への対応は、無駄な努力ではなく、より信頼性の高いシステムを実現し、かつ将来のより高度な機能開発や応用への有効な布石となっていると考えている。

Window98 までは、解放済みのメモリブロックや、スタック上に構築されるオート変数へのポインタを使い続けても、そのメモリブロックの内容が書き換えられるまでは一見正常な動作を続けるような場合があり、物理的なメモリが別の目的に割り当てられ、内容が書き換えられてから、初めて異常な現象が生じるようなケースがあった。デバッグに際しては、これを発見するために、メモリブロックの解放やデータの除却に際して、明らかに異常なデータに書き換えてから解放する方法や、デバッグ用実行形式の中で、メモリブロックのリスト管理を独自に行う方法により、終了時点で検出されたリークの発生源を特定するための工夫を行った。最近の開発環境では、このような機能が次第に充実してきている。その後 OS や開発環境の進化に伴い、OS が提供するメモリ･ブロックの解放処理等が変更され、無効を示すデータが解放時に OS により書き込まれているように見える。また、オート変数やメモリブロックの、取得・定義時点でのメモリの物的初期状態も OS により変化してきたため、例えば変数を初期化しないまま使用しても、偶々メモリの内容が初期状態で 0x00 であったために一見正常に動作しているような場合に、OS の変化に伴い顕在化するような場合があった。

Windows XP から Vista への更新に際しても、いくつかの修正が必要となったが、グラフィックス関連では本質的な修正が 2 個所にとどまったことは、一種の完成度を示していると考える。

１３－２．VS2005 への対応

開発環境は、Windows 系に関しては Visual Studio 2.0→4.0→6.0→8.0(VS2005)と更新を行ってきた。

とりわけ、8.0 への更新に際しては、従来の C 言語で開発されていたライブラリ関数のソースコードの多くにおいて、多数の警告が発せられた。最も修正の手間を要した部分は、最初に UNIX マシンで開発されていたソースコードにおけるコメント等において日本語表記が EUC コードで行われていた部分の扱いである。

更に、ANSI-C の基本的な関数であった、fprintf 等の基本的な関数の内、バッファ・オーバーフローを起こす可能性のある関数に関して、使用を避けるような警告が多く生じた。

これらは、セキュリティを高めるための対応を要求するものである。

この他、各実行形式のコンパイル、リンクおよびデバッグ等の動作を指示するために、より多くの詳細な設定を行う必要があった。

現在までに、新たな開発環境のバグに起因する障害や、解決不能な障害は生じていない。しかし、プロパティの設定などを試行錯誤した結果解決した障害であっても、なぜ解決したのかが必ずしも明快ではないような障害は存在している。

１３－３．Windows Vista への対応

ＯＳの更新に伴い、潜在バグが具体的障害化する場合がある。Windows XP から Vista への移行に際しては、OpenGL の画像表示に関する部分が問題となった。

Windows における各ウィンドウは階層的に構成されている。子ウィンドウは、親ウィンドウの上に表示され、同じ親に帰属するウィンドウ相互は、アクティブなものがそれ以外のものの上に表示される。そして最上位にデスクトップ・ウィンドウが位置する。ひとつのダイアログの中に配置された OpenGL 画面やボタンやエディットボックスなどのコントロールも、同じ階層の子ウィンドウである。

OpenGL の画面は、初期化の手続きによって、ウィンドウに関連づけられる。その際に、図形描画の出力機能を担うデバイス・コンテキストが使用される。デバイス・コンテキストは、各ウィンドウに固有のものと、全てのウィンドウに共通に使用されるものがある。描画に際して、共通に使用するものを描画が必要な時点で取得し、描画が終了すると解放するようにプログラムを構成した方が、資源節約になる。更に共通のデバイス・コンテキストは、最大５まで、という制約があるため、描画終了後に解放しないまま停止するようなスレッドが存在すると、別の画面における描画が失敗する場合がある。

さらに、OpenGL をデバイス・コンテキストと結びつけるために、HGLRC というもう一つのコンテキストが必要とされる。こちらには数の上限がない。さらに、個々の OpenGL 画面の描画条件設定には多くのパラメータが必要であり、複数の画面のそれぞれに関して、一度設定したパラメータは、アプリケーションが終了するまで記憶しておく必要がある。そこで、景観シミュレータにおいては、HGLRC を OpenGL 画面作成時から除却時まで保持するとともに、各 OpenGL 画面に関して、ウィンドウ作成時に、初期化と共に DA 構造体を生成し、パラメータを保存し、ウィンドウ除却時にこれを解放している。個々のウィンドウ描画時には、DA 構造体のパラメータを用いて描画を行っている。

Windows　VISTA においては、初期化処理の終了時点で、OpenGL とデバイス・コンテ

キストの関連づけの解消をより厳密に実行しておかないと、再描画に際して、意図するウィンドウ以外の画面に描画が行われてしまう、という障害が発生した。多くの場合、最上位にあるデスクトップ・ウィンドウに表示が行われてしまう。

これに対して、最も簡単な解決方法は、従来の方法を保持しつつ、初期化の終了時点で、デバイス・コンテキストの解放をより厳密化することである。具体的には、以下のように１行を挿入する。

リスト１３－１：OpenGL 初期化部分の修正

```
pCDC = GetDC();
m_HDC = new HDC;
*m_HDC = pCDC->m_hDC;
set_pixel( *m_HDC);
m_HGLRC = wglCreateContext(*m_HDC);
wg3EntryDrawarea(((void*)&m_HDC(void*)&m_HGLRC);
wg3AssignDrawarea((void*)&m_HDC);
g3InitializeWindow();
(各種初期化)
wg3AssignDrawarea(NULL);
pCDC->ReleaseOutputDC();  //●この１行を追加
ReleaseDC(pCDC);
```

推察するに、Windows　Vista の内部処理においては、OpenGL のコンテキストと関連づけられたまま、デバイス・コンテキストが解放され共用のプールに返されると、次に描画しようとした時に、先刻解放したデバイス・コンテキストを新たに取得している別のウィンドウに描画されてしまう、という事のようである。なお、ReleaseOutputDC を、ウィンドウのメンバに関連づけられたデバイス・コンテキスト（例えば、CClientDC dc(this); 等の命令で取得したもの）に適用すると、エラーとなるため、OpenGL 画面の初期化には GetDC()等の命令で共通のプールから取得した一時的なものを使用する。

もうひとつの方法は、描画をメモリ上のデバイスに行うことにより、トリプル・バッファリングを実現する方法であり、メモリ上に最終的な表示画面と互換のデバイス・コンテキストを作成してこれに描画を行う。OpenGL のダブル・バッファが、描画途上の画面をユーザーに見せずに、描画終了後に瞬間的に切り替えることを目的としているのに対して、メモリ上のデバイスの使用は、描画コンテキストを、他のウィンドウ等との重なり具合といった外部環境から切り離した環境で行うことにある。最終的な表示は、メモリから画面へのデータ転送のみであり、三次元のレンダリングより一般に処理は速い。このメモリ上のコンテキストは、ウィンドウの作成から除却までの間保持していて構わない。

この方法においては、表示内容の変化（地物の変化、視点や光源などの環境の変化）が生じた際の OpenGL の再描画が少し遅くなる一方、単に表示画面の上を覆っていた別のウ

ィンドウが移動した際に無効領域を再描画する処理は、単にメモリのコピーだけの処理となり、高速化する。従って、視点移動アニメーションなど、最上位のウィンドウに高速で再描画を繰り返すような処理には前者が適しており、様々な画面を併用しながらモデリングを行うような場合には、後者の処理が適している。

リスト１３－２：メモリ上のデバイスを用いる方法

```
①OnCreate における処理
  CPaintDC dc(this);
  mDC.CreateCompatibleDC(&dc);
  mBM.CreateCompatibleBitmap(&dc,1,1); //とりあえず１ドットのビットマップ
  mDC.SelectObject(mBM);
  m_sHDC = new HDC;
  *m_sHDC = mDC.m_hDC;//メモリ上のデバイスを用いて初期化
  （この m_sHDC は、OnSize, OnPaint で再初期化せずに、維持する）
  set_pixel(*m_sHDC);
  m_sHGLRC = wglCreateContext(*m_sHDC);
  status = wg3EntryDrawarea((void*)&m_sHDC,(void*)&m_sHGLRC);
  if(!status) エラー処理へ
  status = wg3AssignDrawarea((void*)&m_sHDC);
  if(!status) エラー処理へ
  status = g3InitializeWindow();
  if(!status) エラー処理へ
        各種初期化処理

②OnSize(UINT nType, int cx, int cy) における処理
  CClientDC dc(this);
  *m_sHDC = dc.m_hDC;
  wg3AssignDrawarea((void*)&m_sHDC);
  g3Resize(cx,cy);
  mBM.DeleteObject();
  mBM.CreateCompatibleBitmap(&clientDC,cx,cy);
  mDC.SelectObject(mBM);
  RedrawWindow();

③OnPaint における処理
  RECT Rect;
  int rc, x, y;
```

```
   GetUpdateRect(&Rect,FALSE);
   CPaintDC dc(this);
(モデルや視点が変更されていて、画面全体を更新する必要がある場合のみ)
   g3GetSize(&x,&y);
   wg3RedrawWindow(&m_sHDC);
   rc = dc.BitBlt(0,0,x,y,&mDC,0,0,SRCCOPY);
(被さっているウィンドウが静止し、無効領域だけを更新する場合)
   int x,y,top,bottom,left,right,width,height,rc;
   g3GetSize(&x,&y);
   top = Rect.top, bottom = Rect.bottom;
   left = Rect.left, right = Rect.right;
   width = right – left;
   height = bottom – top;
   rc = dc.BitBlt(left,top,width,height,&mDC,left,top,SRCCOPY);
④OnDestroy における処理
   wg3AssignDrawarea((void*)&m_sHDC);
   g3ReleaseGroundPixel()
   wg3DeleteDrawarea(&m_sHDC);
   wglDeleteContext(m_sHGLRC)
   mBM.DeleteObject();
```

初期のグラフィクス・ワークステーションは、言わば専用のグラフィック・カードが巨大化したような構成であり、直接ハードに GL のコマンドを送ることにより高速描画を実現していた。これに対し、最近の PC では、マザーボードに OpenGL を含むグラフィクス機能が搭載され、転送速度が向上される傾向にある。このため、メモリ上のデバイスに対する間接描画も、直接出力に遜色のない描画速度が実現されているように感じられる。

景観データベースの検索結果を表示するための childfrm.cpp においては、多数のウィンドウを処理するために多大のリソースを消費することを避け、初期化、ペイント、終了処理で、より深い処理を行っている。即ち、HGLRC を保持するのではなく、表示用の画像をイメージとして作成した後は、画像以外のデータを解放している。

リスト１３－３：HGLRC を毎回解放する方法

```
①OnCreate の処理
     CPaintDC dc(this);
     mDC.CreateCompatibleDC(&dc);
     GLFLAG = 1; //OnPaint で GL 描画を要求
```

②OnPaint の処理

```
CPaintDC dc(this);
if(!GLFLAG){ //画像が既にある場合
  RECT rect;
  rect = dc.m_ps.rcPaint;
  dc.BitBlt(rect.left, rect.top,
       rect.right-rect.left, rect.bottom-rect.top,
       &mDC, rect.left, rect.top, SRCCOPY);
  return;
}else{ //OpenGL 描画が必要な場合
  (前処理)
  m_hDC = new HDC;
  mBM.DeleteObject();
  mBM.CreateCompatibleBitmap(&dc,m_size.cx,m_size.cy);
  mDC.SelectObject(mBM);
  *m_hDC = mDC.m_hDC;
  zet_pixel(*m_hDC);
  m_hGLRC = wglCreateContext(*m_hDC);
  wg3EnryDrawarea((void*)&m_hDC, (void*)&m_hGLRC);
  wg3AssignDrawrea((void*)&m_hDC);
  g3InitializeWindow();
  (画像の描画)
  wg3Redraw((void*)&m_hDC);

  wg3AssignDrawarea(NULL);
  wg3DeleteDrawarea(&m_hDC);
  m_hGLRC = wglGetCurrentContext();
  dc.BitBlt(0,0,m_size.cx,m_size.cy,&mDC,0,0,SRCCOPY);
  GLFLAG = 0;
  delete m_hDC;
}
```

③デストラクタの処理

```
mBM.DeleteObject();
```

１４．景観データベース関連ユーティリティ

１４－１．概要

本章では、景観事例、景観構成要素、景観材料の３種類のデータベースの検索及びデータ入力のためのユーティリティについて解説する。

データベース検索機能は、景観シミュレータと連携して動作する、独立した実行形式(.exe)として開発された。この連携は、二つの機構によって実現されている。

①景観シミュレータの配置ダイアログにおいて、景観データベースを起動し、登録されている三次元データを検索・選択する。選択されたオブジェクトが配置の操作対象となる。

②景観データベースにおける検索・選択操作の中で、候補となる三次元オブジェクトを確認する際に、景観シミュレータを起動し、オブジェクトを表示する。

この連携は、CreateProcess()関数により、相互にプロセス起動を行うと共に、その際に処理対象とするデータに関する情報を、小さなコントロールファイルを介して受け渡すことにより実現している。具体的には、上記①の終了時点で、検索操作の結果、特定のファイルが選択されている場合には、ksim/temp ディレクトリに、結果を格納する tmp001.txt というファイルが作成される。また、②の起動時点では、同じディレクトリに、tmp002.txt というファイルを作成し、この中で開くべき LSS-G ファイル名を指定すると共に、景観シミュレータ sim.exe を起動する際に、”f –db”という引数を渡して、景観データベースからの表示要求による起動であることを、sim.exe の側に通知する。

データベース入力用エディタは、３種類のデータベースを編集する際に使用する、独立した実行形式である。

現在では、データベースの多くは市販のデータベース・エンジンを用いて構築されている。しかし、景観検討のための、これらのデータベースを開発した 1993 年当時は、適切な汎用データベース・エンジンがまだ存在していなかった。このため、これらの各処理には、市販のデータベース・エンジンや既存ライブラリ等は使用せず、dbms ライブラリの関数をゼロ・ベースで開発し、これを用いてアプリケーションとデータ内容を構築した。結果的に、データも含む実行形式のセットアップや再配布等においてライセンス上の制約はない。

2001 年度に開発した、ネットワークによる事例検索機能においては、Windows NT 系のサーバー上に商用データベース・エンジンを用いることとした。しかし、このエンジンには、マイクロソフト社により再配布可能なエンジンとして提供されている MSDN を使用したため、ユーザーが独自にサーバーを構築しデータ公開する上でのライセンス制約（追加費用負担等）は生じていない。

１４－２．データ・ファイル

データベースの登録内容を具体的に記述するデータ・ファイルは、com.txt という名称のテキストファイルである。その格納場所（ディレクトリ）は環境設定ファイル(デフォルト

名 kdbms.set)の中で、リスト１に示すラベルによって定義している。

リスト１４－１：環境設定ファイルにおけるデータベースの格納ディレクトリ指定

```
FILE_PATH_JIREI    = kdb/jireitxt;        景観事例
FILE_PATH_YOUSO  = kdb/yousotxt;        景観構成要素
FILE_PATH_ZAIRYO = kdb/zaitxt;          景観材料
```

それぞれのディレクトリには、リスト２に示したファイルが置かれる。

リスト１４－２：各データベースのディレクトリに置かれるファイル

```
com.txt :          データベースの登録内容を記述したテキストファイル
com.txt.save :     入力エディタで編集した場合に作成される編集前のバックアップ
text.del :         編集操作により削除されたエントリーの記録
text.srt :         編集操作により設定された並び順（日付または名称）
```

登録項目は３種類のデータベースで異なっており、その項目構成は、dbms ライブラリの中で固定的に定義している。景観構成要素データベースの com.txt の一部を示す：

リスト１４－３：景観構成要素データベースの登録内容例（com.txt、抄）

```
C1(構成種別)    自然物.里山植物.林床植物
M1(整理番号)    292
M2(作成日)      2008 年 11 月 14 日 01 時 57 分
M3(修正日)      2008 年 11 月 14 日 01 時 57 分
M4(名称)        アズマネザサ
M12(特長)       季節によらず
M13(施工例)     みちのく
V1(画像)        アズマネザサ  m_az_s.jpg
V2(３D)         基本形状  m_az_w1.geo
V2(３D)         群生  m_azg_w1.geo
EOD

C1(構成種別)    自然物.里山植物.林床植物
M1(整理番号)    291
M2(作成日)      2008 年 11 月 14 日 01 時 57 分
M3(修正日)      2008 年 11 月 14 日 01 時 57 分
M4(名称)        カタクリ
M12(特長)       春
M13(施工例)     みちのく
V1(画像)        ｶﾀｸﾘ m_kt_s.jpg
V2(３D)         基本形状  m_kt_w1.geo
V2(３D)         パターン 30 度  m_kt_sm1.geo
V2(３D)         パターン 60 度  m_kt_sm2.geo
EOD
```

C 系列の項目は、プルダウン・メニューで選択・指定する分類項目である。M 系列の項目は、文字列型の登録項目である。V 系列の項目は、画像や三次元データ等の表示用データへの参照である。

それぞれの項目のデータ型は、初期化の過程で、dbSetTitle 関数(dbread.c)の中で、データベース種類及び登録項目番号に従い、switch 文の中で定義される。

１４－３．メニュー項目の定義ファイル

景観データベース検索のメニューは、外部テキストファイルにより定義しており、実行形式を変えることなくユーザーが変更することができる。

データベースの種類毎の定義ファイル名は、環境設定ファイルの以下のラベルによって定義している。

リスト１４－４：環境設定ファイルにおけるプルダウンメニュー定義ファイルの指定

```
CLASS_FILE_JIREI_1      （景観事例データベースのプルダウン１）
CLASS_FILE_JIREI_1      （景観事例データベースのプルダウン２）
CLASS_FILE_JIREI_1      （景観事例データベースのプルダウン３）
CLASS_FILE_YOUSO_1  （景観構成要素データベースのプルダウン１）
CLASS_FILE_YOUSO_1  （景観構成要素データベースのプルダウン２）
CLASS_FILE_YOUSO_1  （景観構成要素データベースのプルダウン３）
CLASS_FILE_ZAIRYO_1 （景観材料データベースのプルダウン１）
CLASS_FILE_ZAIRYO_1 （景観材料データベースのプルダウン２）
CLASS_FILE_ZAIRYO_1 （景観材料データベースのプルダウン３）
```

これらの９種類のファイルにより、階層的なプルダウンのメニューを構築し、検索に使用している。例えば、デフォルトのセットアップにおいては、景観構成要素データベースのプルダウン１を定義している、kdb/yousocls/youso_c1.cls ファイルの内容は以下のようになっている：

リスト１４－５：景観構成要素検索用メニュー項目の定義（youso_c1.cls）

```
1 都市
    2
    2 建物
        3 戸建住宅
        3 低層アパート
        3 マンション
        3 ビル
        3 店舗
        3 店舗併用住宅
        3 駐車場
    2 路上
    2 屋内
1 河川
    2 河道
        3 平面形状
        3 縦横断形状
        3 高水敷・河原
        3 護岸D
        3 護岸E
    2 河道内微地形
        3 州
        3 河床
        3 岩・石・砂利
    2 水面
        3 流量
        3 流れ
        3 水質
    2 河川構造物
        3 堤防
        3 護岸
        3 堰
        3 水門
    2 河川占用物
    2 植栽

1 ダム
    2 ダム本体
    2 ダム下流表面
    2 放流設備
    2 取水設備
    2 高欄、照明設備、舗装
    2 フーチング
    2 貯水池周辺
    2 管理棟

1 海岸
    2 海岸堤防・護岸
    2 消波
    2 突堤・人工岬
    2 離岸堤
    2 養浜
    2 人工リーフ

1 砂防
1 急傾斜地
1 道路
    2 視線誘導標
1 橋梁
1 自然物
    2 樹木
        3 高木
        3 低木
        3 その他
    2 亜熱帯樹木
        3 高木
        3 中木
        3 低木
        3 地被
        3 生垣
    2 里山植物
        3 樹木
        3 林床植物
    2 岩肌
        3 花崗岩
        3 安山岩
        3 凝灰岩
        3 砂岩
        3 大理石
    2 空
    2 土
    2 石
1 亜熱帯建築
    2 民家
    2 最近の家屋
    2 工作物
1 人工物
    2 素材
        3 コンクリート
        3 アスファルト
    2 部品
        3 防護柵
        3 壁高欄
        3 標識
        3 信号
        3 路面表示
        3 ポスト
        3 電話ボックス
        3 カーブミラー
        3 電柱・電線
        3 化粧型枠
        3 可変情報板
        3 トイレ
        3 バス停
        3 街灯
        3 案内板
        3 フラワーポット
        3 ベンチ
        3 歩道橋
    2 擁壁
        3 コンクリート材料
        3 石材他
    2 舗装
        3 ブロック式舗装
        3 二層構造式舗装
        3 加熱混合式舗装
        3 常温混合式舗装
        3 常温塗布式舗装
        3 その他の舗装
1 点景
    2 人
        3 男性
        3 女性
        3 少年
        3 少女
        3 カップル
        3 家族
```

```
        3 座った人
    2 乗り物
        3 車
        3 その他
    2 動物
        3 鳥
1 背景要素
    2 周辺建物
        3 商店
        3 倉庫
        3 美術館
        3 図書館
        3 体育館
        3 洋館
        3 マンション
        3 町並
    2 田園風景
        3 水田
        3 集落
1 公園
```

各行が一つのメニュー項目を示しており、メニューの階層と見出し項目文字列を定義している。これにより階層的なメニュー構成が定義される。

１４－４．初期化処理

景観データベース検索機能は、Document/View アーキテクチャで構築されている。

例えば、景観構成要素データベースにおいては、CKouDoc クラス(Koudoc.h)に、CDataBase m_db というメンバ変数が定義されている。CDataBase クラスは、dbil ライブラリの database.h で定義されている。CKouDoc クラスの構築に際して、CDataBase のメンバである DBInitialize()関数が実行され、データが読み込まれる。

この DBInitialize 関数から、更に Windows 系のビルドにおいては、m3init(データベース種別)関数が、データベースの種類(0～2)を引数として呼び出され、種類に応じた com.txt を読み込む。この実際の読み込みは、dbms ライブラリの dbMakeDataBase()関数(dbread.c)により行われる。

１４－５．各ダイアログ

（1）景観事例検索機能

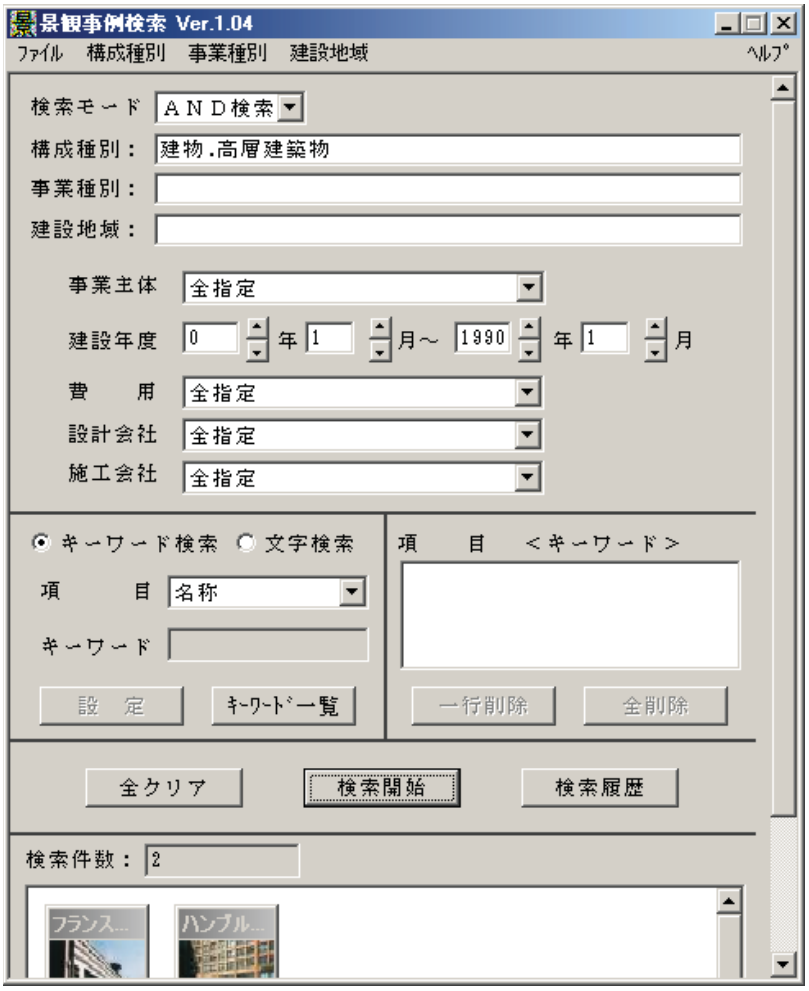


図１４－１：景観事例検索画面

リソース：IDD_YUU_FORM　ハンドラ：CyuuView(yuuvw.cpp)

実行形式は、yuu.exe、ビルドの名称は YUU である。全てのデータをローカル・マシンに置くセットアップ環境において使用する。2001 年以降は、登録データ量が大きくなったため、WEB サーバー上にデータを置く構成も利用可能である。

図１４－２：文字情報表示画面

リソース：IDD_DIALOG3　ハンドラ：CmojiDlg (mojidlg.cpp)

図１４－３：表示データ選択画面

リソース：IDD_DIALOG5　ハンドラ：CselDlg (seldlg.cpp)

図１４－４：画像表示画面

リソース：IDD_DIALOG4　ハンドラ：CImaDlg (imadlg.cpp)

図１４－５：検索履歴画面

リソース：IDD_DIALOG6　ハンドラ：CHistDlg (histdlg.cpp)

（２）景観構成要素

実行形式は、kou.exe、ビルドの名称は KOU である。個別商品ではない一般的な要素（点景、植物、標識など）を登録しており、モデリング作業の中で最も頻繁に用いられる。通常は、景観シミュレータの配置ダイアログから起動し、選択した対象物を配置する。

個別の景観検討に際して作成された各種オブジェクトの内、別の現場に再利用可能なものを追加登録してきたため、現在まで継続的に増え続けている。

図１４－６：景観構成要素検索画面

リソース：IDD_KOU_FORM　ハンドラ：CKouView(kouvw.cpp)

他のダイアログは、景観事例とほぼ同様であるため、省略する。

（３）景観材料

実行形式は、zai.exe、ビルドの名称は ZAI である。システム開発に着手した 1993 年当時、景観に対する関心の高まりから、景観に配慮した各種材料（コンクリート仕上げの化粧や、ストリート・ファニチャ等）が建材メーカー等から発売されていた。当初の構想として、これらの景観材料を業者側の負担においてデータベースに登録し、景観検討作業の中で簡便に利用するという考え方があり、土木研究所の下に検討委員会を設けて具体化が検討された。しかし、その後インターネットの急速な普及があり、各社がホームページから情報公開することが可能となったため、進展しなかった。登録項目には、商品としての情報も含まれている。当時、材料メーカー等の協力を頂き、試験的にいくつかの景観材料の製品を登録したため、現在までアプリケーションの中に継承しているが、その後インターネットが普及し、各社がそれぞれのＷＥＢサイトから製品情報を公開するようになり、検索も容易になったため、本アプリケーションの今後の発展性は余り期待できない。

操作環境の観点からは、このような形状を有するオブジェクトは景観構成要素データベースに統合し、逆に形状が一定ではない加工可能な材料（土、木、石など）のテクスチャや、パラメトリックな関数として形状が決定されるオブジェクトをデータベース化し、編

集操作と関連づけることが有益と考える。

図１４－７：景観材料検索画面

リソース：IDD_ZAI_FORM　ハンドラ：CZaiView(yuuvw.cpp)

メイン画面上の操作に応じて表示される、この他のダイアログは、景観事例検索と同様であるため省略する。

（4）データベース入力エディタ

実行形式は editor.exe、ビルド名は defdbeditor である。上記３種類のデータベースをローカルに登録するためのツールである。

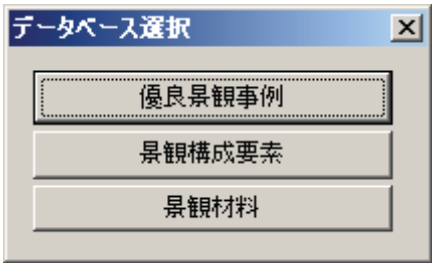


図１４－８：データベース種類選択画面

リソース：IDD_PILIH、ハンドラ：CPilih (pilih.cpp)

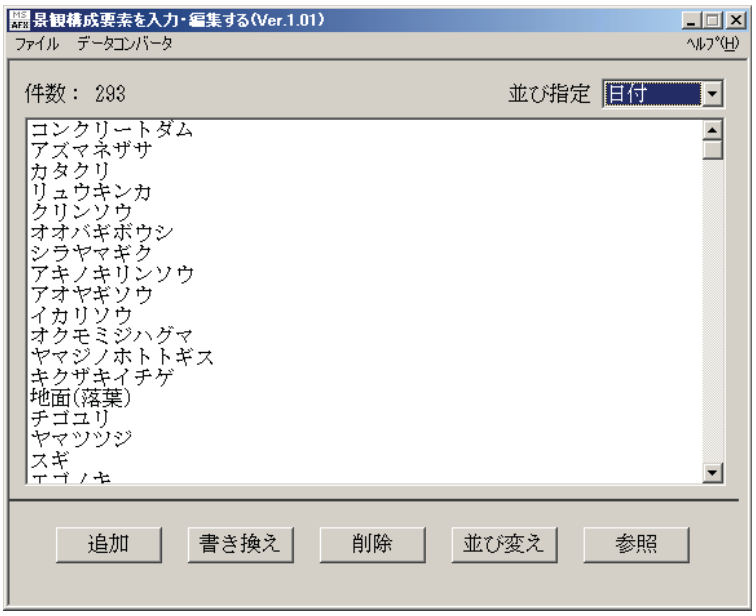


図１４－９：メイン画面（景観構成要素データベースを例示）

リソース：IDD_MAIN_FORM　ハンドラ：CMainView(mainview.cpp)

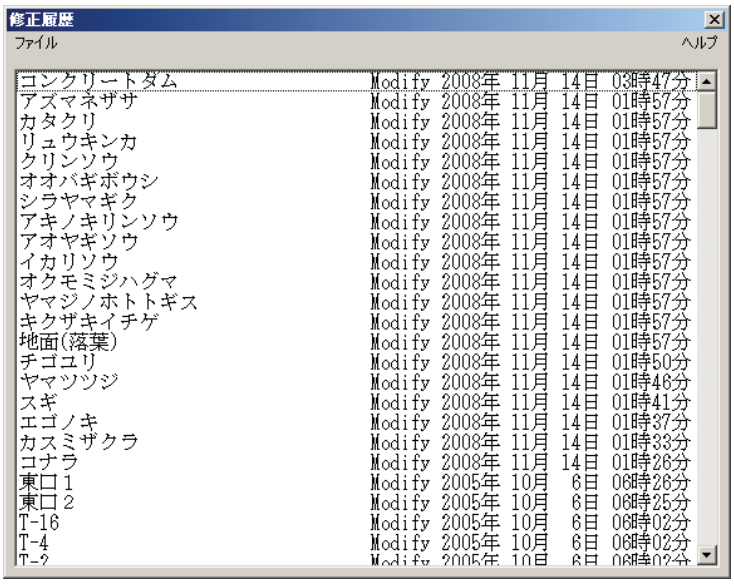


図１４－１０：修正履歴表示画面

リソース：IDD_DLG_HIST、ハンドラ：CHistDlg (histdlg.cpp)

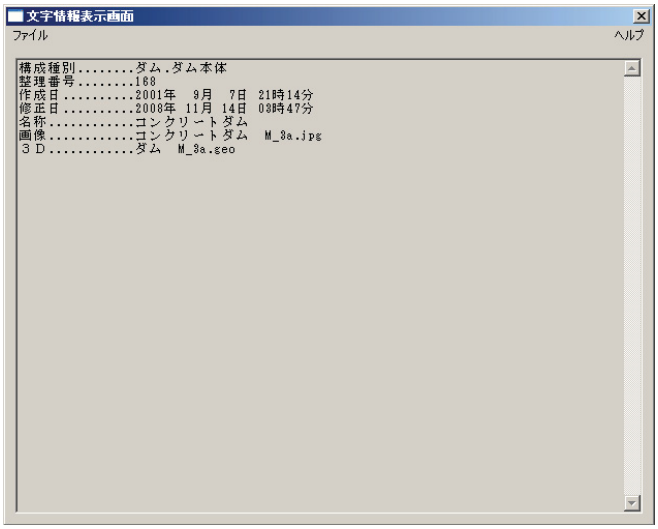


図１４－１１：文字情報表示画面

リソース：IDD_DLG_MOJI、ハンドラ： CMojiDlg(mojidlg.cpp)

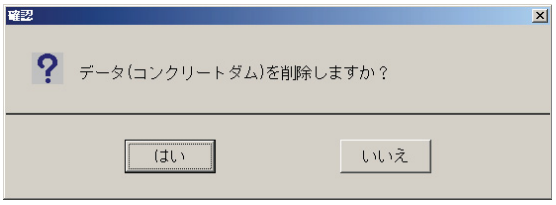


図１４－１２：削除確認画面

リソース：IDD_DLG_QUES、ハンドラ：CQuesDlg (quesdlg.cpp)

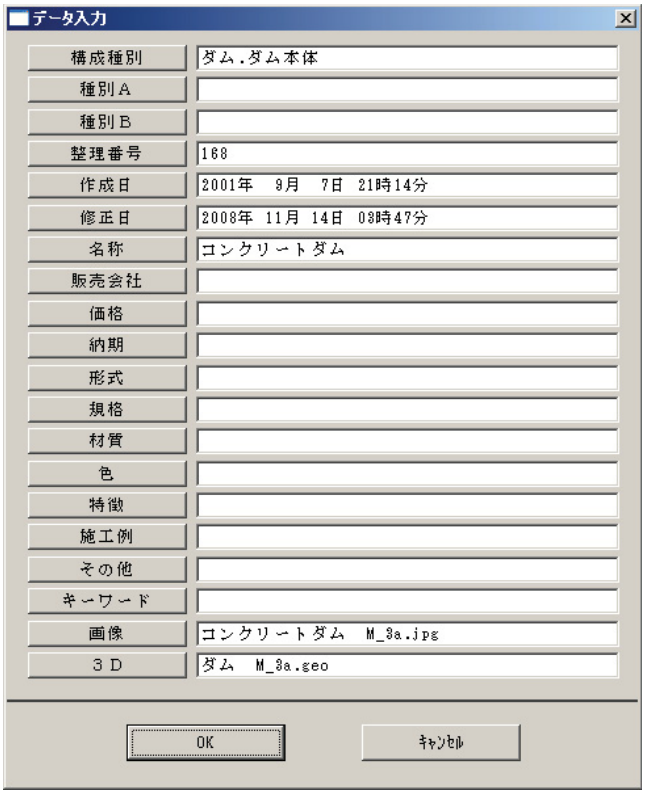


図１４－１３：データ入力画面

景観事例　　：リソース：IDD_DLG_INPUT1、ハンドラ：CInpDlg (inpdlg.cpp)

景観構成要素：リソース：IDD_DLG_INPUT2、ハンドラ：CInpDlg (inpdlg.cpp)

景観材料　　：リソース：IDD_DLG_INPUT3、ハンドラ：CInpDlg (inpdlg.cpp)

その他形式　：リソース：IDD_DEFCSV_FORM、ハンドラ（defcsv.cpp)

景観事例、景観構成要素、景観材料の３種類のダイアログの様式を記述するリソースは、CMainView::OnInitialUpdate 関数(mainview.cpp)で選択しており、ハンドラは共通である。

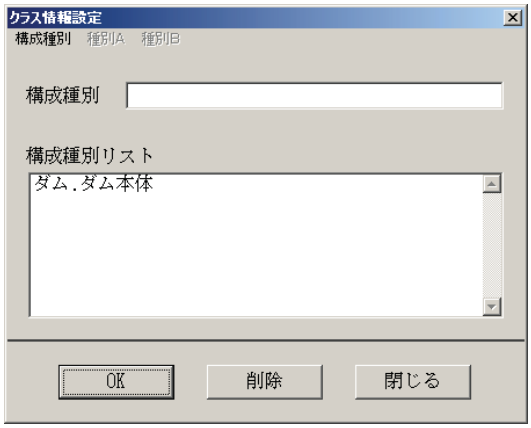


図１４－１４：クラス情報設定画面

リソース：IDD_DLG_CLASS、ハンドラ：CClassDlg(classdlg.cpp)

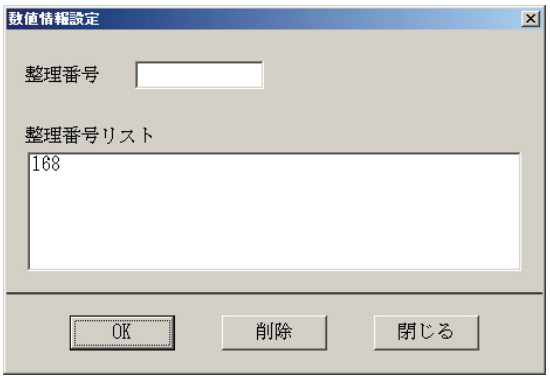


図１４－１５：数値情報設定画面

リソース：IDD_DLG_LONG、ハンドラ：CLongDlg(longdlg.cpp)

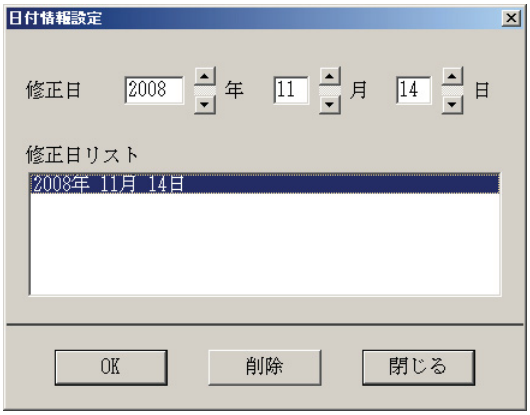


図１４－１６：日付情報設定画面

リソース：IDD_DLG_TIME、ハンドラ：CTimeDlg(timedlg.cpp)

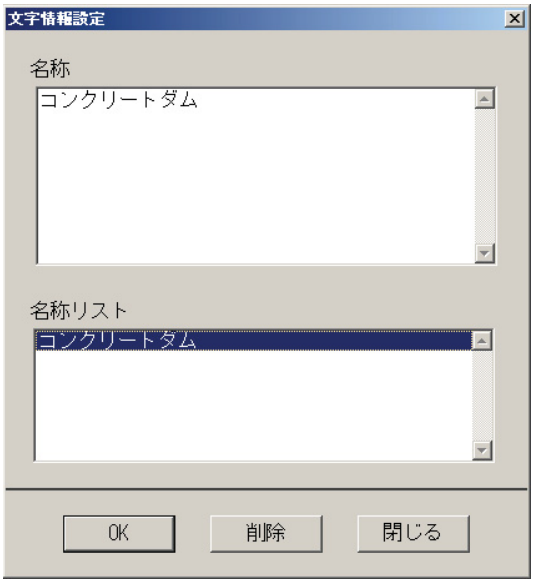


図１４－１７：文字情報設定画面

リソース：IDD_DLG_STRING、ハンドラ：CStringDlg(stringdlg.cpp)

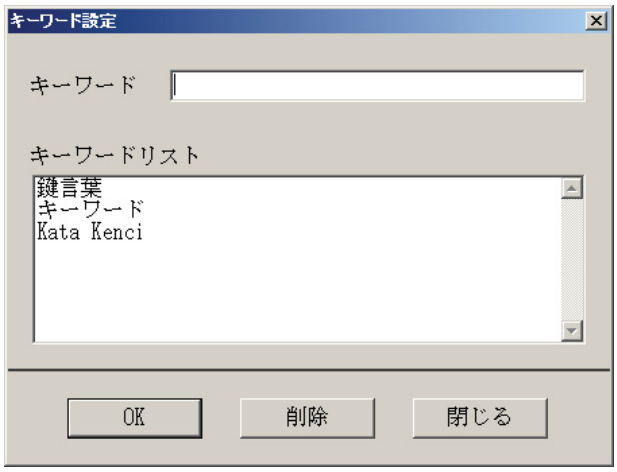


図１４－１８：キーワード設定画面

リソース：IDD_DLG_KEYWD、ハンドラ：CKeywdDlg (keywd.cpp)

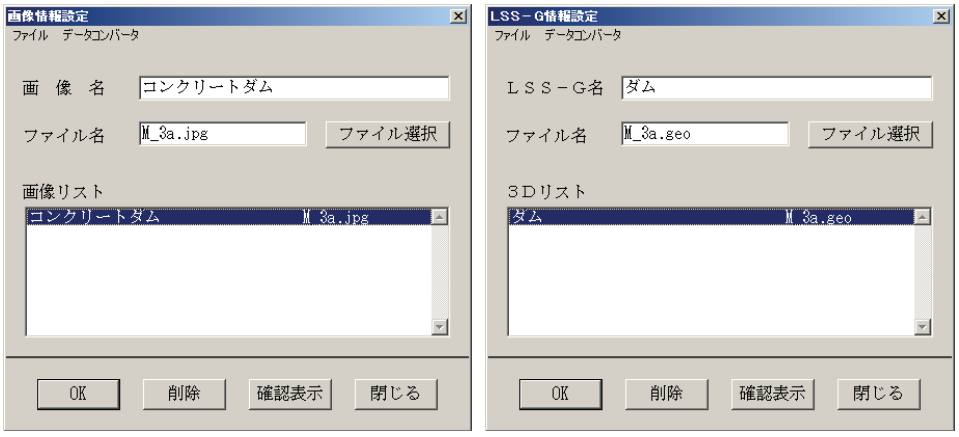


図１４－１９：画像設定画面(左)、３D ファイル設定画面（右）

リソース：IDD_DLG_GAZOU、ハンドラ：CGazouDlg(gazoudlg.cpp)

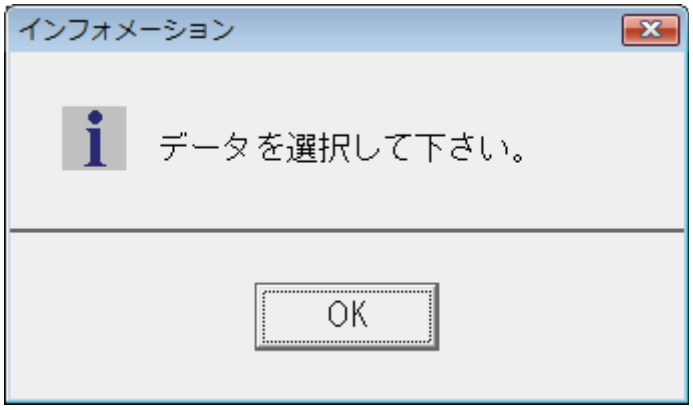


図１４－２０：情報提供画面

リソース：IDD_DLG_INFO、ハンドラ：CInfoDlg(infodlg.cpp)

１４－６．ネットワーク用データベース入力エディタ

データベースにネットワークでアクセスする場合には、サーバー側に MSDE をエンジンとするデータベースを置き、これに WEB アクセスする。その場合には、上記の３種類のデータベースの仕様に限定せずに、任意の構成のデータベースを置けるようにするために、データベースの構成を定義するファイル def.csv を用いている。

WEB 経由でデータを登録する場合には、小規模な帳票構成の場合には、入力用のページ(FORM)を用意すれば十分であるが、大きな帳票構成の場合には、クライアント側で途中まで入力したデータをローカルに保存できる方が便利である。そこで、サーバー側でデータベースを定義しているものと同じ定義ファイルを用いた帳票構成を入力するためのエディタ editorf.exe を用意した。

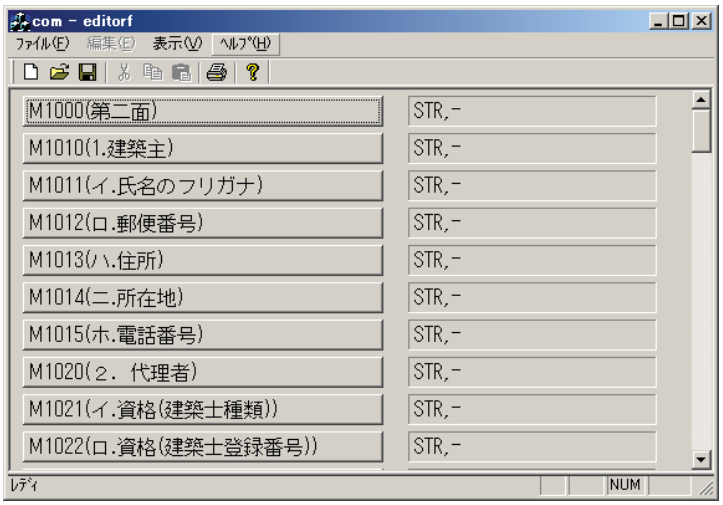


図１４－２１：editorf.exe メイン画面

リソース：IDD_EDITORF_FORM　ハンドラ：CEditorfView(editorfview.cpp)

なお、個々の項目を入力するための、データ型に対応した入力ダイアログは、上記の入力用エディタと同様であるため、省略する。

上記のメイン画面の構成を定義している def.csv ファイルの内容を下記に示す。

リスト１４－６：データベース定義ファイル def.csv の例

```
A1,優良景観材料,JIREI,
A2,巡回時間を指定,5,
A3,審査委員数,2,
A4,審査期間日数,5,
A5,督促開始残日数,2,
A6,仮公開用コンテンツディレクトリー,s:¥仮登録¥JIREI,http://210.146.71.74/KARIKDB/JIREI
A7,本公開用コンテンツディレクトリー,s:¥本登録¥JIREI,http://210.146.71.74/KDB/JIREI
A9,MailFrom,keikan2@nilim.go.jp,
A10,,s:¥優良景観事例¥tehaishi¥IraiMailTemplate.txt,,
A11,,s:¥優良景観事例¥tehaishi¥TokusokuMailTemplate.txt,,
A12,,s:¥優良景観事例¥tehaishi¥GoukakuMailTemplate.txt,,
A13,,s:¥優良景観事例¥tehaishi¥FugoukakuMailTemplate.txt,,
A14,,s:¥優良景観事例¥tehaishi¥SaishinsaMailTemplate.txt,,
A22,ＤＢサーバーアドレス,(local),,
```

A24,Lha32 ファイル名,s:¥優良景観事例¥tehaishi¥Lha.exe,,
A25,審査お断り期間日数,2,
A26,審査お断り ASP ファイル名,http://210.146.71.74/jirei/Inc_sinsa/tTouroku_okotowari.asp,
A27,審査結果 ASP ファイル名,http://210.146.71.74/jirei/Inc_sinsa/tTouroku_sinsa.asp,
A28,cmd.exe ファイル名,s:¥優良景観事例¥tehaishi¥やれ.exe,,
A31,審査記録テンプレートファイル,s:¥優良景観事例¥HP_SinsaAspTemplate.htm,,
A32,審査用画像ファイル一覧テンプレートファイル,s:¥優良景観事例¥HP_SinsaAspFileTemplate.htm,,
A33,com.txt 復元ページテンプレートファイル,s:¥優良景観事例¥HP_Outcomtxt.htm,,
A34,送信結果ページテンプレートファイル,s:¥優良景観事例¥HP_KekkaTemplate.htm,,

A98,デバッグ情報連絡先,debug@bug.or.jp,,
A99,処理記録を残すとき 1 を指定する,1,,

C1,構成種別,CLASS,STR,－
C2,事業種別,CLASS,STR,－
C3,建設地域,CLASS,STR,－

M1000,第二面,STR,STR,-
M1010,１．建築主,STR,STR,-
M1011,イ.氏名のフリガナ,STR,STR,-
M1012,ロ.郵便番号,STR,STR,-
M1013,ハ.住所,STR,STR,-
M1014,ニ.所在地,STR,STR,-
M1015,ホ.電話番号,STR,STR,-

M1020,２．代理者,STR,STR,-
M1021,イ.資格(建築士種類),STR,STR,-
M1022,ロ.資格(建築士登録番号),STR,STR,-
M1023,ハ.氏名,STR,STR,-
M1024,ニ.建築士事務所名,STR,STR,-
M1025,ニ.事務所登録番号,LONG,LONG,-
M1026,ホ.郵便番号,STR,STR,-
M1027,ヘ.所在地,STR,STR,-
M1028,ト.電話番号,STR,STR,-
M1030,３．設計者,STR,STR,-
M1040,４．建築設備に関し意見を聴いた者,STR,STR,-
M1050,５．工事監理者,STR,STR,-
M1060,６．工事施工者,STR,STR,-
M1070,７．備考,STR,STR,-

M2000,第三面,STR,STR,-
M2010,１．地名地番,STR,STR,-
M2020,２．都市計画区域の内外,STR,STR,-
M2030,３．防火地域,STR,STR,-
M2040,４．その他の区域、地域、地区、街区,STR,STR,-
M2050,５．道路,STR,STR,-
M2051,イ.幅員,LONG,LONG,M
M2052,ロ.敷地と接している部分の長さ,LONG,LONG,M
M2060,６．敷地面積,STR,STR,-
M2070,７．主要用途,STR,STR,-
M2080,８．工事種別,STR,STR,-
M2090,９．建築面積,LONG,LONG,平米
M2100,１０．延べ面積,LONG,LONG,平米
M2110,１１．建築物の数,STR,STR,-
M2120,１２．工事着手予定年月日,TIME,DATE,-
M2130,１３．工事完了予定年月日,TIME,DATE,-
M2140,１４．その他必要な事項,STR,STR,-

M3000,第四面（複数）,STR,STR,-
M3010,１．番号,LONG,LONG,-
M3020,２．用途,STR,STR,-
M3030,３．工事種別,STR,STR,-
M3040,４．構造,STR,STR,-

```
M3050,５．耐火建築物,STR,STR,-
M3060,６．屋根,STR,STR,-
M3070,７．外壁,STR,STR,-
M3080,８．軒裏,STR,STR,-
M3090,９．居室の床の高さ,LONG,LONG,cm
M3100,１０．階数,STR,STR,-
M3101,イ.地階を除く階数,LONG,LONG,-
M3102,ロ.地階の階数,LONG,LONG,-
M3103,ハ.昇降機塔等の階の数,LONG,LONG,-
M3110,１１．高さ,LONG,LONG,m
M3111,イ.最高の高さ,LONG,LONG,m
M3112,ロ.最高の軒の高さ,LONG,LONG,m
M3120,１２．床面積,STR,STR,-
M3121,イ.階別,LONG,LONG,-
M3122,ロ.合計,LONG,LONG,-
M3130,１３．便所の種類,STR,STR,-
M3140,１４．建築設備の種類,STR,STR,-
M3150,１５．確認の特例,STR,STR,-
M3151,イ.建築基準法第６条の２第１項の規定による確認の適用の有無,STR,STR,-
M3152,ロ.適用があるときは、建築基準法施行令第 13 条の２各号に掲げる建築物の区分,STR,STR,-
M3153,ハ.建築基準法施行令第 13 条の２第一号又は第二号に掲げる住宅に該当するときは、当該住宅にかかる型式指定
番号,STR,STR,-
M3160,その他必要な事項,STR,STR,-
M3170,備考,STR,STR,-

M4000,第五面　建築物の階別概要,STR,STR,-
M4010,番号,LONG,LONG,-
M4020,階,STR,STR,-
M4030,柱の小径,LONG,LONG,mm
M4040,横架材間の垂直距離,LONG,LONG,mm
M4050,階の高さ,LONG,LONG,mm
M4060,居室の天井の高さ,LONG,LONG,mm
M4070,用途別床面積,STR,STR,-
M4071,用途の区分,STR,STR,-
M4072,具体的な用途の名称,STR,STR,-
M4073,床面積,LONG,LONG,平米
M4080,その他必要な事項,STR,STR,-

V1,図面,IMAGE,STR,－
V2,３Ｄ,LSSS,STR,－
```

editor.exe は、３種類のデータベースのそれぞれに関して、登録されている全てのデータを一覧表示した上で、それを構成するレコードを編集するのに対して、この editorf.exe によって入力し、ファイル保存したデータ・ファイル com.txt は、一つのレコードしか持たない。ゆえに、全てのレコードを一覧表示する画面はもたない。

１５．インストーラ

１５－１．概要

本章では、景観シミュレータ一式をセットアップするインストーラの動作、及びセットアップ一式を構築するための手順について解説する。

セットアップには、VisualC++開発環境に付属している簡単な InstallShield の機能を使用している。セットアップに際して、レジストリの設定などの高度な処理を行っていないため、Ver.2.03 のセットアップに使用した Ver.2.00 を用いて、必要な部分を修正するのみで現在に至っている。従って、素朴な機能を用いているのみである。

１５－２．セットアップの構成と動作

セットアップには、CD-ROM に必要なファイルを置き、ここから setup.exe を起動してセットアップを行う方法と、関係するファイル一式を自己解凍形式のファイル sim209.exe に束ねて WEB 配信し、クライアント側で適当なディレクトリ上に解凍した上でセットアップ作業を行う方法を用意した。

まず、CD-ROM で配布する場合、ディスク上には、リスト１に示したファイルを提供している。

リスト１５－１：セットアップのファイル構成

（標準のファイル）	
Setup.exe	セットアップを行う実行形式
Inst32I.ex	セットアップの実行に使用されるバイナリ
Setup.dll	同上
Uninst.exe	アンインストーラ
_isdel.exe	アンインストール
（景観シミュレーション・システムに固有のファイル）	
Setup.bmp	セットアップに際して表示するロゴ（イメージ）
Setup.ins	セットアップ中に表示されるテキスト等を格納したファイル
Setup.ini	アプリケーション名や必要とするディスクスペース等の定義ファイル
Setup.iss	システムのバージョンやライセンスを記述
（バージョンに固有のファイル）	
Data.z	セットアップするファイルを、ディレクトリ付きで圧縮したファイル
Setup.pkg	セットアップするファイルの一覧
_setup.lib	セットアップの手順などを定義したファイル
DISK1.ID	ID

バージョン・アップやデバッグにより更新される主な部分は、data.z と、これに関連した Setup.pkg である。

Setup.exe を実行すると、標準的なインストーラの画面が表示され、セットアップ先のディレクトリ指定など、必要なパラメータや選択肢を確認しながらセットアップ作業を進める。

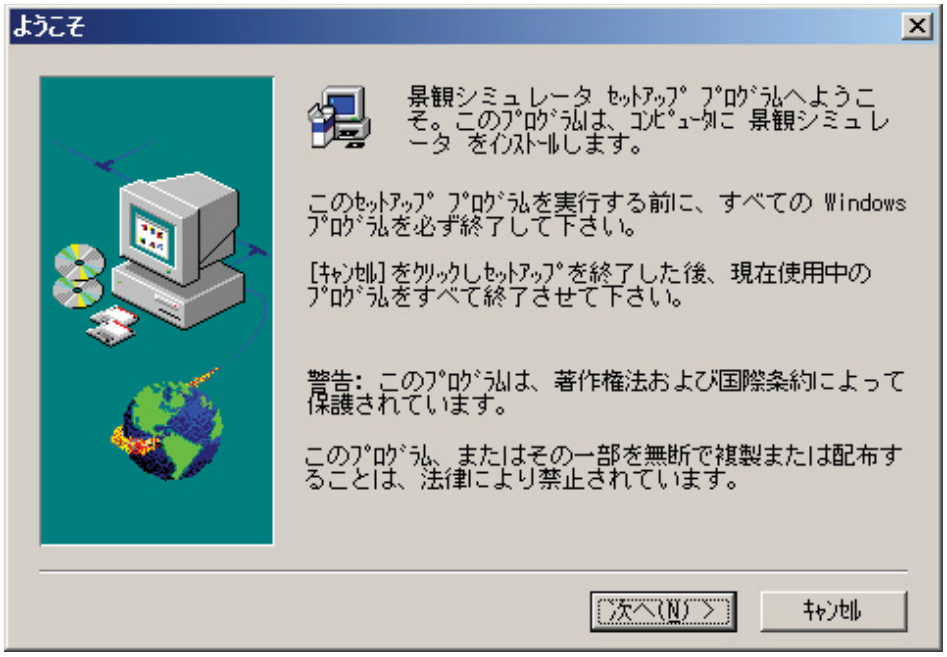


図１５－１：セットアップ画面１

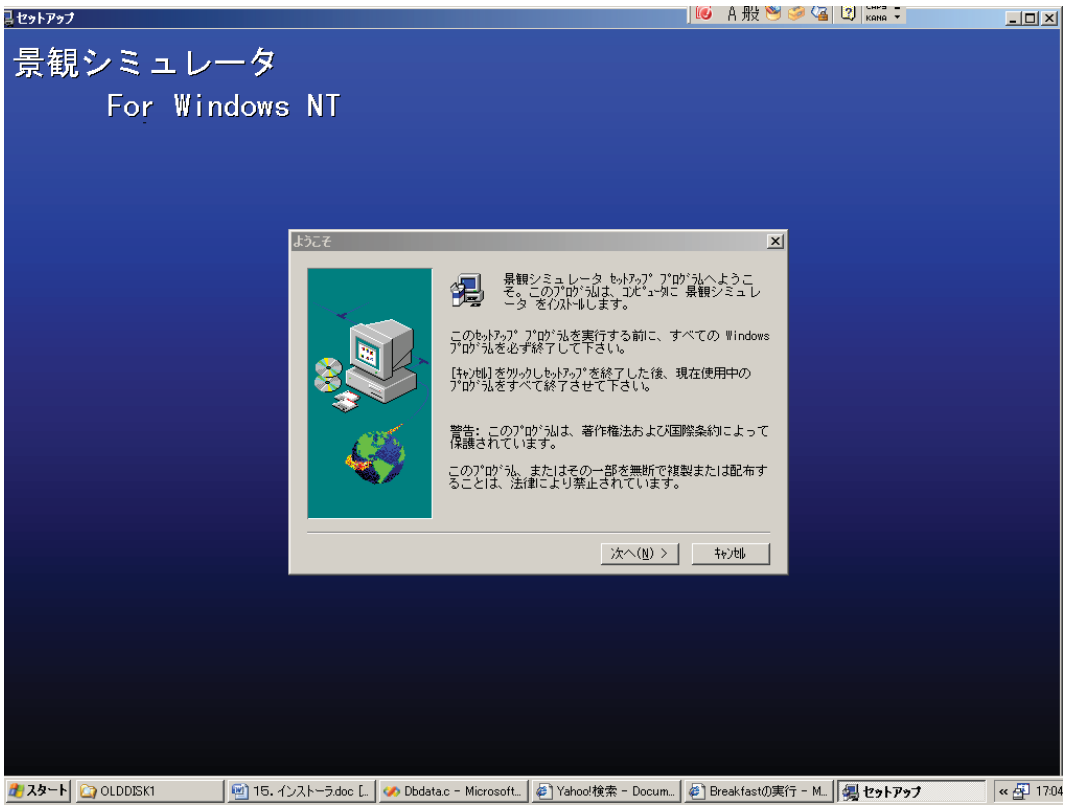


図１５－２：セットアップ画面２

セットアップがインストール先のシステムに適応して行う重要な処理は、環境変数 KSIM_ENV の設定(kdbms.set の所在をフルパスで指定する)と、kdbms.set の中の HOME_PATH エントリーを、ユーザーが指定したセットアップ先に書き換えることである。手動で必要なファイルをコピーした場合であっても、上記の２点の作業を行うことにより、インストーラを用いてセットアップを行った場合と同じ結果を得ることができる。

１５－３．セットアップの構築手順

新たなバージョンのセットアップを構築する場合、簡単には、ホームディレクトリ以下の各ファイルを用意した上で、以下の内容を有する BUILD.BAT を実行する。

リスト１５－２：セットアップ構築処理の全体を記述したバッチ・ファイル(build.bat)

```
rem @echo off
rem
rem ....Build components...
rem
setup¥compile setup¥setup.rul -g
copy setup¥setup.ins disk1
copy setup¥setup.ini disk1

if NOT "%1"=="" goto nopack

rem ======================================
rem  新規にアーカイブファイルを作成する
rem ======================================
del data.z

icomp data¥*.* data.z -i


:nopack

copy disk1.id disk1
cd setup
del setup.pkg
packlist setup.lst
copy setup.pkg ..¥disk1
cd ..
copy setup¥_setup.lib disk1

rem ====================================================
rem  セットアップ起動時に表示されるビットマップをコピー
rem ====================================================
copy setup¥setup.bmp                    disk1

rem ====================================================
rem  タイトルとビルボードのビットマップを圧縮コピーする
rem ====================================================
icomp setup¥title.bmp                                   disk1¥_setup.lib
icomp setup¥bbr*.bmp                                            disk1¥_setup.lib


move data.Z disk1
```

このバッチ・ファイルが実行している主な処理は、

①セットアップの中で行う処理を定義した setup.rul をコンパイルし、setup.exe の動作を定義する setup.ins 等のファイルを作成すること。

②セットアップするファイル（各種実行形式、データファイルを含むディレクトリ構成）を圧縮し data.z を作成すること。
③表示する画像ファイル(title.bmp, bbr.bmp)を圧縮し、_setup.lib とすること
および、結果のファイルのコピー、移動等の補助的な処理である。

図１５－３：title.bmp

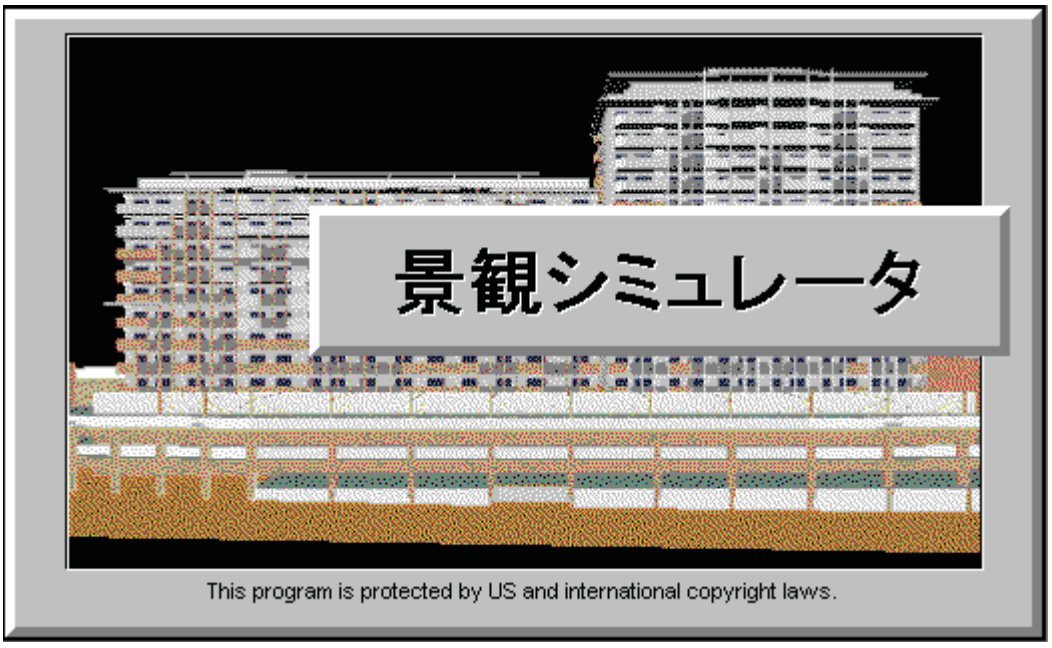


図１５－４：bbr.bmp, setup.bmp

セットアップ中に行われる処理を定義した、setup.rul は以下の内容を含んでいる。

リスト１５－３：setup.rul

```
/*---------------------------------------------------------------------------*¥
 *
 * IIIIIII SSSSSS
 *    II SS          InstallShield (R)
 *    II SSSSSS      (c) 1990-1996, Stirling Technologies, Inc.
 *    II    SS           All Rights Reserved.
 * IIIIIII SSSSSS
 *
 *
 * This code is intended as a supplement to InstallShield documentation
 * and is provided AS IS.
 *
 *
 *    File Name:   SETUP.RUL
 *
 * Description:   InstallShield
 *         32-bit Template Two script.
 *
 *    Author:   InstallShield Corporation      Date:   1-10-96
```

```
  *
  *      Comments:   This template script performs a basic installation to a
  *          Windows 95 or Windows NT platform.   The installation
  *          includes components: Application Program Files, Sample and
  *          Template Files, Online Help Files, and Multimedia Tutorial
  *          Files.   With minor modifications, this template can be
  *          adapted to create new, customized installations.
  *
¥*-------------------------------------------------------------------------*/

// Constant declarations.
#define APP_NAME     "景観 シミュレータ"
#define PROGRAM_FOLDER_NAME   "景観 シミュレータ"
//#define APPBASE_PATH     "¥¥Program Files¥¥keikan¥¥"
#define APPBASE_PATH     "¥¥@keikan"
#define EXTRA_BLOCKSPACE   200000

#define COMPANY_NAME     "NILIM"
#define PRODUCT_NAME     "景観シミュレータ"
#define PRODUCT_VERSION     "2.09"
#define PRODUCT_KEY          "SIM.exe;貿易.exe;都市開発.exe"
#define DEINSTALL_KEY     "景観シミュレータ"
#define UNINSTALL_NAME     "景観シミュレータ"

// File component constant declarations.
#define ITEM_PROGRAMFILES   "プログラム　ファイル"
#define ITEM_ENVFILES     "環境設定　ファイル"
#define ITEM_DLLFILES     "ＤＬＬ　ファイル"

#define STR_DEFTAB       "                "

declare

#include "sddialog.h" // Include script dialog definitions.

   // Global variable declarations.
   STRING svFolder, szMsg, szFileSet, szTitle;
   STRING svInstallDir;
   STRING svTarget, szProgram, szTemp, svName, svCompany, szComponentList;
   STRING svUninstLogFile, szAppPath, szRegKey, szAppSharedDir;
   BOOL bSpaceOk, bWinNT, bWin32S, bIsShellExplorer;
   NUMBER nResult, nType;
   NUMBER nRet;
   LIST listInfo;
   NUMBER nReturn;
   NUMBER nvLineNumber;
   STRING svLine;
   LIST listTopics; //20000406 DR.H.K.
   LIST listDetails; //20000406 DR.H.K.
   // Function declarations.
   prototype CheckRequirements();
   prototype ConstructInfoList( LIST );
   prototype CreateRegDBEntries();
   prototype DetermineComponentInfo( STRING );
   prototype EnoughSpace( STRING, STRING, NUMBER );
   prototype PerformFileTransfer( STRING );
   prototype SetupFileTransfer( STRING, STRING );
   prototype SetupFinish();
   prototype SetupScreen();

program

StartHere:
   //MessageBox("インストーラのデバッグ中",INFORMATION); //20000406 DR.H.K.
   Disable( BACKGROUND );

   // Set installation info., which is required for registry entries.
```

```
    InstallationInfo( COMPANY_NAME, PRODUCT_NAME, PRODUCT_VERSION, PRODUCT_KEY );

    // Set up the installation screen.
    SetupScreen();
    Enable( DIALOGCACHE );


     // Create a Welcome dialog.
     WelcomeDlg:
    Disable( BACKBUTTON );
    SdProductName( PRODUCT_NAME );
//    SdWelcome( //20000407 DR.H.K. 交換
    listTopics = ListCreate( STRINGLIST );
    listDetails = ListCreate( STRINGLIST );
    ListAddString( listTopics,
    "このプログラムは、" , AFTER );
    ListAddString( listDetails,
    "コンピュータに標記システムをインストールします。このセットアッププログラムを実行する前に、全ての
Windows プログラムを必ず終了して下さい。"
    , AFTER );
//    ListAddString( listTopics,
//    "[キャンセル]で抜けて", AFTER );
//    ListAddString( listDetails,
//    "現在使用中のプログラムをすべて終了させて下さい。", AFTER );
    ListAddString( listTopics,
    "          ◎ 一度[キャンセル]で抜けて準備して下さい。", AFTER );
    ListAddString( listDetails,
     "警告：このプログラムは、フリーウェアですが、"+
       "国土交通省が著作権を保持しており、"+
       "著作権法および国際条約によって保護されている公共財です。", AFTER);
    ListAddString( listTopics,
       "このプログラム、またはその一部を", AFTER );
    ListAddString( listDetails,
       "無断で翻訳、引用、複製または配布しても構いませんが、"+
       "何らの価値も付加しないデッドコピーを"+
       "メディア代以上の価格で販売することは制限しています。", AFTER );
    SdDisplayTopics("ようこそ",
    "国土交通省版・ネットワーク景観シミュレータ 2.05 のセットアップ"+
    "プログラムへようこそ。",
    listTopics,   listDetails, 0 );
    Enable( BACKBUTTON );

//インストール先のディレクトリを選択する


SelectInstallFolder:
//    svFolder = "C:" ^ APPBASE_PATH;
    svFolder = "C:" ^ APPBASE_PATH;
    nReturn = SdAskDestPath(PRODUCT_NAME + "のインストール先の選択",
              "セットアップは次のディレクトリに景観シミュレータを" +
              "インストールします。¥n¥n" +
              "このディレクトリへのインストールは［次へ］をクリック。¥n¥n" +
              "他のディレクトリへのインストールは［参照］をクリックしディレクトリを選択。¥n¥n" +
              "景観シミュレータをインストールしない場合は［キャンセル］をクリックして終了。",
              svFolder,
              STYLE_NORMAL);
    if( nReturn = NEXT ) then
      if( ExistsDir( svFolder ) = EXISTS ) then
//        SprintfBox( INFORMATION,
//              "すでに同名のディレクトリが存在します",
//              "ディレクトリ名 '%s' はすでに存在します¥n" +
//              "このディレクトリでよろしければＯＫを押してください¥n",
//              svFolder);
      else
        nReturn = SdConfirmNewDir( "ディレクトリを作成します(デバッグ中)",
                  svFolder,
```

```
                STYLE_NORMAL);
      if( nReturn < 0 ) then
       SprintfBox( SEVERE, "エラー",
           "ディレクトリ '%s' の作成ができませんでした¥n" +
           "インストールは未完了です", svFolder);
       exit;
      elseif( nReturn = NO ) then
       MessageBox( "ディレクトリは作成しませんでした¥n" +
           "インストールは未完了です", SEVERE);
       exit;
      endif;
     endif;
     svInstallDir = svFolder;
   else
      goto WelcomeDlg;
   endif;


    RegisterUserName:
#if 0
   nResult = SdRegisterUser( "", "", svName, svCompany );
   if ( nResult = BACK ) then goto WelcomeDlg; endif;
#endif
   // Test target system for proper configuration.
   CheckRequirements();
   // win32s では動作不可なのでインストールしない
   if ( bWin32S ) then
       MessageBox( "Windows NT 以外のＯＳは現在サポートしていません", SEVERE );
       exit;
       //svTarget = TARGETDISK ^ APPBASE_PATH_WIN32S;
   else
       svTarget = svFolder;
   endif;
   szAppSharedDir = svTarget ^ "System";
   szComponentList = "FileComponents";

   // Get component information.
   DetermineComponentInfo( szComponentList );


    GetTargetLocAndSelection:
#if 0
   // Get type of installation (Typical, Compact, or Custom)
   // インストールのスタイルを選択する（標準、コンパクト、カスタム）
   nType = SdSetupType( "", "",
           svTarget,
           STYLE_NORMAL );

   switch (nType)
   case TYPICAL:
      ComponentSelectItem( szComponentList, ITEM_PROGRAMFILES, TRUE );
      ComponentSelectItem( szComponentList, ITEM_ENVFILES, TRUE );
      ComponentSelectItem( szComponentList, ITEM_DLLFILES, TRUE );
   case COMPACT:
      ComponentSelectItem( szComponentList, ITEM_PROGRAMFILES, TRUE );
      ComponentSelectItem( szComponentList, ITEM_ENVFILES, FALSE );
      ComponentSelectItem( szComponentList, ITEM_DLLFILES, FALSE );
   case CUSTOM:
      nResult = SdComponentDialog( "", "",
                svTarget,
                szComponentList );
      if ( nResult = BACK ) then
      goto GetTargetLocAndSelection;
      endif;
   case BACK:
      goto RegisterUserName;
   default:
```

```
      MessageBox( "SetupType - 不可能な選択肢です。", SEVERE );
      exit;
   endswitch;
#else
     ComponentSelectItem( szComponentList, ITEM_PROGRAMFILES, TRUE );
     ComponentSelectItem( szComponentList, ITEM_ENVFILES, TRUE );
     ComponentSelectItem( szComponentList, ITEM_DLLFILES, TRUE );
#endif

   szAppSharedDir = svTarget ^ "System";

   // Perform space check of target drive.
   if (EnoughSpace( svTarget,
           szComponentList,
           EXTRA_BLOCKSPACE ) = FALSE) goto GetTargetLocAndSelection;


  GetProgramFolderInfo:
   svFolder = PROGRAM_FOLDER_NAME;
   nResult = SdSelectFolder( "", "", svFolder );
   if ( nResult = BACK ) then
#if 0
     goto GetTargetLocAndSelection;
#else
     goto SelectInstallFolder;
#endif
   endif;


  ConfirmCopy:
   // Show SdStartCopy dialog to confirm file transfer operation.
   listInfo = ListCreate( STRINGLIST );
   ConstructInfoList( listInfo );
   szMsg = "セットアップには" + PRODUCT_NAME + "・ファイルのコピーを開始するに十分な情報があります。" +
      "設定を確認または変更する場合「戻る」をクリックします。現在の設定" +
      "のまま実行する場合は「次へ」をクリックしてファイルのコピーを開始します。";
   if ( SdStartCopy( "", szMsg, listInfo ) = BACK ) then
       ListDestroy( listInfo );
       goto GetProgramFolderInfo;
   endif;
   ListDestroy( listInfo );


    SetupRegAndUninstall:
   // Prepare InstallShield to record deinstallation information.
   DeinstallStart( svTarget, svUninstLogFile, DEINSTALL_KEY, 0 );
   RegDBSetItem( REGDB_UNINSTALL_NAME, UNINSTALL_NAME );

   // Set the App Paths key for the main program.
   szAppPath = svTarget ^ "ksim¥¥bin" + ";" + szAppSharedDir;
   RegDBSetItem( REGDB_APPPATH, szAppPath );
   szProgram = svTarget ^ "ksim¥¥bin¥¥sim.exe";
   RegDBSetItem( REGDB_APPPATH_DEFAULT, szProgram );


    SetupAndDecompFiles:
   szFileSet = "General";
   SetupFileTransfer( szComponentList, szFileSet );
   //MessageBox( "SetupFileTransfer : complete" , INFORMATION );
   // Set up progress indicator and information gauge.
   Enable( STATUS );
   Disable( DIALOGCACHE );
   //MessageBox( "Enable Progress Indicator : complete" , INFORMATION );
   // Transfer files to the target system.
   PerformFileTransfer( szFileSet );
   //MessageBox( "PerformFileTransfer : complete" , INFORMATION );
```

```
  SetRegistryEntries:
        SetStatusWindow( 92, "レジストリの設定中...." );
  CreateRegDBEntries( );

  // Set up progress indicator and information gauge.
  //Delay( 2 );
  Delay( 1 );
  //MessageBox( "ゲージが９０％で止っていても、気にする必要なし",INFORMATION); //20000406 DR.H.K.
  //Disable( STATUS );


   // Create program folder and icons.
   CreateProgramIcons:
  SetStatusWindow( 95, "グループとアイコンを作成中...." );

  // NT でシェルがエクスプローラでないなら
  if ( (bWinNT || bWin32S) && !bIsShellExplorer ) then
   //MessageBox( "NT でシェルがエクスプローラでない", INFORMATION ); //20000406 DR.H.K.
      //プログラムマネージャに新規グループを作成する
      AppCommand( PROGMAN, CMD_RESTORE );
      CreateProgramFolder( svFolder );
      ShowProgramFolder( svFolder, 0 );
      LongPathToShortPath( svTarget );
      Delay(1);
  else
   //MessageBox( "NT ではないか、シェルがエクスプローラである場合", INFORMATION ); //20000406 DR.H.K.
      //プログラムマネージャに新規グループを作成する
      AppCommand( PROGMAN, CMD_RESTORE );
      CreateProgramFolder( svFolder );
      ShowProgramFolder( svFolder, 0 );
      LongPathToShortPath( svTarget );
      Delay(1);
  endif;

  TARGETDIR = svTarget;

  if (ComponentIsItemSelected( szComponentList, ITEM_PROGRAMFILES )) then

      szProgram = TARGETDIR ^ "ksim¥¥bin¥¥sim.exe";
      // NT でシェルがエクスプローラでないなら
      if ((bWinNT || bWin32S) && !bIsShellExplorer ) then
     //アイコンを登録する
     //MessageBox("NT でシェルがエクスプローラでない", INFORMATION ); //20000406 DR.H.K.
     //MessageBox( APP_NAME ^ "アイコンを登録する", INFORMATION ); //20000406 DR.H.K.
     AddFolderIcon( svFolder, APP_NAME,
          szProgram,
          TARGETDIR ^ "ksim¥¥bin",
          "", 0, "", REPLACE );
//      Delay( 1 );
//      AddFolderIcon( svFolder, "メインメニュー",
//           "keikan.exe",
//           TARGETDIR ^ "ksim¥¥bin",
//           "", 0, "", REPLACE );
//      Delay( 1 );
//      AddFolderIcon( svFolder, "優良 景観 事例 検索",
//           "yuu.exe",
//           TARGETDIR ^ "ksim¥¥bin",
//           "", 0, "", REPLACE );
//      Delay( 1 );
//      AddFolderIcon( svFolder, "景観 材料 検索",
//           "zai.exe",
//           TARGETDIR ^ "ksim¥¥bin",
//           "", 0, "", REPLACE );
//      Delay( 1 );
//      AddFolderIcon( svFolder, "景観 構成 要素 検索",
//           "kou.exe",
```

```
//              TARGETDIR ^ "ksim¥¥bin",
//              "", 0, "", REPLACE );
//      Delay( 1 );
//      AddFolderIcon( svFolder, "市街地生成",
//               "都市開発.EXE",
//               TARGETDIR ^ "都市開発",
//               "", 0, "", REPLACE );
//      Delay( 1 );
//      AddFolderIcon( svFolder, "貿易",
//               "貿易.EXE",
//               TARGETDIR ^ "貿易 95",
//               "", 0, "", REPLACE );
      Delay( 1 );
      AddFolderIcon( svFolder, "お読み下さい",
            "NOTEPAD.EXE " + TARGETDIR ^ "README.TXT",
            TARGETDIR,
            "", 0, "", REPLACE );
      Delay( 1 );
       else
           //MessageBox("「NT でシェルがエクスプローラでない」ことはない", INFORMATION ); //20000406 DR.H.K.
           //MessageBox( szProgram, INFORMATION ); //20000406 DR.H.K.
           //MessageBox( TARGETDIR, INFORMATION );
           //LongPathToQuote( szProgram, TRUE ); //"" で囲む
       Delay( 1 );
       AddFolderIcon( svFolder, APP_NAME,
           szProgram,
           TARGETDIR ^ "ksim¥¥bin",
           "", 0, "", REPLACE );
//       Delay( 1 );
//       AddFolderIcon( svFolder, "優良景観事例検索",
//            TARGETDIR ^ "ksim¥¥bin¥¥yuu.exe",
//            TARGETDIR ^ "ksim¥¥bin",
//            "", 0, "", REPLACE );
//       Delay( 1 );
//       AddFolderIcon( svFolder, "景観構成要素検索",
//            TARGETDIR ^ "ksim¥¥bin¥¥kou.exe",
//            TARGETDIR ^ "ksim¥¥bin",
//            "", 0, "", REPLACE );
//       Delay( 1 );
//       AddFolderIcon( svFolder, "景観材料検索",
//            TARGETDIR ^ "ksim¥¥bin¥¥zai.exe",
//           TARGETDIR ^ "ksim¥¥bin",
//            "", 0, "", REPLACE );
//       Delay( 1 );
//       AddFolderIcon( svFolder, "景観データベース入力",
//            TARGETDIR ^ "ksim¥¥bin¥¥editor.exe",
//            TARGETDIR ^ "ksim¥¥bin",
//            "", 0, "", REPLACE );
      endif;
      Delay( 1 );
  endif;

  if ((bWinNT || bWin32S) && !bIsShellExplorer ) then

      // Global variable UNINST stores the name and location of the
      // uninstaller file.
      //MessageBox( "Global variable UNINST stores the name and location of the uninstaller file.",
INFORMATION ); //20000406 DR.H.K.
      szProgram = UNINST;
      LongPathToShortPath( szProgram );
      LongPathToShortPath( svUninstLogFile );
      szProgram = szProgram + " -f" + svUninstLogFile;
      AddFolderIcon( svFolder, "アンインストール", szProgram,
          WINDIR, "", 0, "", REPLACE );
      Delay( 1 );
  endif;
```

```
    // Announce setup complete and offer to read README file.
    CloseOfInstall:
     SetStatusWindow( 98 , "環境設定ファイルの調整中...." );
     //MessageBox("ディレクトリの作成",INFORMATION); //20000406 DR.H.K.
     CreateDir( svTarget ^ "¥¥ksim¥¥temp" );
     //MessageBox("環境ファイルの HOMEPATH の設定",INFORMATION); //20000406 DR.H.K.
     VarSave(SRCTARGETDIR);
     SRCDIR = svTarget ^ "ksim¥¥bin";
//     FileGrep("kdbms.set", "KEIKAN_HOMEPATH", svLine, nvLineNumber, RESTART); 970407 DR.H.K.
     FileGrep("kdbms.set", "HOME_PATH", svLine, nvLineNumber, RESTART);
     FileDeleteLine("kdbms.set", nvLineNumber, nvLineNumber);
     FileInsertLine("kdbms.set", "HOME_PATH = " + svTarget + ";", nvLineNumber, BEFORE);

  SetStatusWindow( 100, "インストールの完了。" );

  SetupFinish();

  szMsg = "セットアッププログラムにより環境設定を変更しました" +
    "変更を有効にするには再起動してください";
  if( RebootDialog("windows 再起動", szMsg, SYS_RESTART) = 0) then
   CommitSharedFiles(0);
  endif;


  exit;


/*-------------------------------------------------------------------------*¥
 *
 * Function:   SetupScreen
 *
 * Purpose:   This function will set up the screen look.   This includes
 *        colors, fonts, text to be displayed, etc.
 *
 *
 *     Input:
 *
 * Returns:
 *
 * Comments:
¥*-------------------------------------------------------------------------*/

function SetupScreen()
  string szBitmap;
  number nvDx, nvDy, nDxBillboard, nDyBillboard;
begin
  GetExtents( nvDx, nvDy );

  Enable( FULLWINDOWMODE );
  Enable( INDVFILESTATUS );
  Enable( BITMAP256COLORS );   //256 色のビットマップを使用するために必ず必要

  SetColor( BACKGROUND, BK_BLUE ); // Gradient blue background.
  SetTitle( "セットアップ", 0, BACKGROUNDCAPTION ); // Caption bar text.

  Enable( BACKGROUND );

  // Show the bitmap.
  Enable( BITMAPFADE );
  szBitmap = SUPPORTDIR ^ "TITLE.BMP"; //ビットマップファイル名の作成
  //ビットマップファイルの位置指定と表示
  PlaceBitmap(szBitmap , 1, 10, 10, UPPER_LEFT | BITMAPICON );
  Disable( BITMAPFADE );

end;
```

```
/*---------------------------------------------------------------------------*¥
 *
 * Function:   CheckRequirements
 *
 * Purpose:   This function will check all minimum requirements for the
 *         application being installed. If any fail, then the user
 *         is informed and the installation is terminated.
 *
 *
 *     Input:
 *
 * Returns:
 *
 * Comments:
¥*---------------------------------------------------------------------------*/
function CheckRequirements()
number nvDx, nvDy;
number nvResult;
string svResult;
begin

   // Determine target system's operating system.
   GetSystemInfo( OS, nvResult, svResult );
   bWinNT = FALSE;
   bWin32S = FALSE;
   if (nvResult = IS_WINDOWSNT) then
       bWinNT = TRUE;

       // Check to see if NT is using EXPLORER Shell
       if( QueryShellMgr( svResult ) = 0 ) then
         if( StrCompare( svResult, "EXPLORER.EXE" ) = 0 ) then
           bIsShellExplorer = TRUE;
         endif;
       endif;

   elseif (nvResult = IS_WIN32S) then
       bWin32S = TRUE;
   endif;

   // Check screen resolution.
   GetExtents( nvDx, nvDy );
   if (nvDy < 480) then
       MessageBox( "このプログラムを実行するには VGA 以上の解像度が必要です。", WARNING );
       exit;
   endif;

end;

/*---------------------------------------------------------------------------*¥
 *
 * Function:   SetupFileTransfer
 *
 * Purpose:   This function defines the file set based on the user's choices
 *         of components, then it performs the file set.
 *
 *
 *     Input:
 *
 * Returns:
 *
 * Comments:
¥*---------------------------------------------------------------------------*/

function SetupFileTransfer( szComponentList, szFileSet )
begin

   // Define the file set.
```

```
   TARGETDIR = svTarget;

   FileSetBeginDefine( szFileSet );

       // Always include README files, located at the root of the
       // DATA.Z library file.
       SetStatusWindow( -1, "基本情報 ファイルをコピー中..." );
       TARGETDIR = svTarget;
       nRet = CompressGet( "data.z", "*.*", COMP_NORMAL );
     if( nRet < 0 ) then
     MessageBox("CompressGet failed(COMP_NORMAL) ", WARNING);
    endif; //20000523 DR.H.K.
       // If program files are selected, then add them to file set.
       if (ComponentIsItemSelected( szComponentList, ITEM_PROGRAMFILES )) then
       SetStatusWindow( -1, "ファイルをコピー中..." );
       TARGETDIR = svTarget;
       nRet = CompressGet( "data.z", "*.*", INCLUDE_SUBDIR );
     if( nRet < 0 ) then
       MessageBox("CompressGet failed(INCLUDE_SUBDIR)", WARNING);
            endif;
       endif;
   FileSetEndDefine( szFileSet );

end;


/*---------------------------------------------------------------------------*¥
 *
 * Function:   PerformFileTransfer
 *
 * Purpose:   This function will perform the file transfer and handle
 *        any error that may occur during the file transfer.
 *
 *
 *     Input:
 *
 * Returns:
 *
 * Comments:
¥*---------------------------------------------------------------------------*/

function PerformFileTransfer( szFileSet )

begin

   StatusUpdate( ON, 90 );

   // Perform the file set.
   nResult = FileSetPerformEz( szFileSet, 0 );

   switch (nResult)
   case FS_DONE: // Successful completion.

   case FS_CREATEDIR: // Create directory error.
      MessageBox( TARGETDIR + "の下にディレクトリを作成することができません。" + "."+
          "ディレクトリの書き込み属性を確認して下さい。", SEVERE );
      exit;
   // 20000405 DR.H.K. エラー発生のため、メッセージを追加。
   case 0:   MessageBox( "転送成功", SEVERE );
   case FS_TONEXTDISK: MessageBox( "次のディスク", SEVERE);
    exit;
   case FS_FILENOTINLIB:
    MessageBox( "FILENOTINLIB", SEVERE);
    exit;
   case FS_GENERROR:
    MessageBox( "GENERROR", SEVERE);
    exit;
```

```
   case FS_INCORRECTDISK:
    MessageBox( "INCORRECTDISK", SEVERE);
    exit;
   case FS_LAUNCHPROCESS:
    MessageBox( "LAUNCHPROCESS", SEVERE);
    exit;
   case FS_OPERROR: //圧縮前のソースのルートに DeIsL1.ins 等があると、このエラーが生じる場合がある。
    MessageBox( "OPERROR", SEVERE);
    exit;
   case FS_PACKAGING:
    MessageBox( "PACKAGING", SEVERE);
    exit;
   case FS_RESETREQUIRED:
    MessageBox( "RESETREQUIRED", SEVERE);
    exit;
   default: // Group all other errors under default label.
     NumToStr( szTemp, nResult );
     MessageBox( "一般的な転送エラー。"+
          "インストール先を確認して再度実行して下さい。"+
          "¥n¥n エラー番号:"+szTemp, SEVERE );

     exit;
   endswitch;
end;



/*---------------------------------------------------------------------------*¥
 *
 * Function:   DetermineComponentInfo
 *
 * Purpose:   This function will place all the separatly identifable
 *       file components into a component list. It will also
 *       set the size the files represent and will by default
 *       enable each to be selected.
 *
 *
 *    Input:
 *
 * Returns:
 *
 * Comments:
¥*---------------------------------------------------------------------------*/

function DetermineComponentInfo( szFileList )
number nvSize;
string svSize;
begin

   CompressInfo( "data.1", "ksim¥¥bin¥¥*.exe",
         COMP_INFO_ORIGSIZE | INCLUDE_SUBDIR,
         nvSize, svSize );
   ComponentAddItem( szFileList, ITEM_PROGRAMFILES, nvSize, TRUE );
   CompressInfo( "data.1", "kdb¥¥*.*",
         COMP_INFO_ORIGSIZE | INCLUDE_SUBDIR,
         nvSize, svSize );
   ComponentAddItem( szFileList, ITEM_PROGRAMFILES, nvSize, TRUE );


   CompressInfo( "data.1", "ksim¥¥bin¥¥*.set",
         COMP_INFO_ORIGSIZE | INCLUDE_SUBDIR,
         nvSize, svSize );
   ComponentAddItem( szFileList, ITEM_ENVFILES, nvSize, TRUE );

   CompressInfo( "data.1", "ksim¥¥bin¥¥*.DLL",
         COMP_INFO_ORIGSIZE | INCLUDE_SUBDIR,
```

```
            nvSize, svSize );
    ComponentAddItem( szFileList, ITEM_DLLFILES, nvSize, TRUE );

end;

/*---------------------------------------------------------------------------*¥
 *
 * Function:    EnoughSpace
 *
 * Purpose:    This function will determine if enough space exists on
 *          the target drive based on the selection in thge component
 *          list.
 *
 *
 *      Input:
 *
 * Returns:
 *
 * Comments:
¥*---------------------------------------------------------------------------*/
function    EnoughSpace( svTarget, szComponentList, nExtraSpace )
LIST list;
number nTotal, nvSize, nFreeSpace, nResult;
string svComponent;
begin

    // Get total size of all selected components.
    list      = ListCreate( STRINGLIST );
    nTotal = nExtraSpace;
    ComponentListItems( szComponentList, list );

    nResult = ListGetFirstString( list, svComponent );

    while (nResult = 0)
        ComponentGetItemSize( szComponentList, svComponent, nvSize );
        if (ComponentIsItemSelected( szComponentList, svComponent )) then
       nTotal = nTotal + nvSize;
        endif;
        nResult = ListGetNextString( list, svComponent );

    endwhile;

    ListDestroy( list );

    // Determine if target disk has enough space.
    nFreeSpace = GetDiskSpace( svTarget );

    if (nFreeSpace < nTotal) then
        szMsg = "ディスク上に十分な空き容量がありません。¥n" +
            "'" + svTarget + "' ¥n" +
            "空き容量を確保するか、インストール先を別のディスクに¥n" +
            "変更して下さい。";
        SprintfBox( WARNING, "Setup", szMsg, nTotal );
        return FALSE;
    endif;

    return TRUE; // Enough space.
end;

//---------------------------------------------------------------------------
//
//    Name    : ConstructInfoList
//
//    Purpose : This function will construct a info list for showing the
//          SdStartCopy dialog to confirm the file transfer operation.
//
//---------------------------------------------------------------------------
function ConstructInfoList( listInfo )
```

```
    STRING szItem;
begin
    ListAddString( listInfo, "セットアップ方法", AFTER );

    switch (nType)
    case TYPICAL:
        ListAddString( listInfo, STR_DEFTAB + "標準", AFTER );
        szItem = STR_DEFTAB + STR_DEFTAB + ITEM_PROGRAMFILES;
        ListAddString( listInfo, szItem, AFTER );
        szItem = STR_DEFTAB + STR_DEFTAB + ITEM_ENVFILES;
        ListAddString( listInfo, szItem, AFTER );
        szItem = STR_DEFTAB + STR_DEFTAB + ITEM_DLLFILES;
        ListAddString( listInfo, szItem, AFTER );
    case COMPACT:
        ListAddString( listInfo, STR_DEFTAB + "コンパクト", AFTER );
        szItem = STR_DEFTAB + STR_DEFTAB + ITEM_PROGRAMFILES;
        ListAddString( listInfo, szItem, AFTER );
    case CUSTOM:
        ListAddString( listInfo, STR_DEFTAB + "カスタム", AFTER );
        if (ComponentIsItemSelected( szComponentList, ITEM_PROGRAMFILES )) then
        szItem = STR_DEFTAB + STR_DEFTAB + ITEM_PROGRAMFILES;
        ListAddString( listInfo, szItem, AFTER );
        endif;
        if (ComponentIsItemSelected( szComponentList, ITEM_ENVFILES )) then
        szItem = STR_DEFTAB + STR_DEFTAB + ITEM_ENVFILES;
        ListAddString( listInfo, szItem, AFTER );
        endif;
        if (ComponentIsItemSelected( szComponentList, ITEM_DLLFILES)) then
        szItem = STR_DEFTAB + STR_DEFTAB + ITEM_DLLFILES;
        ListAddString( listInfo, szItem, AFTER );
        endif;
    endswitch;

    ListAddString( listInfo, "", AFTER );
    ListAddString( listInfo, "インストール先ディレクトリ", AFTER );
    ListAddString( listInfo, STR_DEFTAB + svTarget, AFTER );

    ListAddString( listInfo, "", AFTER );
    ListAddString( listInfo, "プログラム フォルダー", AFTER );
    ListAddString( listInfo, STR_DEFTAB + svFolder, AFTER );
end;

/*---------------------------------------------------------------------------*¥
 *
 * Function:    CreateRegDBEntries
 *
 * Purpose:    This function will create necessary keys and values for
 *          the sample program.
 *
 *      Input:
 *
 * Returns:
 *
 * Comments:
¥*---------------------------------------------------------------------------*/

function CreateRegDBEntries()
    string szKey[255], szValue, szDemo, szProgram;
begin



    //ＮＴの環境変数は HKEY_CURRENT_USER/Environment/KSIM_ENV に書く
    Disable(LOGGING);
    RegDBSetDefaultRoot( HKEY_CURRENT_USER );
    szKey = "Environment";
//    szKey = "Environt";
```

```
   RegDBCreateKeyEx( szKey, "" );
   RegDBSetKeyValueEx( szKey, "KSIM_ENV", REGDB_STRING, svInstallDir ^ "ksim¥¥bin¥¥kdbms.set", -1 );
   Enable(LOGGING);

//add .scn file

   RegDBSetDefaultRoot( HKEY_CLASSES_ROOT );
   szKey = ".scn";
   RegDBCreateKeyEx( szKey, "" );
   RegDBSetKeyValueEx( szKey, "", REGDB_STRING, "SCN_auto_file", -1 );

//   szKey ="Scn_auto_file"+"¥¥"+"shell"+"¥¥"+"open"+"¥¥"+"command";
   szKey = "Scn_auto_file¥¥shell¥¥open¥¥command";
   RegDBCreateKeyEx( szKey, "" );
//   RegDBSetKeyValueEx( szKey, "", REGDB_STRING, svInstallDir ^ "ksim¥¥bin¥¥sim.exe ¥"1%¥"", -1 );
//   RegDBSetKeyValueEx( szKey, "", REGDB_STRING, svInstallDir ^ "ksim¥¥bin¥¥sim.exe ¥" ¥"1%¥"", -1 );
//   RegDBSetKeyValueEx( szKey, "", REGDB_STRING, svInstallDir ^ "ksim¥¥bin¥¥sim.exe 1%", -1 );
   RegDBSetKeyValueEx( szKey, "", REGDB_STRING, svInstallDir ^ "ksim¥¥bin¥¥sim.exe ¥%1", -1 );
//ren2001-09-07
//    LaunchAppAndWait( "regsvr32.exe", svInstallDir ^ "ksim¥¥bin¥¥sim_ocx.dll", WAIT ) ; 010927 DR.H.K.
sim_id.ocx への差し替えにより、省く。
//dsb. Zhu@DDD 来所作業
#if 0
   //その他のレジストリを作成する
   RegDBSetDefaultRoot( HKEY_LOCAL_MACHINE );

   // Create PRODUCT_KEY key.
   szKey = "SOFTWARE¥¥" + COMPANY_NAME + "¥¥" + PRODUCT_NAME + "¥¥" +
      PRODUCT_VERSION + "¥¥" + "DESIGNER";
   RegDBCreateKeyEx( szKey, "" );

   RegDBSetKeyValueEx( szKey, "Template", REGDB_STRING, "good.tpl", -1 );
   RegDBSetKeyValueEx( szKey, "TemplatePath", REGDB_STRING, svTarget ^ "TEMPLATE", -1 );
   RegDBSetKeyValueEx( szKey, "x",    REGDB_NUMBER, "2", -1 );
   RegDBSetKeyValueEx( szKey, "y",    REGDB_NUMBER, "3", -1 );
   RegDBSetKeyValueEx( szKey, "dx", REGDB_NUMBER, "637", -1 );
   RegDBSetKeyValueEx( szKey, "dy", REGDB_NUMBER, "448", -1 );
   RegDBSetKeyValueEx( szKey, "LargeDraw", REGDB_NUMBER, "1", -1 );
   RegDBSetKeyValueEx( szKey, "PopupMenu", REGDB_NUMBER, "1", -1 );
   RegDBSetKeyValueEx( szKey, "NoPopup", REGDB_NUMBER, "1", -1 );
   RegDBSetKeyValueEx( szKey, "StartDialogValue", REGDB_NUMBER, "111", -1 );
   RegDBSetKeyValueEx( szKey, "TimeVisible", REGDB_NUMBER, "0", -1 );

   if (ComponentIsItemSelected( szComponentList, ITEM_ENVFILES )) then
      // Create "DEMOS" key.
      szKey = "SOFTWARE¥¥" + COMPANY_NAME + "¥¥" + PRODUCT_NAME + "¥¥" +
         PRODUCT_VERSION + "¥¥" + "DEMOS";
      RegDBCreateKeyEx( szKey, "" );

      szDemo   = svTarget ^ "ksim¥¥bin¥¥sim.exe";
      szProgram = svTarget ^ "ksim¥¥bin¥¥sim.exe";
      RegDBSetKeyValueEx( szKey, "path0", REGDB_STRING, szDemo, -1 );
      RegDBSetKeyValueEx( szKey, "tour",    REGDB_STRING, szDemo, -1 );

      szDemo   = svTarget ^ "SAMPLES¥¥2MINUTE.DBD";
      RegDBSetKeyValueEx( szKey, "path1", REGDB_STRING, szDemo, -1 );

      RegDBSetKeyValueEx( szKey, "exe", REGDB_STRING, szProgram, -1 );
      RegDBSetKeyValueEx( szKey, "active", REGDB_STRING, "Play", -1 );

   endif;

   // Create "HELPMENU" key.
   szKey = "SOFTWARE¥¥" + COMPANY_NAME + "¥¥" + PRODUCT_NAME + "¥¥" +
      PRODUCT_VERSION + "¥¥" + "HELPMENU";
   RegDBCreateKeyEx( szKey, "" );

   RegDBSetKeyValueEx( szKey, "MaxNum", REGDB_NUMBER, "1", -1 );
```

```
   RegDBSetKeyValueEx( szKey, "path0", REGDB_STRING, svTarget ^ "README.TXT", -1 );
   RegDBSetKeyValueEx( szKey, "exe0", REGDB_STRING, "NOTEPAD.EXE", -1 );
   RegDBSetKeyValueEx( szKey, "active0", REGDB_STRING, "Read Me", -1 );

   // Create "MRU" key.
   szKey = "SOFTWARE¥¥" + COMPANY_NAME + "¥¥" + PRODUCT_NAME + "¥¥" +
      PRODUCT_VERSION + "¥¥" + "MRU";
   RegDBCreateKeyEx( szKey, "" );

   szDemo = svTarget ^ "SAMPLES¥¥2MINUTE.DBD";
   RegDBSetKeyValueEx( szKey, "path0", REGDB_STRING, szDemo, -1 );

   // Create "SceneEditor" key.
   szKey = "SOFTWARE¥¥" + COMPANY_NAME + "¥¥" + PRODUCT_NAME + "¥¥" +
      PRODUCT_VERSION + "¥¥" + "SceneEditor";
   RegDBCreateKeyEx( szKey, "" );

   RegDBSetKeyValueEx( szKey, "x",    REGDB_NUMBER, "453", -1 );
   RegDBSetKeyValueEx( szKey, "y",    REGDB_NUMBER, "2", -1 );
   RegDBSetKeyValueEx( szKey, "Visible",    REGDB_NUMBER, "1", -1 );

   // Create "ToolPalette" key.
   szKey = "SOFTWARE¥¥" + COMPANY_NAME + "¥¥" + PRODUCT_NAME + "¥¥" +
      PRODUCT_VERSION + "¥¥" + "ToolPalette";
   RegDBCreateKeyEx( szKey, "" );

   RegDBSetKeyValueEx( szKey, "x",    REGDB_NUMBER, "509", -1 );
   RegDBSetKeyValueEx( szKey, "y",    REGDB_NUMBER, "155", -1 );
   RegDBSetKeyValueEx( szKey, "Visible",    REGDB_NUMBER, "1", -1 );

   // Create "AlignPalette" key.
   szKey = "SOFTWARE¥¥" + COMPANY_NAME + "¥¥" + PRODUCT_NAME + "¥¥" +
      PRODUCT_VERSION + "¥¥" + "AlignPalette";
   RegDBCreateKeyEx( szKey, "" );

   RegDBSetKeyValueEx( szKey, "Visible",    REGDB_NUMBER, "0", -1 );

   // Create "Controller" key.
   szKey = "SOFTWARE¥¥" + COMPANY_NAME + "¥¥" + PRODUCT_NAME + "¥¥" +
      PRODUCT_VERSION + "¥¥" + "Controller";
   RegDBCreateKeyEx( szKey, "" );

   RegDBSetKeyValueEx( szKey, "x",    REGDB_NUMBER, "420", -1 );
   RegDBSetKeyValueEx( szKey, "y",    REGDB_NUMBER, "231", -1 );
   RegDBSetKeyValueEx( szKey, "Visible",    REGDB_NUMBER, "1", -1 );
#endif
end;

/*---------------------------------------------------------------------

    Name      : SetupFinish

    Purpose : This function will construct messages and info for showing
      SdFinish

---------------------------------------------------------------------*/
function SetupFinish()
   string szMsg1, szMsg2;
   string szReadme, szApp, szProgram, szParam;
   BOOL    bReadme, bApp;
begin
   if (BATCH_INSTALL = TRUE) then
      szMsg = "いくつかのファイルは、現在システムの他のプログラムで使用中のため"+
         "インストールする事ができませんでした。"+
         "新しいプログラムにより正しい操作を行うためには、システムを再スタート"+
         "しなければなりません。";
```

```
        CommitSharedFiles(0);
        RebootDialog( "Restart Windows", szMsg, SYS_BOOTMACHINE );
    else

        bReadme = FALSE;
        bApp    = FALSE;

        szMsg1 = "セットアップが完了しました！。インストールされたプログラムを実行するには、"+
        "登録されたアイコンをダブルクリックして下さい。¥n¥n プログラムを実行する前に"+
        "README アイコンをダブルクリックして内容を確認して下さい。";

        szMsg2 = "「終了」をクリックするとセットアップが完了します。";


        bReadme = TRUE;
        szReadme = "ReadMe ファイルを参照します。";

        szApp = "";
#if 0
        if ((ComponentIsItemSelected(szComponentList, ITEM_PROGRAMFILES)) &&
          (ComponentIsItemSelected(szComponentList, ITEM_ENVFILES))) then
        szApp = "はい、Example1 を直ちに実行します。";
        endif;
#else
     szApp = "";
#endif
        MessageBeep( 0 );
        SdFinish( "", szMsg1, szMsg2, szReadme, szApp, bReadme, bApp );

        if ( bReadme ) then
        szProgram = "NOTEPAD.EXE";
        szParam = svTarget ^ "README.TXT";
        LongPathToShortPath( szParam );
        LaunchAppAndWait( szProgram + " " + szParam, "", NOWAIT );
        Delay(2);
        endif;

        if ( bApp ) then
        szProgram = TARGETDIR ^ "ksim¥¥bin¥¥sim.exe ";
        szParam = TARGETDIR ^ "ksim¥¥bin¥¥sim.exe";
        LaunchAppAndWait( szProgram, szParam, NOWAIT );
        endif;
    endif;
end;


#include "sddialog.rul"
```

このソースコードはコンパイルされるものである。BASIC 風の書法で、メインの処理と、そこから呼び出される関数、サブルーチン等を含んでいる。

１６．総括と提言

はじめに

第２～１５の各章においては、1993～2008 年度にわたり開発・改良を進めてきた景観シミュレーション・システムの全体構成と各構成要素、および枝分かれバージョンの各機能の統合を実現するために用いられたアルゴリスムに関して解説した。その主たる目的は、本システムを今後更に改良しようとするプログラマ、あるいは本システムのアルゴリスムやライブラリ関数等を、部品として別システムに組み込もうとするプログラマに対して情報公開を行うことにある。

本章では、国立研究機関により開発した公共財としてのオープンソースのシステムの意義と、関連する諸問題について総括すると共に、15 年間の経験に基づいて、今後 IT に関する同様の、あるいは別の目的のためのシステム開発を行おうとする研究開発機関あるいは建設現場の担当者のための示唆を引き出そうとするものである。

１６－１．IT 化のインパクト

景観シミュレーション・システムの開発に着手した 1993 年当時は、ユーザーとして想定する現場に PC が導入途上にあり、利用状況には事務所間で大きな格差があった。導入されている事務所においては、Macintosh よりは Windows3.1 を OS とするものが多く、ビットマップ型のディスプレイ上でマウスを用いて操作する GUI が普及途上にあった。その後急速に進み、ワードプロセッサや表計算ソフトウェアが業務にも使用され、やがてネットワークによる情報交換が行われるようになっていくオフィス環境変化の初期の段階にあった。

（1）真の持続可能なシステム

ＩＴの導入（とりわけ商用）への勧奨は、しばしばデータの持続性をその効用の一つとして掲げる。アナログな実世界をデジタルに計測し、デジタルに記録された情報は、劣化することなしに長期にわたり保存し、コピーし、活用することが技術的には可能となる。例えば CALS は、LS（ライフ・サイクル）にわたる持続的な情報サポートを理念としている。

しかしながら、現実のコンピュータ化・デジタル化の世界は日進月歩・ドッグイヤーの世界である。今日作成した CAD データは、５年後には遠い過去のデジタル古文書と化す。

例えば、本システムの開発に着手した 1993 年頃の官庁営繕分野における技術研究会に参加したことがあるが、設計時に作成された CAD データを長期にわたり保存・活用するための媒体として、８インチフロッピーディスクを用いるべきか、それとも 3.5 インチ光磁気ディスク（MO）を用いるべきかが真剣に議論されていた。現在、これらの媒体に記録されている情報を読み出して利用できる環境（ドライブ等）は既に希少である。

これは根底からの矛盾であり、本来は建物に関する情報を保管する使命を帯びた「国立

公文書館」のような組織機関こそ検討されるべきである。真剣に情報を長期にわたり保管することを考えている者は、実績のない全く新しく開発されたシステムへの情報の載せ替えには、たとえそれが「情報の永久的な維持管理」を標榜するものであっても躊躇するに違いない。

理論的・技術的に可能であり容易であるはずの情報の長期的な維持管理を実務的に実現するためには、少なくとも保管のための物理的な媒体が十分な耐久性を有し、データ形式などが（たとえ時代遅れではあっても）永久不変であることが求められる。

同じことはソフトウェアに関してもあてはまる。ある技術を具体的な処理システムとして実現したソフトウェア資産は、劣化することなく継承発展させることが可能であるはずである。しかしながら、ハードウェアや OS の急速な進化により、同じプログラムを実行できる動作環境が失われる。継続的に使用するためには、その都度の環境変化に適応するために継続的にソフトウェアのアップデートへの投資を続けなければならない。

１００年後の歴史家は、１７世紀の和紙に筆で書かれた若干の虫食いのある地方文書を現在と同様に解読できるであろうが、倉庫に安全に保管されている 20 世紀末のハードディスク装置の中に記録されている、当時のワードプロセッサで作成された文書を読み出して解読するためには、かなり苦労することが想像される。

つまり、本来永久不変である筈のデジタル化された情報が、現時点では、絶えずアップデートと媒体変換し続けなければ維持継承できなくなっている、という意味で、あたかも稗田阿礼以前の口述伝承のような世界に戻ってしまったかのごとくである。

しかしながら、IT に関する研究開発においては、最先端のデバイスや技術や利用技術への関心に隠れて、上記のような問題に関する議論はあまりに少ないように感じられる。

（2）途上国におけるデジタル化

筆者は、1984 年から継続的にインドネシア共和国の公共事業省人間居住研究所(英語名 Research Institute for Human Settlements, Ministry of Public Works、現地名 Pusat Penelitian dan Pengembangan Permukiman, Departemen Pekerjaan Umum)に対して技術協力のために継続的に派遣される機会があり、都市住宅政策分野に関連するその都度のテーマを具体的な課題としつつ、25 年間にわたり定点観測的に同研究所における最初のコンピュータ導入から、最近のネットワークや GIS 導入に至るまで助言と観察を続けてきた。その間、記憶媒体は、紙から５インチフロッピーディスク、3.5 インチフロッピーディスク、CD-R、スティック・メモリと変化してきた。

初期の状況は、日本建築学会などのような論文をデータベースとして管理する機関が存在しない同国において、国立研究所が発行する機関誌「Masalah Bangunan」（「建築問題」と訳せる）が唯一の専門誌であり、テンポラリーに作成されるセミナーの「Proceedings」が情報源であった。研究所が備品として維持管理し専門職員が操作する「印刷機」は、同研究所の重要なリソースであり、社会的使命を果たしていた。これに概要が報告される研究成果自体は、個別の報告書（多くは国際援助として実施されたプロジェクトの成果であ

る）として記録されるものの印刷部数は少なく、報告書の質は玉石混交であって、整理保管の体制が不十分な図書室等においては、価値のある文献である程よく貸し出され、返却されないことにより失われていく。担当官吏が個人的に保管している報告書等も同様で、「本を貸す馬鹿返す馬鹿」のような状況であった。見知らぬ来訪者に「その問題に関しては調査報告書がある」と言えば、「見せてくれ」と言われる。見せると「貸してくれ」と言われる。貸すと返って来ない。このような状況であったから、報告書を保管している関係官吏は、その所在についてなかなか語らない。結果的に、価値のある文献ほど、アクセスが困難であり、その事が知的財産の社会的普及・活用を妨げていた。

このような状況を変えたのが、1990 年代に入ってからのコピー・サービスの急速な社会的普及で、低価格の小型コピー機を用いた零細なコピー店がまず街中に急増した。人件費が安いことから、コピーし簡易製本まで安価に依頼することができるようになり、これにより原本を長期に借り出すことなく情報を入手することが技術的に可能となった。これは情報公開・流通を大きく促進した。

1987 年頃から、IBM－PC/XT をモデルとした台湾製のコンパチブル機（ハードディスクなし）が約 10 万円程度で市販された）が徐々に普及し、WORD-STAR により入力された文書ファイルを、５インチ・フロッピーディスクにコピーして入手することが可能となった。日本から供与した NEC-9801E とは媒体による互換性がなく、RS-232C クロスケーブルを製作して接続し、ソフトウェアを用いてファイルを転送した。

次の転機は 1997 年のアジア通貨危機に訪れた。ドルや円に対する為替レートが約 5 分の１に急落し、輸入に依存していた文具や紙製品の価格が急騰した。これにより、紙のコピーや FAX を使用することが急減し、必要な情報をフロッピーディスク等により交換し画面で表示して用を足すことが一般的となった。つまり、アジア通貨危機は研究所のペーパーレス・オフィス化を一挙に進めた。日本では現在でも大量の紙コピーの資料を会議で配布するスタイルが継続しているが、これは紙が安いからできる贅沢であると言える。

2000 年代に入ると、安価な CD-R が普及し、これを用いて大規模な CAD や GIS ソフトウェアの海賊版が電気街で販売されるようになった。概ね 1000 円程度で入手可能なこれらの製品は、学生などにも広く普及した。このような状況下で、地図や建築設計図のデータが、紙ではなく CD-R を媒体として交換されるようになった。これらのソフトウェアの価格は日本では 25～100 万円程度であることから、日本の学生は現在でも大学の計算機センターで触れることはあっても、在宅やテイクアウトで日常的に触ることはまだ実現されていない。途上国では建築設計事務所などにおいても、CAD ソフトや GIS ソフトを用いて作業が行われることが直ちに一般的となり、製図版やドラフターは殆ど見られない。

但し、途上国におけるこのような情報流通面におけるデジタル化とは対照的に、情報の長期的な保存管理という点に関しては、未だに非常に脆弱である。コンピュータ・ウィルスが蔓延していることから、ハードディスクに集約・蓄積されている情報の安全性は低い。「10 年前のセミナー論文のデジタルデータ」を現地で探すことはかなり困難である。つま

り、情報のデジタル化・ペーパーレス化が進んだものの、長期的な保存・アーカイブ化に関しては（もし社会的な必要があるとすれば）大きな課題を潜在的に抱えているのが現状である。しかし、少なくとも「簡単にコピーできる」ことが情報の社会的普及に加え原資料の中期的保存に貢献していることは確かである。

（3）三次元データの意味

設計図から出来形を想像することが職業人の役割であるとするならば、三次元 CG 技術は、変化する技術環境下で、その想像力を補助する意味を有している。技術と様式が安定していれば、簡単な板図のみで住宅の施工が可能であった時代は遠い過去ではない。

建築設計の教育においては、パースの作成がカリキュラムに組み込まれており、平面図・立面図・断面図から対応する代表点を拾い、作図により透視図上に点の位置を求め、それらを基点としながら、絵画としてパースを描画する訓練が行われた。筆者が学生であった 1971 年頃には、教育用電算機が使用可能となったため、射影変換のプログラムを組み、平面図・立面図から抽出した代表点の XYZ 座標をパンチカードで入力し、LP 用紙に打ち出された変換結果（UV 座標）を図にプロットする方法で作業能率と精度の向上を試みた。

一方、土木分野では、専ら力学的な基準に基づいて形状が決定され、それが風景に与えるインパクトについては、担当者も事前に十分に把握していないような状況が、1993 年頃まで続いていた。しかし、シビック・デザイン運動の高まりから景観への影響に関する関心が高まり、コンサルタントが作成する分厚い設計図書にパースが添付され始めていた。

紙の上に描かれた設計図は、二次元の図面による表現ではあるが、設計対象物の三次元形状を記述している。パースを作成する作業は、設計とは別の職能として補足的に行われていた。

三次元データによる画像処理は、この部分を大幅に迅速化し、三次元データから、任意視点のパースを瞬時に作成することを可能とした。視点を少しずつ移動させながら、１秒間に数十回パースを再描画することにより動画が生成する。更に、同じシステム上でモデリング、すなわち三次元データの編集を行うことができれば、表示結果を見ながら、設計内容自体を変更する、SBD（Simulation Based Design、シミュレーションに基づく設計）が景観に関して可能となる。

景観シミュレータに、価値判断の機能を持たせるべきか否かについては、未だに議論がある。景観シミュレーション・システムの開発にあたっては、まず三次元データのモデリングとパース表示のためのコストと時間を劇的に小さくすることにより、状況を変えようとした。これは、かつては専門職人に依頼しなければできなかった写真撮影が、測距や照度計測機能を備えたカメラの出現により、家庭の主婦でも可能となった事や、ワードプロセッサにより、印刷物の最終レイアウトまでが原稿執筆者に可能となった事に比較される技術的環境の変化をもたらすと期待したものである。即ち、当面の選択肢が景観に対してどのような影響を及ぼすのか、簡便にチェックできるような作業環境を設計担当者に対して提供することにより、カメラを手にした家庭の主婦やワープロを手にした作家のように、

最終的な出来形を認識しながら作業を進めることができ、同時に感性を磨くことにもつながるであろうと構想した。

景観検討の対象となる領域に関して三次元データを整備するためには、単に設計対象物のみならず、地形や周辺市街地などに関するデータを効率的に取得・作成することが必要であり、数値地図の整備や様々の立体的計測技術の発達、更には設計図書のデジタル化など、そのプロセスは現在進行中である。とりわけ、平成１８ 年度から開始された電子納品の三次元化の取り組みには、大いに期待されるものがある。

景観設計を入口として、設計対象物及び周辺環境の三次元データが次第に整備されるならば、そのデジタルデータは温熱環境、音響環境、電波環境などの領域にも利用可能である。また、道案内などの様々の行政サービスへの応用の展開も期待されている。

（４）サーバー側のアプリケーションとの連携

2001 年に開発した「まちづくり・コミュニケーション・システム」においては、景観シミュレータ sim.exe をローカルにセットアップした作業環境においてネットワーク上のサーバーからダウンロードした LSS-S ファイルを起点として、モデルを記述する LSS-G ファイルやテクスチャ、マテリアル、増補された外部関数実行形式など、関連ファイルをダウンロードし、景観として表示することができる。

サーバーからデータを配信する場合、配信されるファイルは必ずしも固定的である必要はなく、リクエストに応じて、サーバー側が保持するデータベースから選択されるファイルや、サーバー側での計算処理により動的に生成するファイルであってもよい。また、クライアント側での編集操作によりローカルに作成された三次元データをサーバーにアップロードすることも可能とした。

これらのサーバー側の機能により、多様なサーバー側の機能を発展させることが可能であり、そのための出発点となる基本構成を上記のシステムが提供している。

セキュリティを確保したサーバー上に基本データ、アプリケーション、セットアップ、ソースコード、関連ドキュメントを置いて配信するという体制は、デジタルデータの継続的な維持保管の問題を最終的に解決するものではないが、少なくともデータの保存継承の問題を、サーバーの維持管理の問題として集約整理することにはなると考えている。

１６－２．公共財としてのソフトウェア資産

（１）知的所有権

知的所有権は、それ自身が著作物として登録可能であり、昭和 62 年から、財団法人ソフトウェア情報センター(SOFTIC)において、マイクロ・フィッシュの形での登録・記録保存が行われている。ソフトウェアによる情報処理が、ハードウェア資源を用いて具体的に実現されていることを要件としてソフトウェア特許としての登録も可能となっている。景観シミュレーション・システムの場合には、公共財であって、これを使用しようとする者に対価を請求する意図が無いことから、そのような知的所有権保護のための手続きを一切行

ってこなかった。しかしながら、他者が本件に関する知的所有権を主張することや、公開し自由使用に供するソフトウェアを、メディア代以上の価格で再販売されることは好ましくないと考えた。そのため、1996 年以降は、本システムの開発成果を公知の事実とすることで、他者が知的所有物として登録することを防ぐために、その都度技術資料として公開した(付録 C 参照)。併せて、1996 年に建設省建築研究所の WEB サーバーから、また 2001 年以降は国土技術政策総合研究所の WEB サーバーからダウンロード･サービスを開始した。但しダウンロード・サービスを開始した時点では、通信速度に限界があったため、一時回線が混雑する状況も生じた。

（2）公的機関による開発の意義

大学などで研究開発が行われる場合には、公的資金により開発されたコア技術をもとに、投資を募り、更に応用開発を進めて製品化した上で、ベンチャービジネスとして、販売し収益を上げるということが行われる。

公的機関に限らず、ある組織が内部使用のためのシステム開発を行い、成果を公開せずに自社システムとして各地方支局などに導入することはありうる。必ずしも公開する必要はない。

自社システムを構築する場合には、必ずしもゼロからの開発を行う必要はなく、市販ソフトをカスタマイズするような方法が、より早い。景観シミュレーション・システムの場合には、開発に着手した時点で、現場に導入可能な価格で提供された製品などが存在していなかったために、ゼロからの開発を余儀なくされた。

開発のためのコストおよび工期の縮減に資する補助的なシステムは存在していた。例えば、ユーザー・インターフェースをグラフィックな開発環境でデザインした上で、各種プラットフォームのためのソースコードを出力するための開発システムも検討したことがある。しかしながら、最終成果であるアプリケーションを各現場で導入するに際して、ランタイムのライセンスを購入しなければならない、というビジネス・モデルであったために導入を断念した。

国の機関が民間に外注してソフトウェアを開発し、内部的に利用するような場合、当時その知的所有権に関しては必ずしも明確ではなかった。納品された実行形式に関して、仕様通りの動作が実現されていれば十分とされ、ソースコードの納品を求めないような場合も存在していた。本システムの開発にあたっては、建築研究所及び土木研究所からの外注により多くの部分が開発されたが、公開を前提とするソースコードの納品をもって成果とする契約方法とした。開発を受注した業者は、開発された技術を他の目的のために流用することが行為として可能であるが、これを契約上の守秘義務として縛ることは意義も実効性もないと考えられた。むしろソースコードをオープンソースとして公開することにより、他の者と全く同等の条件で開発成果を別の目的のために活用することが可能となる。そのことは、開発の上位目的である、本システムを具体的に活用した景観検討業務にとって何らの妨げになるものではない。

（3）公開の目的

オープンソースとして公開することの消極的な意味は、開発を担当した研究所が、持続的にメンテナンス（デバッグや、OS の変化への対応など）を行う当事者能力を有さないため、デバッグ等をユーザー側にゆだねる、という点にある。

一方、公的機関が国費で開発を行うことは、民業圧迫にあたるという批判も存在していた。ソースコードが公開されなければ、無償ソフトであってもその技術を民間の技術開発に用いることはできないであろう。ソースコードの公開が行われていれば、民間の技術開発は、公開技術を出発点として進めることができる。ソフトウェア開発が、商業的利益を目的とするものではなく、最終的には景観の向上を目的とするものである以上、民間の関連技術開発は推奨される必要がある。

1995 年当時、ソフトウェア分野における技術開発とそのための投資が、市場経済的に合理的な水準まで行われない理由として、違法コピーなどにより開発への投資が回収できないことが挙げられていた。このことを打開するために、製品に違法コピーを防ぐためのプロテクトをかける方法や、頻繁なバージョン・アップを繰り返す方法などが採用されていた。これはユーザー側から見ると、実用的な利用にとって必ずしも快適な環境ではない。

しかし、国際協力の現場で目の当たりにすることは、発展途上国においてプロテクトをはずした CAD や GIS 製品が低価格で販売され、アンダーグランドのマーケットが形成されている、という事実であった。経済水準の低い途上国では、メーカーが販売収益を上げることが見込まれないために、知的財産権を守るためのコストをかけることを断念し、違法コピーの横行を黙認しているように見受けられた。このため、例えば学生における CAD や GIS ソフトの普及率は途上国においてきわめて高いという逆説的な現象が生じている。新興国が経済発展を遂げて、正規ユーザーとしてソフトウェアを購入できるようになった次の世代においては、IT が急速に社会的に普及する下地は十分に形成されている。そのための準備としては、違法コピーの放置はむしろ布石ともなっているように見える。

（4）民間商用ソフトウェアとの関係

開発に着手した 1993 年当初、いくつかの CAD メーカーなどから営業を目的とする訪問があった。メーカー側の意図は、研究所におけるシステム導入であった。当時、建設省中国技術事務所においては、一つのシステムを導入し、これを運用して管内の現場における景観検討を行う試みも行われていた。しかしながら、本システムを開発するに当たって想定した業務形態は、個々の開発現場において、担当者あるいは業務委託を受けたコンサルタントが日常的に景観検討を行うことにあった。民間で開発されている CAD ソフトウェアのライセンスを買い上げた上でオープンソース化し、これに基づいて機能開発を進める方法も検討したことがある。旧バージョンであれば応じても良いという社もあったが、ライセンスの価格において折り合わなかった。

知的財産のプロテクトを前提としつつ継続的収益を目指す民間ビジネスと、公開を原則とする公共財の領域の衝突や矛盾を回避し、連携を図るためには、官民共同研究によるデ

ータ共有の仕組みが有効である。初期の開発が概ね終了し、バージョン 2.01 を公開開始した 1996 年時点で、官民共同研究を開始した。公募の結果、空撮写真から現況市街地データを作成する技術と、地上写真からテクスチャ付建物三次元データを作成する技術が対象となった（詳細は文献 16 参照）。いずれにおいても、考え方として、民側の商用ソフトウェア（有償）や測量サービスないし測量結果としてのデータ（有償）と、官側のオープンソースのフリーウェアの中間に、共同研究成果としてのコンバータ（ソースコードを公開しないフリーウェア）を置く、という方法を採用している。

民間のソフトウェア開発は、知的財産のプロテクトを確保しつつ最先端の技術を追求する方向を志向する。そのことが弊害としてデジタル情報の恒久的保存活用の困難をもたらしているとすれば、公的なＩＴ化はユーザーとしての立場のみならず、研究開発において競合を避け、調和・協調を図りながらも、上記のような欠陥を補完するような役割を果たすことが必要である。既に商品としての寿命を終えたような製品で、基本的なアルゴリズムを有するものについては、今後検討されても良い。

（5）外注の効率性：プログラマとの緊張関係

公共財としてのソフトウェアを開発する方法（業務形態）として、いくつかの異なる体制が考えられる。事業官庁においてインハウスで開発を行う場合には、成果としてのソフトウェアの販売収益が目的ではなく、組織内部の時間節約・効率向上と、それを通してもたらされる社会的便益が上位目的である。

実際には、ソフトウェア開発の多くの部分は、民間企業に対する外注を通して行われる。発注者側においては、公共工事と同様に、一定の工期・予算の中で要求性能を実現するプログラムを完成させることを期待しており、不完全な部分は受注者の責任において修正することが当然と考えられる。しかしながら、これまでに存在していなかったような新しい技術を投入するソフトウェア開発は、研究開発の委託のような性格を帯びる。最終成果物がどのような機能を果たし、どのように使えるのか、操作体験のない発注者側において、契約時点では必ずしも明確に認識されているわけではない。このため、納期に近い打ち合わせ段階でデモンストレーションが行われた段階で、初めて発注者側に明確な使い方に関するイメージが形成され、それが追加要望のような形でリクエストされるような状況が生じうる。一方、受注者の側では、当面期待されているソフトウェアの動作（それは打ち合わせの結果修正を要求される可能性がある）を実現することが短期的目的となるため、基本的なデータの整合性の確保（とりわけ使用済みのメモリブロックの解放処理等）といった、外見では見えない部分の完成は後回しになる傾向が生じる。納期に近づいてから、要求仕様に関する認識のずれが明確化し、修正が行われる場合には、納品されるソフトウェアは、「とりあえず動いている」程度の完成度レベルのものとなる。

このようなソフトウェア資産に関して、安定した確実な動作を行うように改善していくためには、相応の手間とコストがかかり、受注者の品質保証責任の範囲内だけで実現することには限界がある。しかしながら、既に納品され、外見上の動作は実現している作品の

内的な改善・修正（外見上は大きな変化を伴わない）は、別業務としては成立しにくい。このため、継続的な業務発注において、改善（機能の追加等）を一義的な目的とする新たな業務が、通常同一の業者に対して行われ、その中で前回業務の未完成部分に対する修正も行われることが期待される。しかしながら、同一ソースコードに対して手を加え、完成度を高める作業と機能を追加する作業（完成度を低める方向）を両立させることは容易ではない。動作が不安定な既存部分に見切りを付け、ほぼ同一の目的のための別のデータ形式や処理ルーチンが追加されるならば、更に状況は複雑化する。

景観シミュレーション・システムの場合には、結果的にこの部分の作業のほとんどはインハウスで行われてきた。Ver.2.09 に向けた統合化においては、完成度を追求する基幹部分と、機能追加のためのプラグイン DLL を分離する、というアプローチで整理した。

ある段階から機能追加を民間へバトンタッチしていく方法がありうる。景観シミュレーション・システムの場合には、現在がその時期ではないかと考えている。

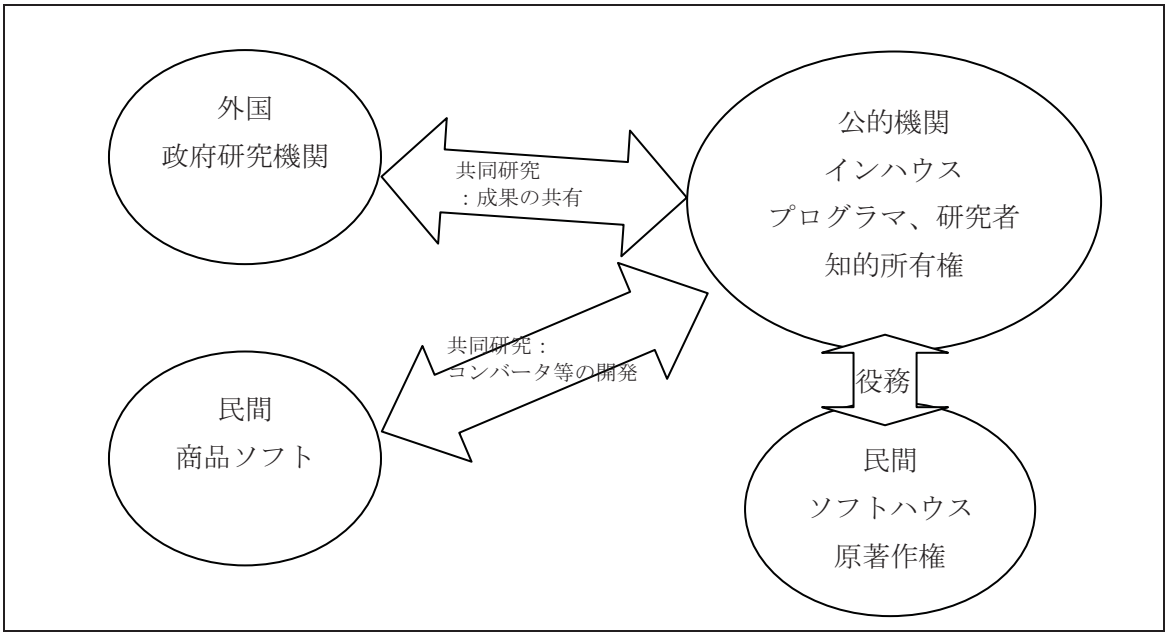


図１６－１：景観シミュレーション・システム開発における連携

（6）国際協力

国際協力の入り口として、ソフトウェアの翻訳を行う必要がある。ライセンスが制約された商用ソフトウェアをベースとして公的機関の間で国際協力を行うことは困難を伴う。一方、オープンソースであることにより、制約条件が無い。公開ＷＥＢサイトからダウンロードすることにより、任意で参照することは可能である。しかしながら、国民が納めた税金を財源として開発された知的財産を外国に対して無償で提供することは不公平と考えられたため、日韓科学技術協力協定に基づく共同研究を実施した際には、研究所間で Record of Discussions を取り交わし、相手国側が同一技術の開発を重複して繰り返すことを避ける代償として、提供した技術の上に開発された成果を共有（還元）することを謳った。ソースコードをベースとして実際に実施した初期の翻訳移植作業においては、開発環境の違

いなどにより発生した障害（開発環境のバグや、ソースコード中の潜在バグの顕在化など）に対処する必要があり、言語専門家による単純なリソースやヘルプファイルの翻訳移植だけでは済まず、両国のプログラマによる協力が必要であることが経験的に分かった。今回リリースする多言語版においては、このような問題は解消している。

１６－３．ユーザーとの共進化

（1）現場の担当者

ワードプロセッサのように、毎日の業務に使用するものではない。工業高校程度の教育を受けた職員が、月に１度程度の頻度で、説明会の前１週間の間操作する、という使用が行われた。このような使用環境においては、基本的なメニュー構成を変更せず、追加機能は奥の方に作りこんでいく戦略が適切であると考えられた。民間の商品ソフトウェアは、プロフェッショナルなユーザーが日常的に使用することを想定して、ショートカット等の能率向上のための機能がふんだんに盛り込まれているが、これらは偶に使用する程度のユーザーにとっては、わかりにくく忘れ易い。また、日常的に使用するワードプロセッサ等においても、バージョン・アップに際して、追加された新しい機能が前面に押し出され、ルック・アンド・フィールが大幅に改定される場合があり、慣れるために時間を要する場合がある。

（2）学生

筑波大学では、MiniCAD を教材として学生の教育を行っていた。彼らに景観シミュレータを用いて作業を依頼すると、操作性に関して、教育を受けたシステムとの違いにおいて意見を述べる場合が多かった。最初に習得したシステムが重要であることがわかる。業務用のソフトウェアに習熟している学生等の場合には、景観シミュレータのメニューから明示的に機能を選択する方法は、プロ仕様のソフトウェアにおけるショートカットやコントロールキーによる迅速な操作と比較して手間がかかるという印象も与えるようである。

（3）子供

つくば科学フェスティバル、土木の日などの公開の機会を積極的に活用して、操作を体験して頂いた。このことを実現するためには、基本的な操作（インターフェース）が単純である必要がある。最新の例では、敷地を自由形状として作成した上に、データベースから選択した住宅を配置し、そこに様々な遊具や樹木などを点景として配置するような操作が好まれる。このような操作に関しては、指導員（職員や高校生など）に対して事前に 30 分程度のレクチャーを行い、直ちに来訪者との対応に入っていただくことが可能であり、助言しながら子供に選択肢を示して操作を進めてもらったり、習熟に応じて自分で操作してもらったりすることができる。基本的な視点移動や、配置等の機能選択方法が単純であることが重要な条件である。

（4）操作性に関する慣習

景観シミュレータの操作性は、「初めてこの種のシステムに触れる」ユーザーを想定して

開発された。メイン画面の上に、ファイル、編集、表示、ツール等の基本的なメニューを示す点、視点移動は機能を明記した操作ボタンによる点、モデリング操作は、まず画面でオブジェクトを選択してから、それに対する操作を選ぶ点などである。この点は、プロフェッショナル向けの本格的な CAD や GIS が最初に一定のトレーニングを必要とする状況とは異なっている。展示会や体験教室においても、高度な処理を含まない基本的な編集操作に限定したコンテンツと体験内容に限定するならば、高校生アルバイトなどに約３０分間の事前トレーニングを施すことにより、指導員として直ちに来場する小中学生に操作指導することが可能である。

一方、最近の土木・建築系学生は、それぞれの大学で特定の製品による教育を受けている。彼らの感想（改善提案）は、概ね既に習熟しているソフトウェアとの対比として述べられることが多い。このことは、多言語版を海外で使用する場合でも同様である。

（５）操作における陳腐性の価値

ソフトウェア商品が新たなユーザーを獲得するために、その独自性・新奇性を強調する場合がしばしば見られる。このことはバージョン・アップに際しても生じうる。一方、既存製品のユーザーにとっては、操作手順の変更は戸惑いの原因となる。とりわけ道具としてソフトウェアを、毎日ではないが一定の頻度で使用しているユーザーにとっては、度々再学習が要求されることは負担である。景観検討や現場説明の業務であれば、数ヶ月に一度、数日から１週間程度の作業時間を投入する場合が多いようである。

一例として、位置の指定や形状生成における画面クリックと座標数値入力を考えてみる。形状生成において、概略の図形を生成するだけであれば、画面上のポイントをマウス・クリックすることにより座標指定する方法が能率的で、素早くデータを構築することができる。ソフトウェアの性能のデモンストレーションにおいては、効果がある。一方、既に設計図が存在するような場合には、点の位置を座標値により指定する方法が速い。そこで、多くのダイアログでは、このいずれの方法でも点の位置が指定できるようにした。画面クリックで点の位置を指定した場合には、座標値を表示するエディット・ボックスに座標値が表示される。一方、座標値を入力し、マウス・カーソルを他の場所でクリックすることにより、画面表示もその座標値に対応した位置に移動する。

マウス操作で指定されたパラメータを直ちに形状表示に反映させる方法は、全体のデータ量が小さい場合には有効であるが、データが大きくなってくると、反応の遅さとして認識されるようになることにも留意する必要がある。

このような別の一例として、カラー編集画面がある。初期のバージョンにおいては、マテリアル編集ダイアログにおいてスライドバーを動かしてユーザーが変更した新たな色彩を、同ダイアログ内のサンプル画像に反映させるだけではなく、メイン画面で選択された編集対象のオブジェクトにも同時に反映させる動作としていた。しかし、一つの市街地全体などが編集対象となっているような場合、メイン画面の再描画に時間を要するために、スライドバーがマウス操作に追随して軽快に動かなくなり、非常に重い動作となる。この

問題を解決するために、ユーザーがスライドバーを動かした後、一定時間スライドバーが動かなかった場合に初めてメイン画面にその結果を反映させるという処理を追加した結果、軽快な操作を確保することができるようになった。

公共財として開発するフリーウェアにおいては、バージョン改訂に際して、操作性を大きく変更することによって、改良を強調表現する必要は一切なかった。このことは継続的なユーザーのニーズと一致していると考えている。

（6）バグ発見の手がかりとしての体験教室等の効用

景観シミュレータを繰り返し操作する者は、開発担当者も含め、無意識にリスクのある動作を避ける傾向がある。また、開発において想定された操作手順に忠実に従う傾向がある。これに対して、初めてシステムを操作する展示会来訪者や、初めて操作する非常勤職員などは、開発者が予想していなかった操作を行うことがしばしばある。その際の戸惑いや、理解の障害を認識することは、マニュアルやヘルプを改善する手がかりとなる。また、バグによるシステムのフリーズやプログラムの異常終了の経験は、そのようなリスクを伴う処理を無意識に避けるように学習させることとなり、バグの温存につながり易い。そのような意味で、初めてシステムに触れるような人々を対象とした体験教室等への参加は意味があると考えている。また、そこで生じた事柄を記録し、以後の改善に役立てていくことも有効である。

１６－４．データ形式

本システムの開発に着手した時点では、既存のデータ形式としてはDXF形式が存在しており、二次元CADにおける流通（例えば、製図に使用する衛生陶器製品の形状データのメーカーから設計事務所への提供）に使用されていた。しかしながら、景観検討に必要となるテクスチャやマテリアルを付したオブジェクトの記述には不十分であった。CADソフトウェアは当時、プロッターを駆動するために、製図における線を記述することが基本となっていた。複数の面から構成されるオブジェクト単位での操作を表現するデータ形式とはなっていなかった。景観シミュレータにおいても、「線」の記述は可能であるが、実質的にこの部分を用いることはまれである。一方、CGでは、Silicon Graphics社が中心となって、テクスチャを張り込んだオブジェクトの出力表示処理を標準化するOpenGLを提示し、高価なグラフィック・ワークステーションのみならずローコストなPCでグラフィックス・ソフトを開発する環境が提示され、CADのデータ形式を拡張して三次元CGと一体化することが課題となっていたが、そのようなデータ形式がまだ存在していなかった。

CADデータの流通は長年の課題であったが、民間のベンダーを集めたフォーラムにより共通の交換形式を定め合意に至ることは、当時難航していた。公共的に使用されるデータ形式を定める場合、特定の民間商用ソフトウェアで用いられている形式をそのまま採用することはできない。必要最小限の独自のデータ形式を定め、公開した上で、実用に向けては、普及しているソフトウェアとの互換を図るコンバータを作成することで対応すること

が適当と考えた。

当時、国土地理院において、「建設技術評価制度」を運用して、ステレオ空中写真解析技術の評価検定を行っていた。その際に、各社に対して、指定したフォーマットでのデータの提出を求めていた。このため、技術を有する各ベンダーは、このデータ形式で出力する機能（コンバータ）を作成し、評価を受けた。従って、景観シミュレーションのためのコンバータを作成するためには、この共通の評価形式をハブとして、これに対応することで大幅に手間を省くことができる。このことが参考となった。このような目的のためのデータ形式は、パフォーマンスや拡張性よりもむしろ、単純明快性が求められる。例えば、公共事業における設計データの納品のために目的を限定した単純明快なデータ形式が定められれば、三次元オブジェクトに関するコンバータ作成の労力は大幅に軽減されると考えられる。

民間各社は、デファクト・スタンダードとすることが利益拡大につながる、という動機を有する反面、一定のデータ形式を固定的に定めることは、日進月歩の技術革新の中で、新たな状況に移行することの障害となる恐れがある。実際に、GIS の SHAPE 形式や CAD の DXF 形式は、論理的に一貫しない形式の拡張や、ヘッダー部分で識別した上で、異なる構造のデータを並存させるような仕様となっている。

景観シミュレータの開発の目的はこのようなデータ形式を定めるということには無い。そこで、将来のそのようなデータ形式の成立に期待しつつ、当面する景観検討ニーズに対応できる必要かつ十分なデータ形式を定め、対応することとした。

その基本的な考え方は以下の通りであった。

・地物を構成する単位となるオブジェクトを、表面の集合により表現する（グループ）。

・オブジェクトを階層構造により表現し、上位のオブジェクトにリンクする際に、相対的な変位（平行移動、回転、スケール）を定義できるようにする。

・リンクを複数設定することにより、群として線的・面的な配置を表現する（例：街路樹や山林等）。これによりファイル・サイズを節約する。

・XYZ 座標から成る三次元の位置情報を有する座標(COORD)を定義する。

・頂点には座標の他に、テクスチャ座標、法線ベクトル、カラーを記述できるようにする(VERTEX)

・面(FACE)及び線(LINE)を、頂点列として定義し、その順番で面の表裏を決める

・面の表面光学特性として、カラーとテクスチャを直接記述し編集することとする。

・グループや面に対して、別途定義されたマテリアルを設定できるようにする。

・マテリアルの定義に、経年変化が記述できるようにする他、カラーやテクスチャの他に、鏡面反射率や輝度等も定義できるようにする。将来材料のＣＧ表現技術が高度化した場合であっても、マテリアルを用いた地物データであれば、マテリアル定義を高度化することにより、地物データを変更することなく対応可能である。

１６－５．システムの性能

（1）処理時間

Sim.exe の処理時間には、CPU 性能に依存する部分と、グラフィックス性能に依存する部分がある。具体的には、ファイルをロードするのに要する時間は前者に依存し、視点移動は後者に依存する。外部関数（パラメトリックな部品）の処理には一時的なファイルを生成するため、これに加えて、ディスク・アクセス時間が影響する。ファイルのロードに際しては、頂点や面やグループの名称などをテーブルで管理するため、保存よりもはるかに処理時間がかかる。頂点や面を定義するために使用する名称は、GROUP_FACE コマンドにより確定するため、それ以後に同じ名称を用いて再定義しても過去に定義した頂点や面に影響を及ぼさない。このため、少ない数の名称を繰り返し使用することにより、テーブル管理に要する時間を節約することができる。最適化保存では、このような処理を行っているため、保存に要する時間は若干長くなるが、次にロードする際に要する時間を大幅に軽減することができる。

（2）ファイル・サイズ

モデルを記述する LSS-G ファイルは、以下の方法により、ファイル・サイズを節約している。

①繰り返し配置されるオブジェクトに関しては、一つのグループだけを使用し、それらの一群を定義する親グループとの間に、配置する個数だけのリンクを定義し、それぞれのリンクに位置情報を示すマトリクスを記述することで、ファイル・サイズを小さくしている。このことは反面、複数個配置されたオブジェクトを編集する（例えばカラーを変更する）場合の影響範囲をわかりにくくしている。

②パラメトリックな部品に関しては、部品名称とパラメータだけをファイル保存し、実際に表示する際に必要となる頂点や面などの情報は、ロードされた状態でメモリ上にのみ展開する。このことの効果は、たとえば滑らかに見せるために多くの面に分割する曲面をもつ要素の定義において大変効果がある。

（3）データ規模と作業量

三次元データのサイズは、しばしば面の数（ポリゴン数）によって記述される。これは表示処理に要する速度に関係している。しかしながら、データ構築に際して要する作業量（手間、従って労賃に関連）は、モデルを記述する LSS-G ファイルのデータ規模と、必ずしも直線的に相関するわけではない。例えば、曲面を多く有する部品を多数配置した場合には、ポリゴン数は急速に増大する。

これに対して、グループ数は、形状生成や配置の操作回数に概ね比例して増大するため、あるデータを作成するために要した手間を評価するためによい指標となる。実際に、2001 年度に市街地のデータを計画案に基づいて、学生アルバイトの作業により作成した際には、作業時間と、作成されたデータのグループ数に関する記録を行った。この分析に基づいて、１グループ当たりの時間、従って作業単価を設定することができた。

グループ数は、sim.exe の報告書作成機能（メイン画面のメニュー［ファイル］［報告書執筆］により表示することができる。

（4）ハードウェアの進化とアルゴリスムの改善

景観シミュレータが処理対象とする、ないし処理可能なデータの規模は、現場で使われている PC のメモリ量や CPU 速度の向上に伴い、大幅に向上してきた。同一のハードウェア環境において、データ作成方法や処理アルゴリスムの工夫により速度や容量を改善する方法は試みられるが、例えば地形を構成する多くの三角形群を擬似的な一つの巨大なポリゴンとして OpenGL の表示系に送出処理する方法などは、地形編集操作も含めると複雑化につながる危険性がある。基本的なデータ構造は、単純明快な形を維持することが適切と考えた。特に速度が要求されるドライブシミュレータや、ゲームなどにおいては、見えない側の壁面を省略した建物等を書割的に用いる場合がしばしば見られる。

2001 年度に作成した全国１５ヶ所の現場のデータ作成に際しては、主に学生アルバイトにより、市街地再開発計画等の三次元データ化の作業を行った。その際に、学生に処理能力の高いマシンを与えると、植栽等の配置により、そのマシンで表示が快適に行える限界までマニアックに複雑なデータを作成する傾向が見られた。そのような大きなデータは、一般市民が使用しているパソコンに WEB 配信しても軽快に動かない可能性が高い。このため、研究室にある旧式の、処理速度の遅いマシンでデータ作成を行わせるように改めた。このように、目的に適した「複雑さ」のデータを作成するためには、あえて処理能力を落とした環境で制作作業を行う、ということも一つの方法である。

１６－６．情報の持続性

景観シミュレーションは、一時的な目的で行われる場合もありうる。作業を行い、検討終了後はデータを破棄して、別の作業に移行するような用法である。しかし、一度作成したデータを、部品として再利用したり、以後の施工・維持管理に向けて継続的に活用したりすることが考えられる。しかし、現実には、そのことを阻むいくつかの障害が存在する。

①物理的に永久保存することの困難

デジタルデータとして保存するためには、具体的な記憶媒体の上に保存する必要がある。メンテナンスなしで永続的な保存を保証するようなメディアは今のところ存在しない。

②物的媒体の陳腐化

データをある媒体に記録しても、その媒体からデータを読み出すための装置が無くなれば、利用が困難となる。例えば、媒体そのものは完全な形で保存されていたとしても、５インチ・フロッピーディスク媒体に保存されている CAD データを読み出そうとすると、ドライブが稀少となっている。

筆者の研究室においても、８インチ、５インチのフロッピーディスク約 400 枚に記録されたデータを、1999 年に CD-R に集約する作業を行った。当時 3.5 インチ FDD が主流となっていたため、中古の５インチドライブを、読み取り不能になる都度別のドライブに交

換しながら何とか変換した。その CD-R も 10 年を経過し、一部は読み取りエラーを生じるようになっている。

ハードディスクに集約したデータも、最近のマザーボードに更新すると、IDE のインターフェースが接続できない場合がある。

この困難を避けるためには、時代の流れに沿って、メディア変換を行わなければならない。つまりは、運用システムという社会的な仕組みが必要となる。

③データ・フォーマットの変化

データそのものが読み出し可能であったとしても、古い形式の DOS（例えば N88BASIC や CP･M）やファイル・フォーマットで記録されたデータは、処理が難しくなる。1980 年代後半に途上国で広く使われていた Word Star の文書なども、当時テキスト形式で保存していないと、利用がかなり困難である。同じ系列の CAD システム等においても、初期のバージョンのデータがそのまま利用できる保障はない。

二次元画像の分野では、JPEG 形式が広く使われている。データ形式が公開され、入出力のライブラリも、ソースコードとして自由に利用することができ、景観シミュレータにおいても活用している。この仕様を改良しようという計画も聞かれない。データ・フォーマットのあるべき姿を示す一例と言えよう。これに対して、三次元データ形式はまだ定番が成立しておらず、引き続き模索段階にあるように見える。

④OS の変化とアプリケーション

OS のバージョン・アップが行われると、古いアプリケーションがそのままでは正常に動作しない場合が生じる。第 1 章で論じたように、その原因の多くは潜在バグの顕在化である可能性が高いと考えられるが、ソースコードが入手できない場合には、対応することができない。商用アプリケーションの場合には、ファイルを開くことができても、グラフィックな表現において微妙に位置ずれが生じるといった障害はしばしば見られる。

アプリケーション自体が廃止となった場合には、データを処理するためには、過去のアプリケーションが動作する OS 環境を用意しなければならない。

凡そ、以上のような状況により、作成されたデータは、日常的に使い続けられるものでない場合には、いずれ使用不可能となる可能性が高い。

１６－７．ソフトウェアの完成の条件

ＩＴの技術開発は、当初のアイデア・構想から出発し、一通りの動作を実現する製品に到達した段階で、また外注においては、初期の仕様を満足する成果品が動作するようになった段階で、一定の完成といえる。この段階を第一の完成とすると、そこにいたる過程では、集中的な予算・マンパワーの投入が行われる。

しかしながら、実際の業務の現場に投入し、安心して利用できるツールとなることを第二の完成とするならば、そのためには、第一の完成までに投入したものと同等以上の時間とコストが必要である。この過程は必ずしも労働集約的に行われることが効果的ではない。

潜在的なバグを、一見正常に動作している環境下で、ソースコードの点検だけから発見することは容易ではない。建築材料に喩えて言えば、耐久性試験のような性格の取り組みである。OS や開発環境の変化、マシン性能の向上とデータ規模の拡大に対応することの中で、時間をかけて達成できるプロセスである。

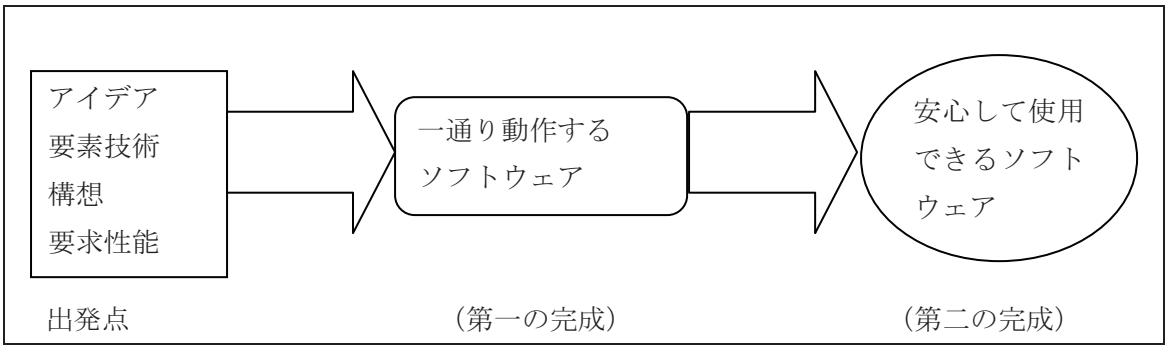


図１６－２：システムの完成に至る二つの工程

建築工事に喩えるならば、第一の完成は上棟段階であり、大勢の職人や大量の資材が動く。以後、造作工事に入ると、現場で働く大工は少数の熟練者となり、外見上は少しずつしか工事は進んでいないように見えるが、内部では細かな神経を注いだ作業が行われている。入居後の不測の雨漏りへの対応などもこのプロセスには含まれるであろう。ＩＴがドッグイヤーで進展する状況下では、この第二の完成に至る前に、新たな開発が開始されるような場合も多いと考えられる。

景観シミュレーション・システムの基幹部分に関しては、この後段の第二の完成に向けた具体的な作業として、以下のような事柄が進行した。

① データ処理における可逆性の確保

データの構築部分が完成すれば、当初のアイデア・構想の正しさは証明され、ソフトウェアの一通りのデモンストレーション等は可能である。しかし、削除処理などに伴い、構築したデータに伴うメモリブロックの解放等の処理において可逆性が確保されていないと、長時間・長期にわたる操作に伴い障害が生じる。

② 潜在バグの除去

スタック上の変数や解放済みのメモリ･ブロックへのポインタが使い続けられることによる障害など、直ちにエラーとはならずに、原因からしばらく経過してから、時に顕在化するようなバグは、異なるＯＳや開発環境への載せ替えなど、長期的なメンテナンスを通じて初めて発見される場合がある。

③ 想定外のデータへの対応

三次元データには、幾何学的にありえない形状に対応したものも含め、様々なデータが生じうる。それらに対して、スキップし、エラー・ログを作成し、あるいは自動的に回復処理を行うなどの処理を実現するためには、正常なデータを通常処理する以上のステップ数を必要とする場合がある。これも、様々な用途のために実際にシステムを活用したり、ファイル・コンバーターによりインポートした様々なデータを処理したりする

ことを通じて鍛えられる部分である。とりわけ、外注により開発し、納品された状態のソフトウェアは、小さなサンプル・データによってのみ動作が確認されている場合が少なくない。発注者側において、なるべく間を措かずに、実際のデータを用いて動作テストすることもまた有効である。

④ 想定外のオペレーションへの対応

システムに初めて触れるユーザーの操作は、様々でありうる。マニュアルに書かれていない様々な「行儀の悪い」操作に対しても、システム・ダウンに陥ることなく、適切なエラーメッセージを出して対応するように改良することは、幾何学的・物理的対象物に関するデータの処理というよりは、むしろ人間工学的な性格を帯びている。そして完成されたシステムを記述するソースコードのうちのかなりのステップ数は、このようなエラー・コーディングのために費やされることとなる。

当初の構想と開発目標においては、設計対象となる土木建築施設やその周辺環境の特性とモデリング方法、幾何学的演算処理、コンピュータ環境（ハード・OS・開発環境）が主たる要素であったが、実用化（第二段階の完成）に向けては、実際の運用に際しての、ヒューマン・エラーへの対応など、むしろ人間工学的な課題への対応が求められ、そのためのエラー・コーディングが必要となった。

どのように操作し、どのようにメッセージを出すかというインターフェースに関する標準化が進み、言葉や生活慣習と同様の社会的資産として共有されアプリケーション開発の標準的条件となるならば、今後同様のシステム開発を行うプログラマの負担は軽減されるであろう。

⑤ OS や開発環境の変化を超えた価値

OS や開発環境の変化は、ゆるやかなマルチ･プラットフォーム対応が時系列で展開したものと見ることもできる。その中で、結果的に制作されたソースコード群の価値で普遍性の高いコアの部分は、ライブラリ関数の中にあると考えられる。一方、前項で触れたユーザーへの人間的対応に関するノウハウ等は、OS に強く依存したリソースファイルやダイアログ・ハンドラの中に蓄積されている。しかしこれらは直ちに別のシステムに載せかえることができるソフトウェア資産ではなく、その背後にある共通で変わらない、より抽象的なノウハウやロジックに価値があると見るべきだろう。

⑥ 将来の拡張方法と、完成部分の分離

システムに対する機能的要求には終わりはない。とりわけ、三次元データ処理はまだ流動的な分野であり、多くのアイデアやニーズが存在している。これらに対応し続ける限り、完成は永久に訪れない。このため、Ver.2.09 においては、今後原則的に手を加えない基幹部分と、それとは様々なリスクを遮断した形で柔軟に新たな機能追加を行うためのプラグインのアーキテクチャにとりまとめた。

⑦ アーカイブ化と記録・解説資料の整備

同じような開発が繰り返され、その中で同じような試行錯誤が繰り返されることは無

駄である。「最先端の技術」の名の下に、社会的に共有されるべき知識・情報が不必要に隠匿されることもまた社会的には損失である。このためには、すでに当たり前となった技術を整理し、再利用できる形で解説・公開する営みが一方で必要である。

1986 年頃、建設省においてニュー・フロンティアに関する関心が高まり、建築研究所において、宇宙開発に関する勉強を少し行った時期がある。それに参加したときに聞いた、「安全性・信頼性の高い宇宙船を実現するためには、使用する部品や要素技術には、古くからの枯れたものを使う」という、当時の国際宇宙ステーション開発担当者の言葉が印象に残っている。

付録 A ソースコード一覧

	A	B	C	D
1		ソースコード名(c,cpp)	行数	.h(同一名略)
2	SIM	antiarea.cpp	304	
3		AviComposeWnd.cpp	587	
4		BMP2DIB.cpp	57	
5		cmd2sndf.c	80	
6		cmovie.c	2017	
7		CNVCOL.C	215	
8		COLORSET.cpp	1003	
9		colsashdlg.cpp	143	
10		Common.c	2133	
11		COPY.c	596	
12		cubedlg.cpp	135	
13		Dataope.c	3487	
14		dib32h		
15		dibapi.h		
16		dibapi2.h		
17		dispkein.cpp	150	
18		drawfrm.cpp	1501	
19		dxfout.c	209	
20		editcopy.cpp	50	
21		editlig2.cpp	247	
22		editligh.cpp	700	
23		editmat2.cpp	1345	
24		editmate.cpp	1262	
25		editmove.cpp	801	
26		editshit.cpp	593	
27		edittex2.cpp	1203	
28		edittexl.cpp	268	
29		edittext.cpp	1877	
30		elembloc.cpp	140	
31		elembrid.cpp	351	
32		element.c	73	
33		elemgras.cpp	199	
34		elemmina.cpp	155	
35		elemnori.cpp	111	
36		elemrive.cpp	219	
37		elemroad.cpp	235	
38		elemste2.cpp	200	
39		elemstee.cpp	445	
40		extwnd.cpp	1210	
41		faceanal.cpp	152	
42		faceinfo.cpp	287	
43		fopen_.c	592	
44		framedc.h		
45		fselect.c	23	
46		genshi.c	456	
47		gjm.cpp	59	
48		glos.h		
49		grid.cpp	194	
50		gydlg.cpp	897	
51		haichidt.cpp	449	
52		haichipr.cpp	438	
53		haichiwn.cpp	3158	
54		haiseldl.cpp	806	
55		his.c	1165	

	A	B	C	D
1		ソースコード名(c,cpp)	行数	.h(同一名略)
56		hist_read.c	250	
57		hst_write.c	256	
58		image.c	489	
59		info.cpp	448	
60		kakunin.cpp	63	
61		kamera2.c	579	
62		kashiwnd.cpp	1278	
63		keirownd.cpp	2227	
64		kelihatan.cpp	80	
65		kinou.txt	122	
66		knoricol.cpp	139	
67		kougen.cpp	637	
68		kshocolw.cpp	139	
69		langset.h		
70		lightlst.cpp	125	
71		lightuni.cpp	167	
72		list.c	13	
73		locuswnd.cpp	845	
74		mainfrm.cpp	7457	
75		mate2dlg.cpp	136	
76		matfrm.cpp	135	
77		matlist.cpp	330	
78		memo.txt	27	
79		mnoricol.cpp	136	
80		moviedlg.cpp	2039	
81		mshocolw.cpp	136	
82		MyAvi.cpp	656	
83		Mydlgbar.cpp	160	
84		MyImage.cpp	149	
85		net_proc_sgi.c	521	
86		nohelp.cpp	74	
87		nori2dlg.cpp	120	
88		noridlg.cpp	504	
89		norimat.cpp	923	
90		norimatl.cpp	237	
91		noritexl.cpp	217	
92		noriwnd.cpp	1197	
93		opedlg.cpp	533	
94		open.c	356	
95		orthovw.cpp	3243	
96		pembantu.c	75	
97		pixel.c	123	
98		planewnd.cpp	2385	
99		pointmovewnd.cpp	940	
100		primcone.cpp	416	
101		primcube.cpp	218	
102		primcyli.cpp	305	
103		primface.cpp	159	
104		primflco.cpp	343	
105		primflcy.cpp	313	
106		primline.cpp	489	
107		primsphe.cpp	238	
108		print.c	777	
109		rd_client.c	297	

	A	B	C	D
1		ソースコード名(c,cpp)	行数	.h(同一名略)
110		rd_getcmd.c	322	
111		rd_server.c	547	
112		rdh_com.c	222	
113		rdh_msg_cmd.c	2008	
114		rdh_msg_controll.c	94	
115		rdh_AnlayseCmd.cpp	3574	
116		readme.txt	107	
117		remotedemohis.cpp	135	
118		resource.h	879	
119		roadcol.cpp	136	
120		roaddan.cpp	286	
121		roadwnd.cpp	1210	
122		s_haichi.h		
123		savedlg.cpp	151	
124		savemode.cpp	132	
125		scenelst.cpp	881	
126		shitenwn.cpp	1498	
127		shutterd.cpp	312	
128		sim.cpp	394	
129		sim.rc	0	
130		simdoc.cpp	95	
131		simview.cpp	288	
132		solidanal.cpp	115	
133		spheredlg.cpp	137	
134		splash.cpp	150	
135		stdafx.cpp	12	
136		sweep1.cpp	236	
137		sweep2.cpp	134	
138		textfrm.cpp	121	
139		textmap.cpp	536	
140		timer.h		
141		topocutwnd.cpp	862	
142		topoeditwnd.cpp	735	
143		utility.c	369	
144		vanal.cpp	229	
145		vrmlout.c	629	
146		vwparam.cpp	75	
147		wg3.c	196	
148		wg3swal.c	224	
149			85229	
150				
151	YUU	childfrm.cpp	303	
152		childnam.cpp	72	
153		database.cpp	778	
154		dbdata.c	2286	
155		dbms.c	1731	
156		dbread.c	1543	
157		histdlg.cpp	189	
158		imadlg.cpp	292	
159		mainfrm.cpp	247	
160		mojidlg.cpp	115	
161		pixelc.	123	
162		scrollli.cpp	300	
163		seldlg.cpp	149	

	A	B	C	D
1		ソースコード名(c,cpp)	行数	.h(同一名略)
164		stdafx.cpp	538	
165		utility.c	369	
166		w3dbsub.c	223	
167		w3ntsub.c	624	
168		yuu.cpp	164	
169		yuu.rc	0	
170		yuudoc.cpp	476	
171		yuuvw.cpp	1030	
172			11552	
173				
174	KOU	childfrm.cpp	303	
175		childnam.cpp	72	
176		database.cpp	778	
177		dbdata.c	2286	
178		dbms.c	1731	
179		dbread.c	1543	
180		histdlg.cpp	189	
181		imadlg.cpp	292	
182		mainfrm.cpp	247	
183		mojidlg.cpp	115	
184		pixelc.	123	
185		scrollli.cpp	300	
186		seldlg.cpp	149	
187		stdafx.cpp	538	
188		utility.c	369	
189		w3dbsub.c	223	
190		w3ntsub.c	624	
191		kou.cpp	164	
192		kou.rc	0	
193		koudoc.cpp	476	
194		kouvw.cpp	1030	
195			11552	
196				
197	ZAI	childfrm.cpp	303	
198		childnam.cpp	72	
199		database.cpp	778	
200		dbdata.c	2286	
201		dbms.c	1731	
202		dbread.c	1543	
203		histdlg.cpp	189	
204		imadlg.cpp	292	
205		mainfrm.cpp	247	
206		mojidlg.cpp	115	
207		pixelc.	123	
208		scrollli.cpp	300	
209		seldlg.cpp	149	
210		stdafx.cpp	538	
211		utility.c	369	
212		w3dbsub.c	223	
213		w3ntsub.c	624	
214		zai.cpp	164	
215		zai.rc	0	
216		zaidoc.cpp	476	
217		zaivw.cpp	1030	

	A	B	C	D
1		ソースコード名(c,cpp)	行数	.h(同一名略)
218			11552	
219				
220	DBEDITOR	classdlg.cpp	274	
221		clsmenu.cpp	36	
222		colordlg.cpp	230	coldlg.h
223		editor.c	2021	
224		gazoudlg.cpp	905	
225		histdlg.cpp	68	
226		imagedlg.cpp	235	
227		infodlg.cpp	55	
228		inpcom.cpp	74	
229		inpdlg.cpp	730	
230		keywddlg.cpp	163	
231		long.c	265	
232		longdlg.cpp	264	
233		main.cpp	183	
234		main.rc		
235		maindoc.cpp	125	
236		mainfrm.cpp	695	
237		mainview.cpp	392	
238		matlist.cpp	123	
239		mojidlg.cpp	135	
240		pilih.cpp	90	
241		quesdlg.cpp	94	
242		size.c	161	
243		sizedlg.cpp	172	
244		stdafx.cpp	18	
245		stringdl.cpp	178	
246		timedlg.cpp	239	
247		top.c	186	
248		topdlg.cpp	186	
249		(total)	8297	
250				
251	LIBRARY	69827		
252	dbil	dbdata.c	2286	
253		dbms.c	1735	
254		dbread.c	1573	
255		image.c	636	
256		isave.c	179	
257		(total)	6409	
258				
259	dml	d3dml.c	1460	
260		d3dmalloc.c	966	
261		d3mat.c	386	
262		d3pick.c	501	
263		(total)	3313	
264				
265	drl	glw2.c	187	
266		glw3.c	197	
267		r3drl.c	2205	
268		r3utl.c	158	
269		(total)	2747	
270				
271	env	e3env.c	1515	

	A	B	C	D
1		ソースコード名(c,cpp)	行数	.h(同一名略)
272		(total)	1515	
273				
274	g3drl	g3cepat.c	4455	
275		g3combine.c	143	
276		g3cut.c	166	
277		g3divtri.c	141	
278		g3font.c	152	
279		g3load.c	250	
280		g3move.c	103	
281		g3plane.c	96	
282		g3road.c	1321	
283		g3round.c	204	
284		g3sdl.c	731	
285		g3steel.c	460	
286		g3stencil.c	44	
287		g3tunnel.c	160	
288		g3utl.c	227	
289		wg3.c	197	wg3.h
290		wg3swal.c	224	
291		(total)	9074	
292				
293	imgsgi	close.c	111	imgsgi.h
294		filbuf.c	45	
295		flsbuf.c	47	
296		name.c	24	
297		open.c	356	
298		pix.c	31	
299		rdwr.c	107	
300		rle.c	179	
301		row.c	228	
302		(total)	1128	
303				
304	ip	fopen_.c	592	
305		i3ip.c	573	
306		i3ipbase.c	730	
307		i3ipcall.c	207	
308		i3iperr.c	158	
309		i3ipex~1.c	124	
310		i3iplssg.c	1528	
311		i3iplsss.c	356	
312		i3ipou~1.c	994	
313		(total)	5262	
314				
315	jpeg	ansi2knr.c	697	
316		cderror.h		
317		cdjpeg.c	179	
318		cjpeg.c	618	
319		ckconfig.c	403	
320		djpeg.c	605	
321		example.c	435	
322		jcapimin.c	237	
323		jcapistd.c	162	
324		jccoefct.c	449	
325		jccolor.c	460	

	A	B	C	D
1		ソースコード名(c,cpp)	行数	.h(同一名略)
326		jcdctmgr.c	388	
327		jchuff.c	847	
328		jcinit.c	73	
329		jcmainct.c	294	
330		jcmarker.c	642	
331		jcmaster.c	579	
332		jcomapi.c	95	
333		jconfig.h		
334		jcparam.c	575	
335		jcphuff.c	830	
336		jcprepct.c	355	
337		jcsample.c	520	
338		jctrans.c	372	
339		jdapimin.c	407	
340		jdapistd.c	276	
341		jdatadst.c	152	
342		jdatasrc.c	213	
343		jdcoefct.c	736	
344		jdcolor.c	368	
345				jdct.h
346		jddctmgr.c	270	
347		jdhuff.c	575	
348		jdinput.c	382	
349		jdmainct.c	513	
350		jdmarker.c	1056	
351		jdmaster.c	556	
352		jdmerge.c	401	
353		jdphuff.c	643	
354		jdpostct.c	291	
355		jdsample.c	479	
356		jdtrans.c	123	
357		jerror.c	229	
358		jfdctflt.c	169	
359		jfdctfst.c	225	
360		jfdctint.c	284	
361		jidctflt.c	242	
362		jidctfst.c	368	
363		jidctint.c	389	
364		jidctred.c	398	
365				jinclude.h
366		jmemansi.c	168	
367		jmemdoc.c	635	
368		jmemmac.c	200	
369		jmemmgr.c	1116	
370		jmemname.c	273	
371		jmemnobs.c	110	
372				jmemsys.h
373				jpegint.h
374		jpegtran.c	373	
375		jquant1.c	857	
376		jquant2.c	1311	
377		jutils.c	180	
378				jversion.h
379		lssif.c	353	

	A	B	C	D
1		ソースコード名(c,cpp)	行数	.h(同一名略)
380		rdbmp.c	440	
381		rdcolmap.c	254	
382		rdgif.c	684	
383		rdjpgcom.c	477	
384		rdppm.c	451	
385		rdrle.c	388	
386		rdswitch.c	333	
387		rdtarga.c	501	
388		wrbmp.c	443	
389		wrgif.c	506	
390		wrjpgcom.c	576	
391		wrprm.c	269	
392		wrrle.c	306	
393		wrtarga.c	254	
394		(total)	31018	
395				
396	sml	s3delete.c	244	
397		s3get.c	138	
398		s3set.c	172	
399		s3sml.c	704	
400		(total)	1258	
401				
402	tex	t3dde.c	63	
403		t3ddesub.c	259	
404		t3define.h	0	t3define.h
405		t3env.c	40	
406		t3ptype.h	0	t3ptype.h
407		t3table.c	92	
408		t3tex.c	549	
409		t3texma.c	48	
410		t3totex.c	361	
411		(total)	1412	
412				
413	tiff	tifwin.c	397	
414		(total)	397	
415				
416	u3	u2bentuk.c	2256	u3f2_defs.h
417		u3cross.c	220	u3f3_divtri.h
418		u3cut.c	650	u3f4_defs.h
419		u3dml.c	60	u3fcom.h
420		u3fcom.c	150	u3.h
421		u3grid.c	524	
422		u3hsvrgb.c	115	
423		u3ofs.c	77	
424		u3rand.c	737	
425		u3same~1.c	187	
426		u3slant.c	226	
427		u3spline.c	235	
428		u3trans.c	72	
429		(total)	5509	
430				
431	z3err	kmsg2.c	153	
432		test.c	38	
433		z3err.c	302	

	A	B	C	D
1		ソースコード名(c,cpp)	行数	.h(同一名略)
434		z3nt.c	83	
435		z3unix.c	209	
436		(total)	785	
437				
438	common	childfrm.cpp	380	
439	(srcnt)	childnam.cpp	72	
440		database.cpp	778	
441		scrollli.cpp	303	
442		(total)	1533	
443				
444	common	cmd2sndf.c	79	
445		cnvcol.c	215	
446		common.c	2135	
447		dataope.c	3487	
448		dmode.c	1	
449		dxfout.c	209	
450		his.c	1165	
451		hst_read.c	249	
452		hst_write.c	246	
453		i3ip_rdh.c	545	
454		kdbms.c	8	
455		rdh_com.c	222	
456		sunpos.c	102	
457		utility.c	369	
458		vrmlout.c	629	
459		(total)	9661	
460				
461	ntcom	b3bitmap.c	600	
462		copy.c	596	
463		dibutil.c	2742	
464		eye.c	51	
465		imaif.c	208	
466		kamera.c	402	
467		kamera2.c	579	
468		kamera2.cpp	389	
469		ksimwin.c	646	
470		light.c	405	
471		opglif.c	166	
472		pembantu.c	75	
473		pixel.c	123	
474		print.c	777	
475		texture.c	101	
476		tempfile.c	120	
477		w3argv.c	115	
478		w3dbsub.c	223	
479		w3dummy.c	45	
480		w3init.c	124	
481		w3mtfif.c	31	
482		w3ntsub.c	624	
483		w3only.c	33	
484		w3wnt.c	201	
485		w3zairyo.c	73	
486		xdraw.c	172	
487		xpixel.c	60	

	A	B	C	D
1		ソースコード名(c,cpp)	行数	.h(同一名略)
488		(total)	9681	
489				
490	ext			
491	bs2lss	bs2lss.c	999	
492		bahasa_D.cpp	134	
493		(total)	1133	
494				
495	cone	cone.c	454	
496		cone_d.cpp	139	
497		cone_dDlg.cpp	462	
498		(total)	1055	
499				
500	cube	cube.c	311	
501		cube_d.cpp	161	
502		cube_ddlg.cpp	382	
503		(total)	854	
504				
505	cylinder	cylinder.c	370	
506		cylind_d.cpp	138	
507		cylind_ddlg.cpp	430	
508		(total)	938	
509				
510	steel	csteel.c	81	
511		hsteel.c	94	
512		lsteel.c	77	
513		tsteel.c	78	
514		elemste2.cpp	200	
515		steel_dcpp	116	
516		steel_ddlg.cpp	482	
517		(total)	1128	
518				
519	flatcone	flatcone.c	316	
520		flatcone_d.cpp	141	
521		flatcone_ddlg.cpp	545	
522		(total)	1002	
523				
524	flatcyli	flatcyli.c	299	
525		flatcyli_d.cpp	118	
526		flatcyli_ddlg.cpp	393	
527		(total)	810	
528				
529	geoload	geoload.cpp	155	
530		StdAfx.cpp	6	
531		(total)	161	
532				
533	period	period.c	59	
534		period_d.cpp	153	
535		period_ddlg.cpp	314	
536		(total)	526	
537				
538	sample	sample.c	47	
539		sample_d.cpp	157	
540		sample_ddlg.cpp	236	
541		(total)	440	

	A	B	C	D
1		ソースコード名(c,cpp)	行数	.h(同一名略)
542				
543	sphere	sphere.c	137	
544		sphere_d.cpp	114	
545		sphere_ddlg.cpp	301	
546		(total)	552	
547				
548	stair	stair.c	58	
549		stair_d.cpp	141	
550		stair_ddlg.cpp	298	
551		(total)	497	
552				
553	string	string_d.cpp	157	
554		string_ddlg.cpp	291	
555		(total)	448	
556				
557	sweep1	sweep1.c	239	
558		sweep1_d.cpp	116	
559		sweep1_ddlg.cpp	295	
560		(total)	650	
561				
562	sweep2	sweep2.c	1169	
563		sweep2_d.cpp	115	
564		sweep2_ddlg.cpp	259	
565		(total)	1543	
566				
567	tamentai	tamentai.c	416	
568		tamentai_d.cpp	164	
569		tamentai_ddlg.cpp	232	
570		(total)	812	
571				
572	vrml2lss	vrml2lss.c	3272	
573		vrml2lss_d.cpp	134	
574		vrml2lss_ddlg.cpp	0	
575		(total)	3406	
576				
577	sxf	scadec21.c	1571	
578		scadec_D.cpp	172	
579		scadec_DDlg.cpp	553	
580		(total)	2296	
581				
582	貿易	貿易.c	1424	
583		FILER.c	78	
584		Showhelp.c	74	
585		貿易.rc	317	
586		dtm2lss.c	689	
587		dxf2lss.c	55	
588		dxf2lss2.c	1109	
589		dxf2lss4.c	1055	
590		error.c	43	
591		mc2lss.c	1855	
592		conv.cpp	329	
593		rgb2sgi.c	78	
594		(total)	7106	
595				

	A	B	C	D
1		ソースコード名(c,cpp)	行数	.h(同一名略)
596	都市開発	dialog.c	290	
597		filer.c	65	
598		flscr.c	92	
599		hutan.c	420	
600		matahari.c	86	
601		motomati.c	1328	
602		output.c	1292	
603		perihara.c	28	
604		things.c	367	
605		tifwin.c	396	
606		yosemune.c	154	
607		(total)	4518	
608				
609	maju	filer.c	291	
610		motomati.c	1350	
611		AddNewProject.cpp	43	
612		areasel.cpp	418	
613		cond.cpp	560	
614		conddet.cpp	92	
615		CondDsb.cpp	56	
616		Cross.cpp	600	
617		Crossing.cpp	227	
618		DddDet.cpp	101	
619		Denken.cpp	43	
620		DisasterForecasting.cpp	179	
621		EconomicalForecasting.cpp	79	
622		Eval.cpp	176	
623		Eval2.cpp	71	
624		FILER.cpp	2720	
625		FireProtection.cpp	42	
626		Gempa.cpp	90	
627		Help.cpp	59	
628		Jiban.c	90	
629		Kenpei.cpp	49	
630		Maju.cpp	72	
631		Majudlg.cpp	470	
632		ProjArea.cpp	70	
633		Show_3d.cpp	644	
634		SHOWHELP.cpp	72	
635		Simulate.cpp	171	
636		STDAFX.cpp	5	
637		Surface.cpp	75	
638		Tipologia.cpp	231	
639		UrbanPlan.cpp	43	
640		UseControl.cpp	43	
641		Util.cpp	575	
642				majucev.h
643				majufun.h
644		maju.rc	1008	resource.h
645		(total)	10815	

付録B. ライブラリ関数

第一版　　平成１０年３月
第二版　　平成２２年３月

はじめに

第４章で解説したように、景観シミュレーションのシステムは、大きく見て、８のライブラリと、ダイアログ毎の８０程度のハンドラ・ルーチンがその大半を占めている。前者は、開発初期の頃から一貫したデータ構造を処理するために維持・改良されてきたものである。可搬性が高いＡＮＳＩ－Ｃ言語をベースとして記述されている。初期の頃と比較すると、データ規模も大きくなってきており、またマシンのメモリ容量、ハードディスク容量、処理速度などは比較にならない程進歩したが、基本的な構造は何も変わっていない。

一方、後者のダイアログ・ハンドラは、操作性等のインターフェースに係る現場のニーズや、新たなＯＳや開発環境に適応しつつ、柔軟に機能追加や改良を行った結果、現状に至っている。その多くは、C++で書かれ、開発環境(Wizard 等)が自動的に生成するソースコードや、開発者向けのサンプルから引用した部分も含んでいる。

長期的に見ると、知的財産としてのソフトウェアの価値の大きな部分は、いわばインフラストラクチャとしてのライブラリ関数に存在する。一方、ユーザーから見える部分に相当するダイアログ・ハンドラは、将来の OS やニーズの変化に伴って、今後も更に柔軟に修正を求められる可能性があり、より鮮度が求められる消耗品的な性格が強い。

この付録Ｂでは、個々のライブラリ関数に関して解説する。

ライブラリ関数はそれぞれの目的別に分かれていて、ＳＭＬ（シーン管理ライブラリ)、ＤＭＬ（データ管理ライブラリ)、ＤＢＩＬ（景観データベース・インターフェース・ライブラリ)、Ｇ３ＤＲＬ（レンダリングライブラリ)、ＩＰ（インタプリタライブラリ)、Ｕ３（ユーティリティライブラリ)、ＥＮＶ（環境設定ライブラリ)、Ｚ３ＥＲＲ（メッセージライブラリ）がある。

ＳＭＬ　：　シーンデータを管理するライブラリ
ＤＭＬ　：　ジオメトリデータを管理するライブラリ
ＤＢＩＬ　：　景観データベースを管理するライブラリ
Ｇ３ＤＲＬ　：　データの描画に関するライブラリ
ＩＰ　：　インタプリタのライブラリ（ファイルの入出力）
Ｕ３　：　ユーティリティ関連で、計算等のライブラリ
ＥＮＶ　：　システムの環境を管理するライブラリ
Ｚ３ＥＲＲ　：　メッセージウインドウを表示させるライブラリ

最も素朴なシステム構成であるビューワにおいては、以下のような処理の流れとなる。

①最初に以下の関数を始めに１度だけ呼び出す。

```
e3Initialize();
s3Initialize();
d3Initialize();
g3Initialize();
dbInitialize(NULL);
z3LoadMessage();
dbSetDB(0);
dbOpen(0, NULL);
dbSetDB(1);
dbOpen(1, NULL);
dbSetDB(2);
dbOpen(2, NULL);
dbSetDB(3);
dbOpen(3, NULL);
```

②プログラム終了時には以下の関数を呼び出す。

```
dbSetDB(0);
dbClose();
dbSetDB(1);
dbClose();
dbSetDB(2);
dbClose();
dbSetDB(3);
dbClose();
dbExit();
z3CloseMessage();
```

③あるシーンを表示したい場合は、先ずシーンファイルをロードし、シーン配列のアドレスを取得する。

```
s3Scene **scn_array;
int scn_num = dbLoadScene("sample", &scn_array);
```

④経年変化情報をセットするには以下のようにする。

```
dbChangeInfo info;
s3Time *t = s3GetSceneTime(scn_array[0]);
s3GetTimeParam(t, &info.date);
```

```
    dbSetChangeInfo(&info);
```
⑤次にマテリアル及びテクスチャをロードする。
```
    g3LoadMaterialAll();
    g3LoadTextureAll();
```
⑥最後にシーンを描画する。
```
    g3AssignDrawarea((void*)&m_hdc);
    g3SetScene(scn_array[0]);
    wg3Redraw((void*)&m_hdc);
```

初版では、関数の仕様に、構想段階のものがあり、実際の関数の動作とは異なっているものがあり、第二版では訂正した。また、初版では、メモリ管理の一貫性がまだ不十分であり、終了時のメモリ解放処理が不足している部分があったが、これを補った。

アプリケーションライブラリに関しては、本文で解説した通り、ライブラリ関数とダイアログ・ハンドラの中間を媒介する機能として、その後多くの関数が作成されたため、これらを追記すると共に、編成を改めた。

目次（付録B）

Ｂ－１．シーン管理ライブラリ（ＳＭＬ）

シーン管理ライブラリ（ＳＭＬ）とは、ＬＳＳ-Ｓファイルの内容をメモリ上に管理するものである。

（１）システム関数

```
char *s3Version();
```

ＳＭＬのバージョンを取得する。
返り値 バージョンを示す文字列

```
int s3Initialize();
```

ＳＭＬを初期化する。
返り値 成功時１　エラー時０

（２）データ構築関数

```
s3Scene *s3CreateScene(char *name, int type);
```

シーンを生成する
name シーン名称
type シーンタイプ
返り値 メモリ・ブロックのアドレス　エラー時 NULL

```
s3Image *s3CreateImage(char *name, char *filename);
```

イメージを読み込む
name イメージ名称
filename イメージファイル名称
返り値 メモリ・ブロックのアドレス　エラー時 NULL
s3Image には既にデータが格納される。以下の通り。

```
typedef struct _s3Image {
 char *name;
 char *filename;
 int width;
 int height;
 int format;
 int type;
 void *pixels;
} s3Image;
```

width～pixels はＯpenＧＬの glDrawPixels のパラメータである。

イメージファイルを読み込む為にＤＢＩＬの dbLoadImage を用いる

s3Model *s3CreateModel(char *name, char *filename);
LSS-G ファイルをロードし、モデルを作成する。
name モデル名称
filename 形状ファイル名称
返り値 メモリ・ブロックのアドレス　エラー時 NULL
s3Model には既にデータが格納されている。以下の通り。

```
typedef struct _s3Model {
  char *name;
  char *filename;
  d3Group **root; /* array of root groups */
  int num;/* numbers of root groups */
} s3Model;
```

num は通常 1 である。
形状ファイルを読み込む為にＤＢＩＬの dbLoadGeometry を用いる

s3Camera *s3CreateCamera(char *name);
視点データを生成する。
name 視点名称
filename 形状ファイル名称
返り値 メモリ・ブロックのアドレス　エラー時 NULL

s3Light *s3CreateLight(char *name, int type);
光源ユニットを生成する。
name 光源名称
type 光源タイプ
返り値 メモリ・ブロックのアドレス　エラー時 NULL

s3LightGroup *s3CreateLightGroup(char *name);
光源グループを生成する。
name 光源グループ名称
返り値 メモリ・ブロックのアドレス　エラー時 NULL

s3Effect *s3CreateEffect(char *name, int type);
効果ユニットを生成する。

name 効果ユニット名称
type 効果タイプ
返り値 メモリ・ブロックのアドレス　エラー時 NULL

s3EffectGroup *s3CreateEffectGroup(char *name);
効果グループを生成する。
name 効果グループ名称
返り値 メモリ・ブロックのアドレス　エラー時 NULL

s3Time *s3CreateTime(char *name);
時間を生成する。
name 時間名称
返り値 メモリ・ブロックのアドレス　エラー時 NULL

void s3InitSceneName();
ユニークなシーン名称作成機能を初期化する。

const char *s3NewSceneName();
ユニークなシーン名称を作成する。
返り値 メモリ・ブロックのアドレス
新たに割り当てたメモリ・ブロック上に「SCNnnn」(nnn は整数値) という形式の文字列を生成し、これを全てのシーンの名称と比較する。もし一致するシーンが無ければ、このメモリ・ブロックを返す。一致するものがあれば、整数値を一つ増加させ同じ比較を行う。整数値はスタティック変数に保持され、次に関数が呼び出された時は前回よりも一つ大きな値から出発する。s3InitSceneName 関数により、スタティック変数を 1 にリセットする。

void s3InitImageName();
ユニークなイメージ名称作成機能を初期化する。

const char *s3NewImageName();
ユニークなイメージ名称を作成する。
返り値 メモリ・ブロックのアドレス

void s3InitModelName();
ユニークなモデル名称作成機能を初期化する。

const char *s3NewModelName();
ユニークなモデル名称を作成する。
返り値 メモリ・ブロックのアドレス

void s3InitCameraName();
ユニークなカメラ名称作成機能を初期化する。

const char* s3NewCameraName();
ユニークなカメラ名称を作成する。
返り値 カメラ名称のアドレス

void s3InitLightGroupName();
光源グループ名称の初期化

const char *s3NewLightGroupName();
ユニークな光源グループ名称を作成する。
返り値 メモリ・ブロックのアドレス。

void s3InitLightName();
ユニークな光源ユニット名称作成機能を初期化する。

const char *s3NewLightName();
ユニークな光源ユニット名称を作成する。
返り値　メモリ・ブロックのアドレス　エラー時 NULL

void s3InitEffectGroupName();
ユニークな効果グループ名称作成機能を初期化する。

const char *s3NewEffectGroupName();
ユニークな効果グループ名称の名前を作成する。
返り値 メモリ・ブロックのアドレス

void s3InitEffectName();
ユニークな効果ユニット名称作成機能を初期化する。

const char *s3NewEffectName();

ユニークな効果ユニット名称を作成する。

void s3InitTimeName();
ユニークな時間名称の作成機能を初期化する

const char *s3NewTimeName();
ユニークな時間名称を作成する。
返り値 メモリ・ブロックのアドレス

（３）データ定義/更新関数

void s3SetSceneImageBack(s3Scene *scn, s3Image *img);
シーンに背景イメージをセットする。
scn 対象となるシーンのアドレス
img セットするイメージのアドレス
イメージは１つだけセットできる

void s3SetSceneImageFront(s3Scene *scn, s3Image *img);
シーンに前景イメージをセットする。
scn 対象となるシーンのアドレス
img イメージのアドレス
イメージは１つだけセットできる

void s3SetSceneModel(s3Scene *scn, s3Model *mdl);
シーンにモデルをセットする。
scn シーンのアドレス
mdl モデルのアドレス
モデルは１つだけセットできる

void s3SetSceneCamera(s3Scene *scn, s3Camera *cam);
シーンに視点をセットする。
scn シーンのアドレス
cam 視点のアドレス
視点は１つだけをセットできる

void s3SetSceneLightGroup(s3Scene *scn, s3LightGroup *lg);
シーンに光源グループをセットする。

scn シーンのアドレス
lg 光源グループのアドレス
光源グループは１つだけセットできる

void s3SetSceneEffectGroup(s3Scene *scn, s3EffectGroup *eg);
シーンに効果グループをセットする。
scn シーンのアドレス
eg 効果グループのアドレス
効果グループは１つだけセットできる

void s3SetSceneTime(s3Scene *scn, s3Time *t);
シーンに時間をセットする。
scn シーンのアドレス
t 時間のアドレス

void s3SetCameraParam(s3Camera *cam,double eyex, double eyey, double eyez,
  double centerx, double centery, double centerz,double upx, double upy, double upz,
  double fovy, double aspect, double zNear, double zFar);

s3Camera 構造体にパラメータをセットする。
cam カメラ情報を格納するアドレス
eyex,eyey,eyez　視点座標
centerx,centery,centerz　注視点座標
upx,upy,upz アップベクトル（カメラのレンズ軸回りの傾きを示す）
（以上９変数は OpenGL の gluLookAt 関数の引数に対応する）
fovy　垂直方向の視野角（度）
aspect　画面の横サイズ／縦サイズ
zNear　表示前後範囲の下限距離
zFar　表示前後範囲の上限距離
（以上４変数は OpenGL の gluPerspective 関数の引数に対応する）

void s3SetLightGroupLight(s3LightGroup *lg, s3Light *lit);
光源グループに光源ユニットをセットする。
lg 光源グループのアドレス
lit　光源ユニットのアドレス
光源ユニットは複数個セットできる

void s3SetLightColor(s3Light *lit, float r, float g, float b);
光源ユニットに色をセットする。
lit 対象となる光源ユニットのアドレス
r,g,b 色 (0.0≦≦1.0)

void s3SetLightPosition(s3Light *lit,float x, float y, float z, float w);
光源に位置をセットする
lit 光源ユニットのアドレス
x,y,z 座標値
w 光源の種類 (0.0 平行光源　1.0 点光源)

void s3SetEffectGroupEffect(s3EffectGroup *eg, s3Effect *ef);
効果グループに効果ユニットをセットする
eg 効果グループのアドレス
ef 効果ユニットのアドレス
効果ユニットは複数個セットできる

void s3SetEffectParam(e3Effect *e, int i, char *param);
効果ユニットにパラメータをセットする
e 対象とする効果ユニット
i セットするパラメータの番号
param セットするパラメータ (文字列) の内容
(すでにパラメータがセットされていた場合には、これを解放する。引数で示したパラメータのコピー(メモリ・ブロック)を作成し、セットする。)

void s3SetTimeParam(s3Time *t, float date);
時間に日数をセットする。
t 時間のアドレス
date 日数 (不動小数) で記述した時間

(4) データ取得関数

int s3GetSceneNum();
シーンの個数を得る。
返り値 シーンの総数

s3Scene *s3GetScene(int index);
一つのシーンを取得する。
index シーンの通し番号
返り値 メモリ・ブロックのアドレス　エラー時 NULL (index の値が範囲外)

```
全てのシーンを得る方法：
n = s3GetSceneNum();
for (i = 0; i < n; ++i) {
  scn = s3GetScene(i);
  ［scn に対する処理］
}
```

s3Image *s3GetSceneImageBack(s3Scene *scn);
シーンの背景イメージを取得する。
scn シーンのアドレス
返り値 メモリ・ブロックのアドレス　エラー時 NULL

s3Image *s3GetSceneImageFront(s3Scene *scn);
シーンの前景イメージを取得する。
scn シーンのアドレス
返り値 前景イメージのアドレス

s3Model *s3GetSceneModel(s3Scene *scn);
シーンのモデルを取得する。
scn シーンのアドレス
返り値 モデルのアドレス

s3Camera *s3GetSceneCamera(s3Scene *scn);
シーンのカメラ情報を取得する。
scn シーンのアドレス
返り値 カメラ情報 (構造体) のアドレス

s3LightGroup *s3GetSceneLightGroup(s3Scene *scn);
シーンの光源グループを取得する。
scn シーンのアドレス
返り値 光源グループのアドレス

s3EffectGroup *s3GetSceneEffectGroup(s3Scene *scn);
シーンの効果グループを取得する。
scn シーンのアドレス
返り値 効果グループのアドレス

s3Time *s3GetSceneTime(s3Scene *scn);
シーンの時間を取得する。
scn シーンのアドレス
返り値　時間構造体のアドレス

void s3GetCameraParam(s3Camera *cam,double *eyex, double *eyey, double *eyez, double *centerx, double *centery, double,*centerz, double *upx, double *upy, double *upz, double *fovy, double *aspect,double *zNear, double *zFar);
視点のパラメータを s3Camera 構造体から取得し、各構成要素に分解する。
cam カメラ情報のアドレス
eyex,eyey,eyez　視点座標
centerx,centery,centerz　注視点座標
upx,upy,upz アップベクトル（カメラのレンズ軸回りの傾きを示す）
（以上９変数は OpenGL の gluLookAt 関数の引数に対応する）
fovy　垂直方向の視野角（度）
aspect　画面の横サイズ／縦サイズ
zNear　表示前後範囲の下限距離
zFar　表示前後範囲の上限距離
（以上４変数は OpenGL の gluPerspective 関数の引数に対応する）

int s3GetLightGroupLightNum(s3LightGroup *lg);
光源グループに割り当てられた光源ユニットの個数を取得する。
lg 光源グループのアドレス
返り値 光源ユニット個数

s3Light *s3GetLightGroupLight(s3LightGroup *lg, int index);
光源グループの光源ユニットを取得する。
lg 光源グループのアドレス
index 光源グループにおける光源ユニットの通し番号
返り値 光源ユニットのアドレス エラー時 NULL(index が範囲外)

光源グループの全ての光源ユニットを得る方法：

```
n = s3GetLightGroupLightNum(lg);
for (i = 0; i < n; ++i) {
  lit = s3GetLightGroupLight(lg, i);
   [lit に対する処理]
}
```

void s3GetLightColor(s3Light *lit,float *r, float *g, float *b);
光源ユニットの色を取得する。
lit 光源ユニットのアドレス
r,g,b 色の格納先アドレス

void s3GetLightPosition(s3Light *lit, float *x, float *y, float *z, float *w);
光源ユニットの位置を取得する。
lit 光源ユニットのアドレス
x,y,z,w のアドレス

int s3GetEffectGroupEffectNum(s3EffectGroup *eg);
効果グループに割り当てられた効果ユニットの個数を取得する。
eg 効果グループのアドレス
返り値 効果ユニット個数

s3Effect *s3GetEffectGroupEffect(s3EffectGroup *eg, int index);
効果グループから一つの効果ユニットを取得する。
eg 効果グループのアドレス
index 効果ユニットの通し番号
返り値 効果ユニットのアドレス エラー時 NULL(index が範囲外)

効果グループの全ての効果ユニットを得る方法：

```
n = s3GetEffectGroupEffectNum(eg);
for (i = 0; i < n; ++i) {
  ef = s3GetEffectGroupEffect(eg, i);
}
```

void s3GetEffectParam(s3Effect *e, int index);
効果ユニットのパラメータを取得する
e 効果グループのアドレス
index 求めるパラメータの通し番号

void s3GetTimeParam(s3Time *t, float *date);
時間（構造体）から時間（日数）を取得する。
t 時間（構造体）のアドレス（取得先）
date 時間のアドレスの格納先

int s3SearchLightGroupLightName(s3LightGroup *lg, char *name);
光源グループ内で、指定した名称と同名の光源ユニットを検索する。
lg 光源グループのアドレス
name 光源ユニット名称
返り値 一致した光源ユニットの通し番号 ないとき－1

int s3SearchEffectGroupEffectName(s3EffectGroup *eg, char *name);
効果グループ内で、指定した名称と同名の効果ユニットを検索する。
eg 効果グループのアドレス
name 効果ユニット名称
返り値 一致した光源ユニットの通し番号 ないとき－1

int s3IsSameScene(const char *name);
同じシーン名称がすでにあるかチェックする。
name シーン名称
返り値 1:ある 0:ない

int s3IsSameImage(int scnno, const char *name);
同じイメージ名称がすでにあるかチェックする。
scnno 検査から外すシーンの番号
name イメージ名称
返り値 1:ある 0:ない

int s3IsSameModel(int scnno, const char *name);
同じ名称のモデルがすでにあるかチェックする。
scnno 検査から外すシーンの番号
name モデル名称
返り値 1:ある 0:ない

int s3IsSameCamera(int scnno, const char *name);

同じ名称のカメラ情報がすでにあるかチェックする。
scnno 検査から外すシーンの番号
name カメラ名称
返り値 1:ある 0:ない

int s3IsSameLightGroup(int scnno, const char *name);
同じ光源グループ名称がすでにあるかチェックする。
scnno 検査から外すシーンの番号
name 光源グループ名称
返り値 1:ある 0:ない

int s3IsSameEffectGroup(int scnno, const char *name);
同じ効果グループ名称がすでにあるかチェックする。
scnno 検査から外すシーンの番号
name 効果グループ名称
返り値 1:ある 0:ない

int s3IsSameLight(const char *name);
同じ光源ユニット名称がすでにあるかチェックする。
name 光源ユニット名称
返り値 1:ある 0:ない

int s3IsSameEffect(const char *name);
同じ効果ユニット名称がすでにあるかチェックする。
name 効果ユニット名称
返り値 1:ある 0:ない

int s3IsSameTime(int scnno, const char *name);
同じ時間名称がすでにあるかチェックする。
scnno 検査から外すシーンの番号
name 時間名称
返り値 1:ある 0:ない

（5）データ削除関数

void s3DeleteScene(s3Scene *scn);
シーンを削除する。

scn シーンのアドレス

void s3DeleteImage(s3Image *img);
イメージを削除する。
img イメージのアドレス

void s3DeleteModel(s3Model *mdl);
モデルを削除する。
mdl モデルのアドレス

void s3DeleteCamera(s3Camera *cam);
視点を削除する。
cam カメラ情報のアドレス

void s3DeleteLightGroup(s3LightGroup *lg);
光源グループを削除する。
他のグループに登録されていない光源ユニットも削除する
lg 光源グループのアドレス

void s3DeleteLightGroup2(s3LightGroup *lg);
光源グループをシーンから登録抹消する。ただし、光源ユニットは削除しない。
lg 光源グループのアドレス

void s3DeleteLightGroupLight(s3LightGroup *lg, int index);
光源グループの光源ユニットを削除する。
lg 光源グループのアドレス
index lg 内の光源ユニットの通し番号

void s3DeleteLightGroupLight2(s3LightGroup *lg, int index);
光源グループの光源ユニットの登録を解除する。
lg 光源グループのアドレス
index lg 内の光源ユニットの通し番号

void s3DeleteLightGroupLight2(s3LightGroup *lg, int index);
光源グループの光源ユニットを登録抹消する。
lg 光源グループのアドレス

index 光源グループ内の光源ユニットの通し番号
（光源ユニットのメモリ・ブロック自体は残す）

void s3DeleteLight(s3Light *lit);
光源ユニットを削除する。
lit 光源ユニットのアドレス

void s3DeleteEffectGroup(s3EffectGroup *eg);
効果グループを削除する。配下の効果ユニットに関して、他の効果グループからの重複利用状況を検査し、兼業していない効果ユニットだけ削除する。
eg 効果グループのアドレス

void s3DeleteEffectGroup2(s3EffectGroup *eg);
効果グループを削除する。ただし、配下の効果ユニットは削除しない。
eg 効果グループのアドレス

void s3DeleteEffectGroupEffect(s3EffectGroup *eg, int index);
効果グループの通し番号で指定した効果ユニットを削除する。
eg 効果グループのアドレス
index 処理対象とする eg 配列内の効果ユニットの番号

void s3DeleteEffectGroupEffect2(s3EffectGroup *eg, int index);
効果グループの効果ユニットを登録解除する。効果ユニット自体は存置する。
eg 効果グループのアドレス
index 処理対象とする eg 配列内の効果ユニットの番号

void s3DeleteEffect(s3Effect *ef);
効果ユニット(メモリ・ブロック)を解放する。
ef 効果ユニットのアドレス

void s3DeleteTime(s3Time *t);
時間を削除する。
t 時間のアドレス

void s3AllClear();
全てのシーンを削除する。

Ｂ－２．データ管理ライブラリ（ＤＭＬ）

データ管理ライブラリ（ＤＭＬ）は、地物をモデリングするためのデータ構造（ＬＳＳ-Ｇファイルとして保存される）をメモリ上で管理する。基本的な単位は、グループであり、その配下に面、線、頂点から構成される幾何学的要素、カラー、マテリアル、テクスチャから成る表面仕上げを扱う。更に、外部ファイル参照や、地面、住宅物件、物理・化学的属性等、任意に追加可能な属性データを処理するための基本的機能を提供している。

ＤＭＬライブラリにおいてじゃ、d3db[]というスタティック変数(配列)を起点としたグループのテーブルを管理しており、ここに全てのグループを登録する。全てのグループを対象とした検索等の処理は、このテーブルを用いるのが速い。G3DRL ライブラリが管理する表示処理においても、グループ間のリンク構造を辿って全てのグループにアクセスする処理を行っているが、表示のためのリンクでは、一つの子グループが複数の親を有することができる構造となっているため、同じグループが異なる文脈で複数回アクセスされる。

グループのテーブル d3db の実体は、DML ライブラリの中にスタティック変数として定義された d3DB 構造体の配列である。一つの構造体は、グループのリストと、グループ総数を管理している。グループのリストには、グループへのポインタと削除フラグがあり、処理速度が求められる削除処理に当たって、このフラグを用いて実際のグループに関連したメモリ・ブロックの解放処理（リンク構造の確認を含むやや複雑な処理）を先送りすることができる。

（１）システム関数

char *d3Version();
ＤＭＬのバージョンを取得する
返り値 バージョンを示す文字列

int d3SetDB(int no);

グループのテーブル番号をセット（選択）する。デフォルトは０である。このコマンド以降に実行される関数は、選択されたテーブル、即ち一つの d3DB 構造体に対して適用される。Ver.2.09 の基幹部分(sim.exe)においては、一つのテーブルしか使用していない。
no テーブル番号 (0≦≦9)
返り値 成功値 1 エラー時 0

int d3GetDB();
グループのテーブル番号を取得する。
no テーブル番号 (0≦≦9)
返り値 データベース番号

int d3Initialize();

現在選択されているグループのテーブルに登録されている全てのグループとこれらに関連したメモリ・ブロックを解放し、テーブルを初期化する。
返り値 成功値1 エラー時0

void d3Bebas();
現在選択されているグループのテーブルに登録されている全てのグループと、これらに関連する各種メモリ・ブロックを解放する。終了時に呼び出す。

（2）データ構築関数

d3Group *d3CreateGroup(char *name, int type, char *kind);
グループを生成する。
name グループ名称 （ユニークな名前）
type グループタイプ
kind グループ種別
返り値　グループ(メモリ・ブロック)のアドレス, エラー時NULL

d3Link *d3CreateLink(d3Group *parent, d3Group *child);
リンクを定義する
変換マトリクスは単位行列となる
parent 親グループ
child 子グループ
返り値　リンクのアドレス, エラー時NULL

int d3RegisterMaterial(char *name);
マテリアルを登録する
name マテリアル名称
返り値　マテリアルID番号(＞0)
エラー時0
既に同じ名称が存在する場合には、同じID番号が返る

本関数は名称とIDを対応付けするのみであり、他には何も行わない
実際のデータを使用したい場合はMILを用いる

int d3RegisterTexture(char *name);
テクスチャを登録する
name テクスチャ名称

返り値　テクスチャ ID 番号(＞0), エラー時 0
既に同じ名称が存在する場合には、同じ ID 番号が返る

本関数は名称と ID を対応付けするのみであり、他には何も行わない
実際のデータを使用したい場合は TIL を用いる

int d3CalcBoundingBox(d3Group *g);
グループのバウンディングボックスを計算し、セットする
g グループのアドレス
返り値 成功時 1
エラー時 0

int d3CalcBoundingBoxInc(d3Group *g, d3face *f);
面のバウンディングボックスをグループにセットする
g グループのアドレス
f 面のアドレス
返り値 成功時 1
エラー時 0

const char *d3NewGroupName();
ユニークなグループ名称を取得する
返り値 ユニークなグループ名称のアドレス

d3Group *d3GetNewGroupFromPoints(int count, double *points);
頂点配列からの新しいグループ（面を一つもつ）を作成する
count 頂点の数
points 頂点配列（頂点数分）
返り値 新しいグループのアドレス

d3Group *d3BangunFromPoints(int count, double *points, double t);
頂点配列からの新しいグループ（頂点列を下底とし、高さを t とする立体）を作成する。
count 頂点の数
points 頂点配列（頂点数分）
t 高さ
返り値 新しいグループのアドレス

int isconcave(d3Face *f);
面が凹ポリゴンかどうかを検査する。
f 検査対象とする面
返り値　－１（凹）　０（凸）

ketemu(double A0[3], double A1[3], double B0[3], double B1[3]);
線分 A0-A1 と線分 B0-B1 の交差を検査する。
戻り値
　３：通常交差　（通常、０＜戻り値を以て交差と見なす）
　０：同軸上で重複する場合または接続する場合
－１：交差なし（A0 と A1 が B0B1 に関して同じ側）
－２：交差なし（B0 と B1 が A0A1 に関して同じ側）
－１０：同軸上で離れている場合

（３）初期化関数

int d3Initface(d3face *f);
面データを初期化する
f グループ名称　（ユニークな名前）
返り値　成功時 1 エラー時 0
d3SetGroupface をコールする前に使用する

int d3InitVertex(d3Vertex *v, int size);
頂点データを初期化する
v 頂点配列
size 配列のサイズ
返り値　成功時 1 エラー時 0
d3SetGroupface をコールする前に使用する

void d3InitGroupName();
グループ名称作成の初期化

（４）データ定義/更新関数

int d3SetGroupName(d3Group *g, char *name);
グループに名称を付加する
g グループのアドレス
name グループ名称

返り値 成功時1 エラー時0

int d3SetGroupMaterial(d3Group *g, int id);
グループにマテリアル属性を付加する
g グループのアドレス
id マテリアルID番号
返り値 成功時1 エラー時0

int d3SetGroupTexture(d3Group *g, int id);
グループにテクスチャ属性を付加する
g グループのアドレス
id テクスチャID番号
返り値 成功時1 エラー時0

int d3SetLinkMatrix(d3Link *l, d3Matrix mat, int mode);
リンクに変換マトリクスを定義する
l リンクのアドレス
mat 変換マトリクス
mode 変換モード
D3_MM_LOAD　－　初期化する
D3_MM_PRE　－　前から乗ずる
D3_MM_POST　－　後から乗ずる
返り値 成功時1 エラー時0
変換マトリクスの要素は以下の順序で格納されるものとする
0　4　8　12
1　5　9　13
2　6　10　14
3　7　11　15
変換されるベクトルは列（column）ベクトルとする

d3face *d3SetGroupface(d3Group *g, d3face *f, d3Vertex *v);
グループに面を割り当てる
g グループのアドレス
f 面のアドレス
ｖ 頂点配列（頂点数分がある）
返り値 割り当てられた面のアドレス

f、v は内容が関数内バッファへコピーされる
f は d3Initface にて初期化しておくこと
v は d3InitVertex にて初期化しておくこと
f で指定すべきメンバは以下の通り
 unsigned long va_f (面の属性フラグ：D3_VA_***のビット OR で指定)
 unsigned long va_v (頂点の属性フラグ：D3_VA_***のビット OR で指定)
 int vnum(頂点数)
 int material(マテリアル ID)
 int texture(テクスチャ ID)
 float n[3](法線ベクトル、正規化必要)
 float c[4](色、0.0≦ ≦1.0)
v で指定すべきメンバは以下の通り
 unsigned long va(各頂点の属性フラグ：D3_VA_***のビット OR で指定)
 double v[3] (座標)
 float n[3] (法線ベクトル)
 float t[2] (テクスチャ座標)
 flaot c[4] (色)

d3Face *d3SetGroupFaceConcave(d3Group *g, d3Face *f, d3Vertex *v);
グループにコンケーブ属性を付けて面を割り当てる
g グループのアドレス
f 面のアドレス
v 頂点配列 (頂点数がある)
返り値 割り当てられた面のアドレス

int d3SetGroupUserInfo(d3Group *g, void *u, int no);
グループに任意の情報を割り当てる
g グループのアドレス
u 任意情報のアドレス
no 情報番号 (0≦ ≦9)
返り値 成功時 1 エラー時 0
u は別ライブラリにより管理される

int d3SetGroupDispInfo(d3Group *g, void *d);
グループに表示情報を割り当てる
g グループのアドレス

d 表示情報のアドレス
返り値 成功時1 エラー時0
dは別ライブラリにより管理される

int d3SetMaterialName(int id, char *name);
指定されたIDのマテリアルを変更する
id マテリアルID
name マテリアル名称
返り値 成功時1 エラー時0
指定されたIDは既にd3RegisterMaterialによって取得されていなければならない

int d3SetMaterialInfo(int id, void *info);
マテリアルに任意の情報を割り当てる
id マテリアルID
info 任意情報のアドレス
返り値 成功時1 エラー時0

int d3SetMaterialTexture(int id, int id_tex);
マテリアルにテクスチャＩＤを割り当てる
id マテリアルID
id_tex テクスチャID
返り値 成功時1 エラー時0
本関数は、マテリアルにテクスチャが付随しているような場合に利用できる

int d3SetTextureName(int id, char *name);
指定されたＩＤのテクスチャを変更する
id テクスチャID
name テクスチャ名称
返り値 成功時1 エラー時0
指定されたidは既にd3RegisterTextureによって取得されていなければならない

int d3SetTextureInfo(int id, void *info);
テクスチャに任意の情報を割り当てる
id テクスチャID
info 任意情報のアドレス
返り値 成功時1 エラー時0

int d3AddTravMatrixToTailLink(int travLevel, d3TravInfo *trav, d3Matrix m);
トラバースの最後のリンクマトリクスにマトリクスを乗じる。
tranLevel トラバースのレベル
trav トラバース情報のアドレス
m マトリクス
返り値 成功時1 エラー時0

int d3AddMatrixToTailLink(d3PickPath *path, d3Matrix m);
選択したオブジェクトのリンク連鎖の最低レベルのリンクマトリクスにマトリクスを乗じる。
path 選択
m マトリクス
返り値 成功時1 エラー時0

int d3UVMatrix(double ori[3], double u[3], double v[3], double uscale, double vscale, d3Matrix m);
テクスチャ座標を計算するためのＵＶマトリクスを、パラメータ群から生成する。
ori 原点
u U軸ベクトル
v V軸ベクトル
uscale Ｕ方向スケール
vscale Ｖ方向スケール
m 算出されたマトリックスの格納先
返り値　1

int d3GetUV1x1Frame(d3Matrix uvMat, double *vertices);
テクスチャ座標を計算するためのＵＶマトリクスに対応する、１×１の大きさの正方形の４頂点の座標を計算する。
uvMat ＵＶマトリクス
vertices 算出した４頂点の格納先
返り値　1

void d3SetFaceTextureCoord(d3Matrix uvMat, d3Matrix pathMat, d3Face *f);
面にテクスチャ座標をセットする
uvMat テクスチャのＵＶマトリクス

pathMat パスマトリクス
f 面のアドレス

void d3ClearFaceTextureCoord(d3Face *f);
面のテクスチャ座標を削除する
f 面のアドレス

void d3SetGroupTextureCoord(d3Matrix uvMat, d3Matrix pathMat, d3Group *g);
面にテクスチャ座標をセットする
uvMat テクスチャのＵＶマトリクス
pathMat パスマトリクス
g グループのアドレス

void d3ClearGroupTextureCoord(d3Group *g);
グループのテクスチャ座標を削除する
g グループのアドレス

（５）データ取得関数

int d3GetGroupNum();
グループ数を得る
返り値 グループ数

int d3GetGroup(d3GGFunc func);
グループを全て得る
func グループ毎に呼び出される関数
返り値 正常終了時１ 途中中断時０

d3GGFuncは以下のように定義される
typedef int (*d3GGFunc)(d3Group *g);
g 現在のグループのアドレス
返り値　正常終了時１ 途中中断要求時０

int d3GetRootGroup(d3GGFunc func);
ルートグループを全て得る
func ルートグループ遭遇毎に呼び出される関数
返り値 正常終了時１ 途中中断時０

int d3TraverseGroup(d3Group *g, d3TGFunc func);
グループの木構造をトラバースする
深さ優先探索で行う
g トラバースルートグループのアドレス
func トラバース毎に呼び出される関数
返り値 正常終了時1 途中中断時0
d3TGFuncは以下のように定義される

typedef int (*d3TGFunc)(d3Group *g, d3Matrix mat);
g　現在のグループのアドレス
mat　現在の変換マトリクス
返り値 正常終了時1 途中中断要求時0

int d3TraverseGroupRec(d3Group *g, d3TGFunc func, d3Matrix mp, d3Matrix ml);
グループの木構造をトラバースする
g グループのアドレス
func トラバース毎に呼び出される関数
mp マトリクス
ml マトリクス
返り値 端終了時1
途中中断時0

int d3TraverseGroup2(d3Group *g, d3TGFunc func);
グループの木構造をトラバースする
深さ優先検索で行う
g トラバースルートグループのアドレス
func トラバース毎に呼び出される関数
返り値 正常終了時1 途中中断時0

int d3TraverseGroupRec2(d3Group *g, d3TGFunc func, d3Matrix mp, d3Matrix ml);
グループの木構造をトラバースする
g グループのアドレス
func トラバース毎に呼び出される関数
mp マトリクス
ml マトリクス

返り値 端終了時 1 途中中断時 0

void d3GetTravMatrix(int travLevel, d3TravInfo *trav, d3Matrix m);
指定したトラバースまでのマトリクスを取得する
travLevel トラバースのレベル
trav トラバース情報のアドレス
m マトリクス

void d3GetPathMatrix(d3PickPath *path, d3Matrix m);
ピッキングパスまでのマトリクスを取得する
path ピッキングのアドレス
m マトリクス

void d3GetLayoutMatrix(double trn[3], double rot[3], double scl[3], double center[3], d3Matrix layMat);
配置操作におけるスケール、回転、平行移動の操作を合成したマトリックスを計算する。
trn 平行移動
rot 回転
scl スケール
center 拡大縮小・回転の中心
layMat 結果を格納するマトリックス (配列)
void d3GetLocalLayoutMatrix(d3Matrix pathMat, d3Matrix worldLayMat, d3Matrix localMat);
ワールド座標系で表現された配置マトリックス (移動・回転・スケール) と同じ結果を、ローカル座標系において実現するために、配置オブジェクトに対して適用するマトリックスを計算する。
pathmat ローカル座標系を記述するマトリクス
worldLayMat ワールド座標系で表現した配置マトリクス
localMat 算出したローカル座標系で表現した配置マトリクスの格納先

int d3GetGroupMaterial(d3Group *g);
グループを指定してマテリアルを得る
g グループのアドレス
返り値 マテリアル ID 番号

char *d3GetGroupName(d3Group *g);

グループを指定して名称を得る
g グループのアドレス
返り値 グループ名称

int d3FilingFiles(FILE *fp);
ファイル属性を有するグループのリストを出力する。
fp 出力先のファイル
戻り値 ファイル数

int d3GetGroupTexture(d3Group *g);
グループを指定してテクスチャを得る
g グループのアドレス
返り値 テクスチャ ID 番号

d3Link *d3GetGroupULink(d3Group *g, d3Link *l_prev);
グループを指定して上位のリンクを得る
g グループのアドレス
l_prev 直前のリンクのアドレス
NULL は先頭からの意
返り値 リンクのアドレス
存在しない場合 NULL
全てのリンクを得るには以下のようにする。

```
d3Link *l;
for (l = 0; l = d3GetGroupULink(g, l); ) {
          ...
}
```

d3Link *d3GetGroupDLink(d3Group *g, d3Link *l_prev);
グループを指定して下位のリンクを得る
g グループのアドレス
l_prev 直前のリンクのアドレス
NULL は先頭からの意
返り値 リンクのアドレス 存在しない場合 NULL

全てのリンクを得るには以下のようにする

```
d3Link *l;
```

```
for (l = 0; l = d3GetGroupDLink(g, l); ) {
        ***
}
```

```
int d3GetBoundingBox(d3Group *g, double *xmin, double *xmax, double *ymin, double
*ymax,double *zmin, double *zmax);
```

グループのバウンディングボックスを得る
g グループのアドレス
xmin バウンディングボックスの最小 x 座標のアドレス
xmax バウンディングボックスの最大 x 座標のアドレス
ymin バウンディングボックスの最小 y 座標のアドレス
ymax バウンディングボックスの最大 y 座標のアドレス
zmin バウンディングボックスの最小 z 座標のアドレス
zmax バウンディングボックスの最大 z 座標のアドレス
返り値 成功時 1 エラー時 0

```
int d3GetLinkMatrix(d3Link *l, d3Matrix mat);
```

リンクを指定して変換マトリクスを得る
l リンクボックス
mat 変換マトリクス
返り値 成功時 1 エラー時 0

```
d3Group *d3GetLinkUGroup(d3Link *l);
```

リンクを指定して上位のグループを得る
l リンクのアドレス
返り値 グループのアドレス エラー時 NULL

```
d3Group *d3GetLinkDGroup(d3Link *l);
```

リンクを指定して下位のグループを得る
l リンクのアドレス
返り値 グループのアドレス エラー時 NULL

```
d3face *d3GetGroupface(d3Group *g, d3face *f_prev);
```

グループを指定して面を得る
g グループのアドレス
f_prev 直前の面のアドレス

NULL は先頭からの意
返り値 面のアドレス 存在しない時 NULL
全ての面を得るには以下のようにする

```
d3face *f;
for (f = 0; f = d3GetGroupface(g, f); ) {
    ・・・・・・・・・処理・・・・・・・・・・
}
```

d3Vertex *d3GetfaceVertex(d3face *f);
面データから頂点データを得る
f 面のアドレス
返り値 頂点配列（頂点数分がある） エラー時 NULL

void *d3GetGroupDispInfo(d3Group *g);
グループの表示情報を得る
g グループのアドレス
返り値 表示情報のアドレス

int d3Gwaktu(float tim);
時間をパラメータとする外部関数の形状を再生成する。
tim 新たな時間
返り値　再生成の対象となったグループの数

void *d3GetGroupUserInfo(d3Group *g, int no);
グループの属性情報を得る。
g グループのアドレス
no 属性番号（0:ファイル属性 1:その他の任意属性）
返り値 属性情報のアドレス

int d3GetMaterialNum();
登録されたマテリアル数を取得する
#0 も登録と見なす
返り値 マテリアル数

char *d3GetMaterialName(int id);

マテリアル名称を取得する
id マテリアル ID 番号
返り値 マテリアル名称

void d3DaftarMaterialName(FILE *fp);
使用されているマテリアル名称の一覧をファイルに出力する。
fp 出力するファイル

void *d3GetMaterialInfo(int id);
マテリアルの任意情報を取得する
id マテリアル ID 番号
返り値 任意情報のアドレス

int d3GetMaterialTexture(int id);
マテリアルに割り当てられたテクスチャＩＤを取得する
id マテリアル ID 番号
返り値 テクスチャ ID　割り当てられていない場合 0 が返る
※本関数は、マテリアルにテクスチャが付随しているような場合に利用できる

int d3GetTextureNum();
登録されたテクスチャ数を取得する
#0 も登録と見なす
返り値 テクスチャ数

char *d3GetTextureName(int id);
テクスチャ名称を取得する
id テクスチャ ID 番号
返り値 テクスチャ名称

void *d3GetTextureInfo(int id);
テクスチャの任意情報を取得する
id テクスチャ ID 番号
返り値 任意情報のアドレス

d3Group *d3SearchGroup(const char *gname);
グループ名でグループを検索する

gname グループ名のアドレス
返り値 発見時グループのアドレス　未発見時 NULL

int d3GetTravInfo(d3TravInfo **trav);
トラバースの情報を取得する
trav トラバース情報のアドレスのアドレス
返り値 トラバース情報の数

void d3PrintMatrix(d3Matrix m);
マトリクスをコンソールにプリントする
m マトリクス

void d3PrintPointd(double *p);
頂点をコンソールにプリントする
p 頂点(x、y、z)

void d3PrintPointf(float *p);
頂点をコンソールにプリントする
p 頂点(x、y、z)

int d3FindNormal(d3face *f, double *n);
面の法線のベクトルを取得する
f 面のアドレス
n 法線のアドレス
返り値 成功時 1　エラー時 0

int d3FindNormalf(d3face *f, float *n);
面の法線のベクトルを取得する
f 面のアドレス
n 法線のアドレス
返り値 成功時 1　エラー時 0

int d3CalcNormal(double p[][3], int n, double *v);
頂点から法線ベクトルを求める
p 頂点配列
n 頂点数

v 法線ベクトル（単位ベクトル）
返り値 成功時１　エラー時０

int d3CalcNormalf(double p[][3], int n, float *v);
頂点から法線ベクトルを求める
p 頂点配列
n 頂点数
v 法線ベクトル（単位ベクトル）
返り値 成功時１　エラー時０

（６）データ削除関数

int d3DeleteGroup(d3Group *g);
グループを消去する
g グループのアドレス
返り値 成功時１　エラー時０
本関数は指定されたグループの上位のリンクを削除する
更に下位のリンクが無ければグループそのものも削除する

int d3DeleteGroupAll(d3Group *g);
グループを削除する
g グループのアドレス
返り値 成功時１　エラー時０
本関数は以下のものを削除する
　指定されたグループの上位リンク
　指定されたグループ
　指定されたグループの下位リンク及びグループ全て

int d3DeleteGroupCepathabis(d3group *g);
グループとその配下のグループを全て削除する。他のグループからのリンクの検査を省略し、迅速に処理する。ルートグループ以下全てのグループを省略する場合に使用する。
g 削除するリンク連鎖の頂上のグループ
戻り値　1

int d3DeleteLink(d3Link *l);
リンクを削除する
l リンクのアドレス

返り値 成功時１　エラー時０

int d3DeleteGroupBebas(d3Group *g);
グループとその配下のグループを削除する。但し、子グループが別の親グループからもリンクされている場合には、その子グループは存置する。
g 削除するグループ
返り値　１

int d3DeleteGroupFace(d3Group *g, d3Face *f);
グループの面を消去する
g グループのアドレス
f 面のアドレス
返り値 成功時１　エラー時０

int d3DeleteGroupFaceAll(d3Group *g);
グループの面を全て消去する
g グループのアドレス
返り値 成功時１　エラー時０

（７）データ複写関数

d3Group *d3CopyGroup(d3Group *g);
グループをコピーする
g グループのアドレス
返り値 グループのアドレス

（８）ピッキング関数

int d3PickGroup(float sx, float sy, d3Group *root, d3Matrix projmat, d3Matrix viewmat,
  d3Matrix r2wmat, d3PickInfo *pi, int pi_num);
スクリーン座標からグループなどを得る
sx スクリーンｘ座標（-1.0≦≦1.0）
sy スクリーンｙ座標（-1.0≦≦1.0）
root ルートグループのアドレス
projmat 投影変換マトリクス
viewmat 視野変換マトリクス
r2wmat ルートグループ座標系からワールド座標系への変換マトリクス
pi ピッキング情報が格納される配列

pi_num pi の配列数
返り値 ピッキング情報数
ピッキング情報は以下のメンバが利用できる
 d3Group *g（ピッキングされたグループのアドレス）
 d3face *f（ピッキングされた面のアドレス）
 double p[3]（ピッキングされた面上の座標値）
ピッキング情報は距離の短い順にソートされる
即ち pi[0]が最も前にある

void d3PickBebas();
ピッキング情報を解放する。
int d3PickSegment(float sx, float sy, d3Group *root, d3Matrix projmat, d3Matrix viewmat, d3Matrix r2wmat, d3PickInfo *pi, int pi_max);
スクリーン座標からグループなどを得る
sx スクリーン x 座標（-1.0≦≦1.0）
sy スクリーン y 座標（-1.0≦≦1.0）
root ルートグループのアドレス
projmat 投影変換マトリクス
viewmat 視野変換マトリクス
r2wmat ルートグループ座標系からワールド座標系への変換マトリクス
pi ピッキング情報が格納される配列
pi_num pi の配列数
返り値 ピッキング情報数

int d3GetPickPath(d3PickPath **path);
ルートグループからピッキングされたグループまでのパスを取得する
path ピッキングパスのアドレスのアドレス
返り値 ピッキング数

int d3PickLine2Cube(double *p, double *d, double xmin, double xmax, double ymin, double ymax, double zmin, double zmax, double q[][3], int *status);
直方体と直線の交点を求める
p 直線上の点
d 直線の方向ベクトル
x,y,zmax 直方体を定義する最大値
x,y,zmin 直方体を定義する最小値

q 交点座標
status 各面（６面）の交点の有無　1:交点あり　0:交点なし
返り値 ピッキング情報が格納される配列
pi_num pi の配列数
返り値　1:交点あり　0:交点なし

int d3PickLine2Patch(double *p, double *d, double *v0, double *v1, double *v2, double *n, int kind, double *q);
面と直線の交点を求める
p 直線上の点
d 直線の方向ベクトル
v0,v1v,2 面の３頂点
n 面の法線ベクトル
kind 1:平行四辺形
0:三角形
q 交点座標
返り値　1:交点あり　0:交点なし

（９）問い合わせ関数

int d3IsGroupFace(d3Group *g);
グループに面があるか調べる
g グループのアドレス
返り値　面がある場合１　面がない場合０

int d3IsSameGroup(const char *name);
同じグループ名称がすでにあるかチェックする
name グループ名称
返り値 1:ある　0:ない

const char* d3NewGroupName();
まだ使用されていないグループ名称を作成する（「GRP[番号]」の形式）
返り値：メモリ・ブロックのアドレス

（１０）マトリクス/ベクトル関数

void d3IdentMatrix(d3Matrix m);
与えられた行列を単位行列にする

m 行列

void d3CopyMatrix(d3Matrix m1, d3Matrix m2);
行列 m2 を行列 m1 にコピーする(m1=m2)
m1 行列(to)
m2 行列(flom)

void d3MultMatrix(d3Matrix m1, d3Matrix m2, d3Matrix m3);
行列 m2 と m3 を掛け合わせ、結果 m1 に返す(m1=m2*m3)
m1 行列 (結果)
m2 行列 (前から掛ける)
m3 行列 (後ろから掛ける)

void d3TransposeMatrix(d3Matrix m1, d3Matrix m2);
行列 m2 の転置行列を求め、結果 m1 に返す(m1=m2Ｔ
m1 行列 (結果)
m2 行列(from)

int d3InverseMatrix(d3Matrix m1, d3Matrix m2);
行列 m2 の逆行列を求め、結果を m1 に返す(m1=m2-1
m1 行列 (結果)
m2 行列(from)

void d3TransformPointd(double *p1, double *p2, d3Matrix m);
点を変換する
p1 変換された点 (３次元)
p2 変換される点 (３次元)
m 変換マトリクス

void d3TransformPointf(float *p1, float *p2, d3Matrix m);
点を変換する
p1 変換された点 (３次元)
p2 変換される点 (３次元)
m 変換マトリクス

void d3TransformVectord(double *v1, double *v2, d3Matrix m);

ベクトルを変換する
v1 変換されたベクトル（３次元）
v2 変換されるベクトル（３次元）
m 変換マトリクス

void d3TransformVectorf(float *v1, float *v2, d3Matrix m);
ベクトルを変換する
v1 変換されたベクトル（３次元）
v2 変換されるベクトル（３次元）
m 変換マトリクス

int d3TransformNormald(double *n1, double *n2, d3Matrix m);
法線を変換する
n1 変換された法線（３次元）
n2 変換されれる法線（３次元）
m 変換マトリクス

int d3TransformNormalf(float *n1, float *n2, d3Matrix m);
法線を変換する
n1 変換された法線（３次元）
n2 変換されれる法線（３次元）
m 変換マトリクス

double d3DotProductd(double *v1, double *v2);
ベクトルの内積を計算する
v1 ベクトル１（３次元）
v2 ベクトル２（３次元）
返り値 内積

float d3DotProductf(float *v1, float *v2);
ベクトルの内積を計算する
v1 ベクトル１
v2 ベクトル２
返り値 内積

void d3CrossProductd(double *v1, double *v2, double *v3);

ベクトルの外積を計算する (v1=v2×v3)
v1 ベクトル (結果)
v2 ベクトル (３次元)
v3 ベクトル (３次元)

void d3CrossProductf(float *v1, float *v2, float *v3);
ベクトルの外積を計算する(v1=v2×v3)
v1 ベクトル (結果)
v2 ベクトル (３次元)
v3 ベクトル (３次元)

int d3UnitVectord(double *v1, double *v2);
単位ベクトルを計算する
v1 ベクトル (結果)
v2 ベクトル (３次元)
返り値　成功時１　エラー時０

int d3UnitVectorf(float *v1, float *v2);
単位ベクトルを計算する
v1 ベクトル (結果)
v2 ベクトル (３次元)
返り値　成功時１　エラー時０

double d3VectorLengthd(double *v);
ベクトルの長さを計算する
v ベクトル (３次元)
返り値 長さ

float d3VectorLengthf(float *v);
ベクトルの長さを計算する
v ベクトル (３次元)
返り値 長さ

void d3Translated(d3Matrix m, double x, double y, double z);
平行移動のマトリクスを得る
m 変換マトリクス

x 移動ｘ成分
y 移動ｙ成分
z 移動ｚ成分

void d3Translatef(d3Matrix m, float x, float y, float z);
平行移動のマトリクスを得る
m 変換マトリクス
x 移動ｘ成分
y 移動ｙ成分
z 移動ｚ成分

void d3Rotated(d3Matrix m, double angle, double x, double y, double z);
回転のマトリクスを得る
m 変換マトリクス
angle 回転角(degree)
x 回転軸ｘ成分
y 回転軸ｙ成分
z 回転軸ｚ成分

void d3Rotatef(d3Matrix m, float angle, float x, float y, float z);
回転のマトリクスを得る
m 変換マトリクス
angle 回転角 (degree)
x 回転軸ｘ成分
y 回転軸ｙ成分
z 回転軸ｚ成分

void d3Scaled(d3Matrix m, double x, double y, double z);
スケールのマトリクスを得る
m 変換マトリクス
x スケールｘ成分
y スケールｙ成分
z スケールｚ成分

void d3Scalef(d3Matrix m, float x, float y, float z);
スケールのマトリクスを得る

m 変換マトリクス
x スケール x 成分
y スケール y 成分
z スケール z 成分

void expo(double A[3], double B[3], double H[3]);
Ｈ＝Ａ×Ｂ

void inpo(double A[3], double B[3])
返り値　Ａ・Ｂ

（１１）メモリ関数

void *d3Calloc(size_t nobj, size_t size);
malloc に準ずる
但し NULL は返らない

void *d3Malloc(size_t size);
malloc に準ずる
但し NULL は返らない

void *d3Realloc(void *p, size_t size);
realloc に準ずる
但し NULL は返らない

void d3Free(void *p);
メモリ・ブロックを解放する。free 関数を使用するが、デバッグモードにおいては、メモリ・ブロックのリストを作成し、終了時点で検出されるメモリリークの発生源を特定するための情報を提供する。
p メモリ・ブロックのアドレス

void d3Frei(void **p);
メモリ・ブロックを指すポインタが NULL でなければ、これを解放すると共に、ポインタに NULL を代入する。
p　メモリ・ブロックのアドレス。

char *d3Strdup(char *s);

文字列のコピーを作成する。元の文字列が NULL の場合には、エラー表示し、エラーを意味する文字列を生成する。
s コピー元の文字列のアドレス
返り値　成功時　コピー文字列(メモリ・ブロック)のアドレス
　　失敗時 NULL(メモリ不足)　または、エラーを明示する文字列（引数不適）

int d3GetJumlah();
使用中のメモリ・ブロックの総数を返す。
返り値　メモリ・ブロック総数

void d3ResetJumlah();
メモリ・ブロック数のカウントをゼロにリセットする。

Ｂ－３．景観データベースＩＦライブラリ（ＤＢＩＬ）

景観データベースに関する各種操作を行うために、CDataBase クラス(database.cpp)を支える機能を果たしている他、景観シミュレータで使用する外部ファイルの入出力処理機能を担っている。景観データベースにアクセスする際は、景観データベースＩＦライブラリ（ＤＢＩＬ）によりアクセスすることにより一貫性が保持される。

主に、３種類の景観データベース検索機能(yuu.exe, kou.exe 及び zai.exe)と、編集エディタ(editor.exe)で用い、データベースの登録内容は、外部ファイル com.txt の形で保存される。この外部ファイルに直接アクセスすることのない基幹部分(sim.exe)においても、モデルや画像やマテリアル情報を格納した各種の外部ファイルアクセスに際して、ＤＢＩＬライブラリの諸関数を用いている。

データベースは、ＤＢＩＬライブラリのスタティック配列 DBtab[]を基点として管理されている。DBtab は、dbDataBase 構造体の配列となっており、一つの要素が一つのデータベース（構成は異なるものであってよい）となっている。

ひとつの dbDataBase 構造体は、中心となる rechead メンバーの他に、外部ファイル名、操作履歴などに関する情報が登録されている。

rechead メンバーは、dbRecordHead 構造体へのポインタの配列となっている。要素の数は、登録物件数に対応しており、dbRecordHead 構造体(メモリ・ブロック)が、一つの建設事業や、景観構成要素などの物件の単位を記述している。

dbRecordHead 構造体には、rec メンバーがあり、これは、dbRecord 構造体の配列となっていて、その長さは登録項目数となっている。この他のメンバーとして、削除や変更を記録するフラグ等がある。

rec 配列の要素である dbRecord 構造体は、その項目に登録されたデータの配列を表す data メンバと、データの個数を表す cnt メンバーから成る。data メンバは、様々のデータ

形式を格納しうる union として定義されており、上記の rec 配列の何番目の項目であるかによって、決定される。

要約すると、DBtab から、求めるデータにアクセスする手順は次のようになる。

DBtab[データベース番号].rechead[物件番号]->rec[項目番号].data[データ番号]->型名

詳細については、dbms.h における構造体定義を参照されたい。

景観データベースの一つの特徴は、ひとつの物件のある項目に関して、複数のデータを登録できる点にある。例えば、Ａという建設事業を担当したコンサルタントが複数者であった場合、「Ａ社とＢ社」のように、ひとつの文字列にまとめることなく、「Ａ社」「Ｂ社」のように複数のデータとして、設計会社の項目の下に無制限個登録できる。

検索に際しては、各項目に関して、候補となる条件を複数設定することができる。例えば、担当者に関して「Ｉ氏」「Ｊ氏」「K氏」を検索条件として設定した上で検索操作を行うと、「G氏」「H氏」「Ｉ氏」のようにこの内の１者でも登録されている物件はヒットする。

検索条件が設定されていない項目については、常にヒットする。全ての検索項目に関してヒットした物件が最終的に検索結果となる。

ＡＮＤ検索においては、前回の検索結果を対象セットとして、その中から新たに設定された条件に合致する物件だけに絞り込みが行われる。また、ＯＲ検索においては、前回検索結果を存置して、これに新たな条件に合致する物件が追加される。

（１）システム関数

int dbInitialize(char *path);

タイムスタンプを初期化する(2.09 では意味を失っている)。

（２）データ構築関数

int dbCreateSelectionList(int itemno, int sort,int *count, int *datatype, dbData **list);

検索条件リストを作成する。

itemno 項目番号

datetype データタイプのアドレス（キーワードとそれ以外で処理が異なる）

list 検索条件に関するデータ配列のアドレス

返り値 成功時１　エラー時０

int dbReadSearchedRecord(int recno);

検索条件に合致するレコードを一つ読み込む。

recno 現在のレコード番号。これより以降を探す。

返り値 時: 合致する次のレコードの番号　ない場合:-2(レコード末尾)

void dbCutSpace(char *str, char *label);
文字列から余分なスペースを除去する（見出し用文字列作成)。
str 元の文字列
label 新しい文字列

void dbReadDeleteData();
削除したファイル(text.del)を読み込む(非使用)。

void dbReadSortType();
ソートファイル(text.srt)を読み込む(非使用)。

int dbLoadScene(char *name, s3Scene ***scn);
シーンファイル(LSS-S 形式)を読み込む。
name ファイル名
scn シーンのアドレス配列のアドレス
返り値 シーン数

dbImage *dbLoadImage(char *name);
イメージファイルを読み込む。
name 画像ファイル名
返り値 メモリ・ブロックのアドレス

int dbLoadGeometry2(char *name, d3Group ***grp);
ジオメトリファイルを読み込む
name ファイル名
grp グループのアドレス配列ののアドレス
返り値 成功時１　エラー時０

dbMaterial *dbLoadMaterial(char *name);
マテリアルを読み込む
name マテリアル名称
返り値 マテリアル情報のアドレス

dbTexture *dbLoadTexture(char *name);
テクスチャを読み込む
name ファイル名

返り値 テクスチャ情報のアドレス

int dbMakeDataBase(char *path);
データベースを作成する
path ファイルパス
返り値 成功時１　エラー時０

int dbMakeClassMenu(char *fname, int itemno);
クラスファイルを読み込み、クラステーブルを作成する
fname クラスファイル名
itemno 項目番号
返り値 成功時１　エラー時０

void dbMakeLongListTBL();
費用／価格情報テーブルを作成する。

void dbAllocItemData(dbRecord *rec);
データ格納領域を確保する
rec レコードのアドレス

unisigned long *longimagedata(char *name,long *width, long *height);
ＳＧＩ形式の画像ファイルを読み込む。
name ファイル名（拡張子.sgi）
width 幅の格納先アドレス
height 高さの格納先アドレス
返り値：画像データ（32ビット×ピクセル数）

unsigned long　*tifimagedata(char *name, long *width, long *height);
ＴＩＦＦ形式の画像ファイルを読み込む。
name ファイル名（拡張子.sgi）
width 幅の格納先アドレス
height 高さの格納先アドレス
返り値：画像データ（32ビット×ピクセル数）

unsigned long　*jpgimagedata(char *name, long *width, long *height);
ＪＰＥＧ形式の画像ファイルを読み込む。

name ファイル名（拡張子.sgi）
width 幅の格納先アドレス
height 高さの格納先アドレス
返り値：画像データ（32ビット×ピクセル数）

unsigned long　*bmpimagedata(char *name, long *width, long *height);
ＢＭＰ形式の画像ファイルを読み込む。
name ファイル名（拡張子.sgi）
width 幅の格納先アドレス
height 高さの格納先アドレス
返り値：画像データ（32ビット×ピクセル数）

（３）初期化関数

char *dbVersion();
ＤＢＩＬのバージョンを取得する。
返り値 バージョンを示す文字列

（４）データ定義/更新関数

int dbSetDB(int no);
使用するデータベースの番号をセットする。
no データベース番号（0≦≦9）
返り値 成功時１　エラー時０

int dbOpen(int type, char *name);
データベースを初期設定する。ロードは行わない。
type 対象となるデータベースの種類
　DB_SCENE 優良事例ＤＢ
　DB_GEOMETRY 景観構成要素ＤＢ
　DB_MATERIAL 景観材料ＤＢ
name データベース名称（関数内では使用していない）
返り値 成功時１　エラー時０
　不適切な type が指定された場合エラーとなる。

void dbClose();
何もしない。

void dbSetChangeInfo(dbChangeInfo *info);
時間をセットする。

dbChangeInfo *dbGetChangeInfo();
時間を取得する
返り値　構造体のアドレス

int dbExit();
何もしない。
返り値　常に１を返す

int dbSetSearchKeywd(int itemno, char *keywd);
特定の項目に検索のためのキーワードを追加する。いくつでも登録できる。
itemno 項目番号
keywd キーワードのアドレス
返り値 成功時１　エラー時０

int dbSetSearchString(int itemno, char *string)
特定の項目に検索のための文字列を追加する。いくつでも登録できる。
itemno 項目番号
string 文字列のアドレス
返り値 成功時１　エラー時０

int dbSetSearchRange(int itemno, dbData *from, dbData *to);
特定の項目に検索のための範囲をセットする。既にセットされている検索条件に上書きされる。
itemno 項目番号
from 範囲の下限
to 範囲の上限
返り値 成功時１　エラー時０

int dbSetSearchClass(int itemno, char *path);
特定の項目に検索のための分類をセットする。既にセットされている分類に上書きされる。
itemno 項目番号
path クラスパス
返り値 成功時１　エラー時０

void dbSetKeyList(char *keywd, char *menustr, int kind, int itemno);
検索のためのキーワードリストにキーワードと表示用文字列を追加する。
keywd キーワード
menustr 表示用文字列
kind 検索種別
 SELECT_KEYLIST 0 キーワード一覧からの選択
 SELECT_INPUT　1 手入力
itemno 項目番号

char *dbSetTextData(int recno, int mode);
物件の全項目を文字列として整形出力する。
recno レコード番号
mode モード
 DB_SAVE 0 ファイル保存用
 DB_DISP 1 文字情報画面出力用
返り値 整形した文字列(メモリ・ブロック)のアドレス

void dbSetHistory(char *str, int cnt, int *no);
検索履歴を追記する。DBtab[DBno].history[]の str, cnt,no[0-cnt]メンバーに書き込み、
カウンターhiscnt をインクリメントする。
str 検索条件を表現した文字列
cut 検索結果レコード数
no 検索結果レコード番号を格納した配列

void dbAllSpecifykeywd(char *keywd);
全てのレコードにキーワードを指定する
keywd キーワード

void dbSpecifykeywd(int recno, char *keywd);
一つのレコード（物件）にキーワードを設定する。
recno レコードの通し番号
keywd キーワード

void dbSetKeywd(dbStringData *str, char *keywd);
キーワードを、文字データ構造体のキーワードメンバ（配列）の末尾に追加し、カウンタ

をインクリメントする。
str 文字データ構造体のアドレス
keywd キーワード

void dbSave();
データベースをファイルにセーブする。

void dbSetModifyTime(int recno, int itemno)
変更した日時をセットする。対象とするレコードの対象項目の最初のデータに時間形式で書き込む。
recno 変更時間をセットするレコードの通し番号
itemno 変更時間をセットする項目番号

void dbWriteDeleteData();
削除データをファイル(text.del)に書き込む。

void dbWriteSortType();
ソートファイル(text.srt)に書き込む。

void dbSetDeleteData(char *data);
削除データをセットする。DBtab[DBno].deldata[]の末尾に追記し、delcnt をインクリメントする。

char *dbCutSpace2(char *data);
タブ、スペース、改行を取り除く。
data 文字列
返り値 整形後の文字列

char *dbCutSpace3(char *data);
タブ、スペースを取り除く。
data 文字列
返り値 編集後の文字列

char *dbCutBackSlush(char *data);
「¥」を取る
data 文字列

返り値 編集後の文字列

char *dbSetSpace(char *data);
文字列にスペースを入れて１行の文字数を調整する。
data 文字列
返り値 編集後の文字列

int dbSaveScene(s3Scene **scn, int num, char *name);
シーンをセーブするような関数名だが、現在何も処理しない。
scn シーンのアドレス配列のアドレス
num シーン数
name 保存に用いるファイル名
返り値 常に１

int dbSaveGeometry(d3Group **grp, int num, char *name);
ジオメトリをセーブするような関数名だが、現在何も処理しない。
grp グループのアドレス配列のアドレス
num ルートグループ数
name 保存に用いるファイル名
返り値 常に１を返す

void dbSetTopData(dbData *dbdata, char *data);
代表項目情報をセットする。
dbdata データのアドレス
data 項目タイプ配列

void dbSetColorData(dbData *dbdata, char *data);
色彩情報をセットする。
dbdata データのアドレス
data 項目タイプ配列

void dbSetLoadData(dbData *dbdata, int type, char *data);
読み込みタイプデータをセットする。
dbdata データのアドレス
type データタイプ
data 項目タイプ配列

void dbSetString(char **setbuf, char *data);
文字列をコピーしたメモリ・ブロックのアドレスをセットする。
setbuf 新しいメモリ・ブロックのアドレスの格納先
data セット元文字列のアドレス

void dbSetTitle(int itemno);
項目のタイトルをセットする。
itemno 項目番号

void dbSetRange(dbData *dbdata, int type, char *data);
範囲をセットする。範囲（数値）は、文字列から解析する。
dbdata セット先のデータのアドレス。
type データタイプ（入力値）
data 情報源文字列のアドレス

void dbSetKeywords(dbStringData *str, char *data);
キーワードをセットする。
str 設定先のアドレス
data キーワード文字列のアドレス

void dbSetClassName(dbClass *lev1, dbClass **lev2, int *c2, char *name, dbClass *cls1);
CLS ファイルから、検索のための分類項目のプルダウン・メニュー構成を構築するために、一つの分類名をセットする機能である。lev2 は lev1 のサブメニューに対応する。name 文字列を、lev2 の名称に設定すると共に、lev1 のサブメニュー・リストの末尾に、lev2 を追加する。
lev1 セットするクラスのアドレス
ilev2 セットされるクラスのアドレス
c2 階層
name クラス名
cls1 クラス情報のアドレス（単位、規格、基準に関する情報の取得先）

void SaveImageData(char *name, long xsize, long ysize, long zsize, unsigned long *pixels);
画像をイメージファイル（ＳＧＩ形式）に保存する。

name ファイル名
xsize,ysize,zsize イメージサイズ
pixels ピクセルデータのアドレス

（５）データ取得関数

dbDataBase *dbGetDataBaseAddr();
システムのデータベース配列のアドレスを得る。
返り値　スタティック配列のアドレス

int dbGetDB();
現在アクティブなデータベースの番号を得る。
返り値 データベース番号 (0≦≦9)

dbData *dbGetData(int recno, int itemno, int datano, int *datatype);
データを取得する。
recno レコード番号
itemno 項目番号
datano データ番号
datatype データタイプのアドレス
返り値 成功時:データのアドレス　エラー時:NULL

dbClass *dbGetClass(int itemno);
データベースの検索条件として、指定した項目に設定されたクラス（分類）を取得する。
itemno 項目番号
返り値 クラスのアドレス

char *dbGetModifyData(int recno, char *m_type);
修正履歴リストに表示する文字列を取得する。
recno レコード番号
m_type 修正タイプ文字列
返り値 リスト表示文字列

char *dbGetItemDataStr(dbRecordHead *rechead, int itemno, int datano, char *format);
項目データを取得する。
rechead レコードヘッドのアドレス
itemno 項目番号

datano データ番号
format フォーマット文字列（例：”%d”）
返り値 文字列

char *dbGetTitle(int itemno, int *type);
項目のタイトルを取得する(2.09 では非使用)。
itemno 項目番号
type 項目タイプ
返り値 項目タイトル文字列

int dbGetMaterialFileName(char ***name);
データベースの種類毎のディレクトリにある、マテリアルファイル名の一覧表を作成する。
name マテリアルファイル名配列のアドレス格納先
返り値 マテリアルファイル数

int dbGetMaterialName(char *name, char ***matname);
指定したマテリアルファイルの中に登録されたマテリアル名称を取得し一覧表を作る。
name マテリアルファイル名
matname マテリアル名称配列のアドレス格納先
返り値 マテリアル名称数

int dbGetItems (char *data, char ***items);
文字列をトークン（スペースまたはタブ）で区切られた項目のリストに変換する。
data 解析対象とする入力文字列
items 項目データのアドレスのアドレスのアドレス
返り値 項目データ数

int dbGetItemNo(char *data);
ラベルから、項目番号を取得する。
data 項目タイプ配列
返り値 成功時:項目番号　エラー時:-1

（６）データ削除関数

int dbClear();
DBno で選択されている一つのデータベースをクリアし、検索条件やヒストリーに至るまで、これに係る全てのメモリ･ブロックを解放する。

返り値 常に 1

void dbBebas()
変更履歴情報をクリアする。

void dbAllDeleteRecord();
DBno で選択されているデータベースの全ての登録物件を削除する。

void dbDeleteSelectionList(int count, dbData **list) ;
検索条件リストを削除する(2.09 では非使用)。
count 削除する検索条件の数、即ち配列の長さ
list 検索条件の配列のアドレス

void dbResetSearchKeywd(int itemno, char *keywd);
指定したキーワードをリスト (配列) から削除する(2.09 では非使用)。
itemno 項目番号
keywd キーワード

void dbResetSearchItem(int item);
特定の項目の検索情報を削除する。
item 削除対象とする項目番号

void dbResetSearch();
全ての項目の検索情報を削除する。

void dbResetKeyList(int index);
特定のキーワードを削除する。
index 削除するキーワードの通し番号

void dbDeleteKeyList();
全てのキーワードを削除する。

void dbResetHistory(int index);
特定の検索履歴情報を削除する。
index 削除する検索履歴情報の通し番号

void dbResetItemData(dbRecordHead *rechead, int itemno, int index);
ある登録物件の特定項目の特定登録データを削除し、登録解除する。
rechead 一つのレコード（物件）のアドレス
itemno 削除するデータ項目の番号
index 削除するデータの通し番号

void dbResetRecord(int recno);
特定のレコード（物件）を削除し、データベースから登録解除する。
recno レコードの通し番号

void dbDeleteRecord(dbRecordHead *rechead);
レコード（物件）を削除し、関連するメモリ･ブロックを解放する。
rechead レコード（物件）のアドレス

void dbDeleteItemData(dbRecord *rec, int itemno, int no);
ある登録物件の特定項目の特定登録データを削除し、関連するメモリ･ブロックを解放する。
rec レコード（物件）の全項目（配列）のアドレス
itemno 項目番号
no データの通し番号

（7）データ複写関数

void dbCopyRecord(dbRecordHead *motohead, dbRecordHead *sakihead);
レコード（物件）をコピーする。
motohead コピー元のレコード（物件）のアドレス
sakihead コピー先のレコード（物件）のアドレス

（8）比較/検索関数

int dbDataCmp(dbData *a, dbData *b);
データの文字列部分を比較する。
a データのアドレス
b データのアドレス
返り値 strcmp に準ずる

int dbSearch(int mode);
指定したモードで検索を実行する。
mode 検索モード

DB_SR_AND 絞込検索（AND 検索）
DB_SR_OR 補追検索（OR 検索）
返り値 発見数

int dbSearchRecData(dbRecord *rec);
レコード（物件）が検索条件と合致するかどうか検査する。
rec レコードの項目配列のアドレス
返り値 全ての項目が合致:1　それ以外:0
（検索条件に含まれていない項目に関しては合致として扱う）

int dbSearchString(dbRecord *rec, dbSearchData *s);
レコード（物件）の特定項目を文字列検索する。検索データは文字列の配列であり、特定項目に登録された複数データ（文字列）の一つが、検索データの文字列のいずれかと合致すれば合格とする。但し、キーワードが複数設定されている場合には、全てのキーワードがいずれかの登録データと一致するのでなければ不合格とする。
rec レコードの特定項目のアドレス
s 検索データ（文字列等の配列）のアドレス
返り値 成功時１　エラー時０

int dbSearchClass(dbRecord *rec, dbSearchData *s);
レコード（物件）の分類クラスを検索する。
rec レコードの特定項目のアドレス
s 検索データのアドレス
返り値 成功時:1　エラー時:0

int dbSearchLong(dbRecord *rec, dbSearchData *s);
レコードを long 情報検索する。検索条件は一つだけとし、条件に設定された数値範囲（不等式）が成立すれば合格とする。
rec レコードの特定項目のアドレス
s 検索条件のアドレス
返り値 成功時:1　エラー時:0

int dbSearchFloat(dbRecord *rec, dbSearchData *s);
レコードを Float 情報検索する。検索条件は一つだけとし、条件に設定された数値範囲（不等式）が成立すれば合格とする。
rec レコードの特定項目のアドレス

s 検索データのアドレス
返り値 成功時:1　エラー時:0

int dbSearchTime(dbRecord *rec, dbSearchData *s);
レコードを時間情報検索する。検索条件は一つだけとし、条件に設定された期間（不等式）にレコードの全てのデータが合致すれば合格、一つでも期間外なら不合格とする。条件として期間が設定されていなければ(その場合開始年と終了年が共にゼロ)、合格とする。
rec レコードの特定項目のアドレス
s 検索データのアドレス
返り値 成功時:1　エラー時:0

int dbMenuListCmp(dbKeywdList *a, dbKeywdList *b);
キーワードを文字列比較する。
a キーワードのアドレス
b キーワードのアドレス
返り値 strcmp に準ずる

int dbHistCmp(dbHistory *a, dbHistory *b);
検索履歴を比較する。検索履歴オルソトするための比較関数として用いる。
a 検索履歴のアドレス
b 検索履歴のアドレス
返り値 strcmp に準ずる

void dbNameSort();
データベースを「あいうえお」順にソートする。比較関数として dbNameSort 関数を指定する。

int dbNameCmp(dbRecordHead **a, dbRecordHead **b);
二つの物件の名称を比較する。
a レコードヘッド（物件の配列）のアドレス
b レコードヘッド（物件の配列）のアドレス
返り値 strcmp に準ずる

void dbTimeSort();
データベースを時間順にソートする。比較関数として dbTimeSort 関数を指定する。

int dbTimeCmp(dbRecordHead **a, dbRecordHead **b);
時間（rec[5]）を比較する
a レコードヘッド（物件の配列）のアドレス
b レコードヘッド（物件の配列）のアドレス
返り値 strcmp に準ずる

int dbCheckKeywd(dbRecord *rec, char *keywd);
一つのレコードの特定項目のデータ（配列）のいずれかにキーワードが含まれるかチェックする。
rec レコードの特定項目のアドレス
keywd キーワード
返り値 いずれかのデータにある:1　どのデータにもない:0

int dbCheckData(float *min, float *max, char *set);
マテリアルファイルの定義行を解析し、情報を取得する。
min 最小値（日数）　の格納先
max 最大値（日数）　の格納先
set 定義内容（数値等）の格納先
返り値 min-max あり：1、min-max なし:0、エラー時-1

void dbDaftarTexture(FILE *fp);
現在使用されているテクスチャの一覧表をファイルに出力する。
fp 出力先のファイル

(dbms.c)

void　　MaterialInit(dbMaterial *m);
dbMaterial 構造体をデフォルト値で初期化する。

void　　SetDiffuse(dbMaterial *m, dbMaterial *s);
マテリアルのカラー情報をコピーする。
m コピー先マテリアルのアドレス
s コピー元マテリアルのアドレス
※diffuse 拡散色(点光源や平行光源に対する拡散反射の色)

void　　SetAmbient(dbMaterial *);

マテリアルの拡散色から環境色を自動計算する。
m マテリアルのアドレス
※ambient 環境色（環境光に対する色）

void　SetSpecular(dbMaterial *m, dbMaterial *s;
マテリアルの鏡面色情報をコピーする。
m コピー先マテリアルのアドレス
s コピー元マテリアルのアドレス
※specular 鏡面色

void　SetEmission(dbMaterial *, dbMaterial *);
マテリアルの発光色情報をコピーする。
m コピー先マテリアルのアドレス
s コピー元マテリアルのアドレス
※emission 発光色

int　SetMaterial(dbMaterial *m, dbMaterial *s, char *str);
マテリアルファイルの１行分の解析を行う。
m 設定先マテリアルのアドレス
s コピー元マテリアルのアドレス
str 解析するマテリアルファイルの１行
返り値
DBERROR エラー
ENAK 内容充実
BUSUK 内容腐敗
NO コメント行、空白行
YES マテリアル名称が１カラムから定義された宣言行

void　CheckPrint(void);
現在選択されているデータベースをコンソールにデバッグ出力する(2.09 では非使用)。

Ｂ－４．レンダリングライブラリ（Ｇ３ＤＲＬ）

（１）システム関数

char *g3Version();
G3DRL のバージョンを取得する。
返り値 バージョンを示す文字列

（2）データ構築関数

int g3CombineGroup(d3Group *g, float value, float ratio);
グループ内の面を結合する(2.09 では、land.dll　に移管)
g グループのアドレス
value 値
ratio レート
返り値 成功時１　エラー時０

int g3CutGroup(d3Group *g, d3Group *setGrp,
　int count, double *points, int inside);
グループの形状（配下の面）を、面で切断する(2.09 では、land.dll に移管)。
g 処理するグループ（入力）
setGrp 処理結果を格納するグループ（出力）
count 切断面の頂点数
points 切断面座標
inside 内外選択フラグ（0:外側　1:内側）
返り値 成功時１　エラー時０

int g3DivtriGroup(d3Group *g, double area);
グループを三角形分割する(2.09 では、land.dll に移管)。
g グループのアドレス
area 三角形のエリアサイズ
返り値 成功時１　エラー時０

int g3CreateDrawarea();
新たな表示領域を作成する。
返り値 作成した表示エリアの通し番号

int g3sCreateDrawarea();
新たな補助表示領域を作成する。OpenGL 画面で、カラーやテクスチャ等の見出し画像を表示するための領域に用いる。
返り値 作成した表示エリアの通し番号

int g3CreateGroundPixel(double left, double right, double bottom, double top);
地面データのデプスピクセルを作成する。

left, right, bottom, top 地面データの範囲 (X,Y)
返り値 成功時１　エラー時０

int g3CreateGroundPixelByBBox();
バウンディングボックス内の地面データのデプスピクセルを保存する。
返り値 成功時１　エラー時０

int g3AplGroundDepth(g3Ground *grd);
グラウンドのデプスを作成する。
grd グラウンドのアドレス
返り値 成功時１　エラー時０

int g3AplGroundDepthNearFar(g3Ground *grd, double zmin, double zmax);
グラウンドのデプスを NearFar 内で作成する。
grd グラウンドのアドレス
zmin Near
zmax Far
返り値 成功時１　エラー時０

int g3AplEndDepth(g3Ground *grd);
高さを計測するためのデプスバッファに地面の属性のある地物を書き込む。
grd グラウンドのアドレス
返り値 成功時１　エラー時０

int g3AplStartDepth(g3Ground *grd, g3Ground *newgrd, int reverse);
高さを計測するためのデプスバッファを作成する。
grd 既存グラウンドのアドレス
newgrd 生成されているグラウンドのアドレス
reverse 上下の逆転フラグ　　0:高　　1:低
返り値 成功時１　エラー時０

void g3AplAddDepth(g3Face *f);
高さを計測するためのデプスバッファに面を追加記録する。
f 面のアドレス

void g3MakeRasterFont();

ラスターフォントを作成する。画面への寸法線書き込みに使用する。

int g3LoadMaterial(int id);
マテリアルを読み込む。
id マテリアル ID
返り値 成功時１　エラー時０

int g3LoadMaterialAll();
マテリアルを全て読み込む。
返り値 成功時１　エラー時０

int g3LoadTexture(int id);
テクスチャを読み込む。
id テクスチャ ID
返り値 成功時１　エラー時０

int g3LoadTextureAll();
テクスチャを全て読み込む。
返り値 成功時１　エラー時０

int g3PlaneMovePoint(d3Group *g, int type, double ext, double p[3],
  double moveP[3], double left, double right, double bottom, double top);
頂点を移動させる(2.09 では、land.dll に移管)。
g グループのアドレス
type 追従パターン
ext 波及範囲
p[3] 移動する点 (X,Y,Z)
moveP[3] 移動する量 (X,Y,Z)
返り値 成功時１　エラー時０

int g3HeightMovePoint(d3Group *g, double ext, double p[3], double h, double left,
  double right, double bottom, double top);
頂点を高さ移動させる(2.09 では、land.dll に移管)。
g グループのアドレス
ext 波及範囲
p[3] 移動する点 (X,Y,Z)

h 移動量(z)
left,right,bottom,top エリア
返り値 成功時１　エラー時０

```
int g3Road(
    int pointCount, double *points, double roadWidth,
    double cutHeight,double cutStepWidth, double cutSlope, int cutCount,
    double pileHeight,double pileStepWidth, double pileSlope, int pileCount,
    double cutC, double pileC, float roadColor[4], s3Image *texture,
    float cutColor[4], s3Image *cutTex,
    float cutStepColor[4], s3Image *cutStepTex,
    float pileColor[4], s3Image *pileTex,
    float pileStepColor[4], s3Image *pileStepTex,
    double *cutVolume, double *pileVolume) ;
```

道路法面を作成する(2.09 では、nori.dll に移管)。
pointCount 補間された道路軌跡点数
points 補間された道路軌跡座標(pointCount×3)
roadWidth 道路幅
cutHeight 切土高
cutStepWidth 切土小段幅
cutSlope 切土傾斜 (1:nan)
cutCount 切土小段数
pileHeight 盛土高
pileStepWidth 盛土高小段数
pileSlope 盛土高傾斜 (1:nan)
pileCount 盛土小段数
cutC 切土の土量変化率
pileC 盛土の土量変化率
roadColor[4] 道路色
texture 道路テクスチャ(NULL はテクスチャなし)
cutColor[4] 切土色
cutTex 切土テクスチャ(NULL はテクスチャなし)
cutStepColor[4] 切り土の小段色
cutStepTex 切土のテクスチャ(NULL はテクスチャなし)
pileColor[4] 盛土色
cutStepTex 盛土テクスチャ(NULL はテクスチャなし)

pileStepColor[4] 盛土テクスチャの小段色
pileStepTex 盛土テクスチャの小段テクスチャ
cutVolume 切土量（出力）
pileVolume 盛土量（出力）
返り値 成功時１　エラー時０
※道路軌跡線のループの場合は結果は保証されない

int g3Round(d3Group *g, double radius, int division, int ednum, double *points);
角の丸め処理をする
g グループのアドレス
radius 半径
division 分割数
ednum 頂点数（線データ）
points 角取り処理する辺（線データ）の頂点（X,Y,Z･･･の並び）
返り値 成功時１　エラー時０

void g3StencilPrepare();
ステンシル・バッファを使えるように準備する(2.09 では各関数で直接 gl 関数を使用するため、本関数は非使用)。

void g3StencilStart();
ステンシル・バッファの開始(2.09 では各関数で直接 gl 関数を使用するため、本関数は非使用)。

void g3StencilEnd();
ステンシルの終了(2.09 では各関数で直接 gl 関数を使用するため、本関数は非使用)。

int g3Tunnel(d3Group *g, int tCount,double *points, double *startP, double *endP);
トンネルを作成する(2.09 では、tunnel.dll に移管)。
g グループのアドレス
tCount 断面頂点数
points 断面頂点列（X,Y,Z･･･の並び）
startP 始点（X,Y,Z）
endP 終点（X,Y,Z）
返り値 成功時１　エラー時０

g3TsTop *g3LoadTsf(const char *filename);
テクスチャ自動貼り付け設定ファイルを読み込む。
filename ファイル名
返り値 g3TsTop のアドレス

（３）初期化関数

int g3Initialize();
表示領域登録用配列にメモリ・ブロックを割り当て、初期化する。
返り値 成功時１　エラー時０

int g3InitializeWindow();
表示エリアを初期化する。
返り値 成功時１　エラー時０

int g3sInitializeWindowColorSashArea();
マテリアルを一覧するウインドウを初期化する
返り値 成功時１　エラー時０

int g3sInitializeWindowLightArea();
光源を表示する色球を表示するウインドウを初期化する
返り値 成功時１　エラー時０

int g3sInitializeWindowMaterialArea();
マテリアル・カラーを表示するウインドウを初期化する
返り値 成功時１　エラー時０

int g3sInitializeWindowMaterialArea2();
グラフィックなマテリアル表示を行うウインドウを初期化する
返り値 成功時１　エラー時０

int g3sInitializeWindowPlainMaterialArea();
グラフィックなマテリアル表示を行うウインドウを初期化する
返り値 成功時１　エラー時０

int g3sInitializeWindowsSteelArea();
型鋼を表示するウインドウを初期化する

返り値 成功時１　エラー時０

int g3sInitializeWindowTextureArea();
グラフィックにテクスチャを編集するウインドウを初期化する
返り値 成功時１　エラー時０

（４）データ定義/更新関数

int g3SetDrawarea(int num);
表示エリアを初期化する。
num 表示エリアの通し番号
返り値 成功時１　エラー時０

void g3SetColor(int ix, float r, float g, float b, float a);
表示画面の様々な部分に色設定を行う。
ix 設定対象部分の指定
 G3_COL_CLEAR クリアカラー(0,0,0,1) 地物が無い部分の画面色
 G3_COL_GRID グリッドカラー(1,1,1,1) グリッド（格子）の線の色
 G3_COL_MEASURE メジャーカラー(1,1,1,1) 縮尺の色
 G3_COL_AXIS_X 座標軸ｘ(1,0,0,1) Ｘ軸の色
 G3_COL_AXIS_Y 座標軸ｙ(0,1,0,1) Ｙ軸の色
 G3_COL_AXIS_Z 座標軸ｚ(0,0,1,1) Ｚ軸の色
 G3_COL_ORBITPOINT 軌跡ｘ点(1,1,1,1) 移動経路のコントロールポイント
 G3_COL_ORBITLINE 軌跡線(1,1,1,1) 移動経路の折れ線・曲線
 G3_COL_HILIGHT 強調表示線の色(1,0,0,1) 選択したオブジェクトの識別に用いる
 G3_COL_CAMERAMARK カメラマーク(1,1,1,1) 視点設定におけるカメラのアイコン
（ ）内はデフォルト値
※各ixで（ ）内はデフォルトカラー値でr,g,b,a 色の値で0.0-1.0の値で
　ある

void g3Enable(int ix);
表示における様々なモードを有効にする。
ix モードの設定対象を指定する
　G3_DRA_TEXTURE テクスチャ表示モード（Enable）
　G3_DRA_ANTIALIAS アンチエリアシング　（Disale）
　G3_DRA_GRID グリッド表示（Disable)、オルソだけ
　G3_DRA_AXIS 座標軸（Disable）

G3_DRA_LIGHT 光源 (Enable)
G3_DRA_BACKIMAGE 背景画像表示 (Enable)
G3_DRA_FRONTIMAGE 前景画像表示 (Enable)
G3_DRA_CAMERAMARK カメラマーク (Disable)、オルソだけ
G3_DRA_MEASURE
メジャー (Disable)、オルソだけ画面の左隅に横方向の縮尺を表示する。
画面の約 10%の切りのよい長さ(1.5,10,50･･･)を線と数字で表示する。
G3_DRA_ORBIT 軌跡 (Disable)、オルソだけ
G3_DRA_HILIGHT 強調表示 (Disable)
G3_DRA_RECT 矩形表示 (Disable)、オルソだけ
G3_DRA_WIRE 線画表示 (Disable=shading)
G3_DRA_VISUAL 可視範囲表示 (Disable)
オルソの平面の場合(G3_CAM_TOP)だけ有意
( ) 内はデフォルト値
※モードが有効になっていても、g3Set 関数で必要な情報が設定されていないと無効

void g3Disable(int ix);
表示モードを無効にする。
ix モードの設定対象の指定
G3_DRA_TEXTURE テクスチャ表示モード (Enable)
G3_DRA_ANTIALIAS アンチエリアシング (Disale)
G3_DRA_GRID グリッド表示 (Disable)、オルソだけ
G3_DRA_AXIS 座標軸 (Disable)
G3_DRA_LIGHT 光源 (Enable)
G3_DRA_BACKIMAGE 背景画像表示 (Enable)
G3_DRA_FRONTIMAGE 前景画像表示 (Enable)
G3_DRA_CAMERAMARK カメラマーク (Disable)、オルソだけ
G3_DRA_MEASURE
メジャー (Disable)、オルソだけ画面の左隅に横方向の縮尺を表示する。
画面の約 10%の切りのよい長さ(1.5,10,50･･･)を線と数字で表示する。
G3_DRA_ORBIT 軌跡 (Disable)、オルソだけ
G3_DRA_HILIGHT 強調表示 (Disable)
G3_DRA_RECT 矩形表示 (Disable)、オルソだけ
G3_DRA_WIRE 線画表示 (Disable=shading)
G3_DRA_VISUAL 可視範囲表示 (Disable)
オルソの平面(G3_CAM_TOP)だけ

（ ）内はデフォルト値
※モードが有効になっていても、g3Set 関数で必要な情報が設定されていないと無効

void g3DrawImage2D(dbImage *img);
イメージを描画する。
img イメージ情報のアドレス

void g3SetCameraOrtho(int kind, g3Ortho *ortho);
オルソビューの視点情報をセットする。
kind
  G3_CAM_FRONT zx 平面：南立面図（正面図）
  G3_CAM_TOP zy 平面：平面図
  G3_CAM_SIDE yz 平面：東立面図（側面図）
  G3_CAM_UTARA -xz 平面：北立面図
  G3_CAM_BARAT -yz 平面：西立面図
ortho オルソ構造体のアドレス
ortho はコピー（メモリ・ブロック）を作成した上で表示画面の属性として管理する。pers と ortho は別個に管理される。

void g3SetCameraPers(s3Camera *pers);
パースビュー（透視図）の視点情報をセットする。
pers パース構造体のアドレス
pers は、コピー（メモリ・ブロック）を作成した上で表示画面の属性として管理する。pers と ortho は別個に管理される。

void g3SetCameraOptimize();
現在のデータの最大最小を調べて最適視点を設定する。

void g3SetCameraChangeOptimize(int kind);
表示画法（パース、各種オルソ）を変更した上で、最適化した視点を設定する。
視点パラメータを計算するのが面倒な場合に使える。
kind
  G3_CAM_FRONT zx 平面：南立面図（正面図）
  G3_CAM_TOP zy 平面：平面図
  G3_CAM_SIDE yz 平面：東立面図（側面図）
  G3_CAM_UTARA -xz 平面：北立面図

G3_CAM_BARAT -yz 平面：西立面図

void g3SetCameraZoom(double rate);
視点をズームさせる。
rate ズーム率（1.0 はズームなし）
透視図の場合は、視点と注視点を結ぶ直線に沿って視点を接近または後退させる。
オルソの場合は、left,right,top,bottom を変更し、表示の範囲と縮尺を変える。

void g3SetCameraHorizontalRound(double rate);
透視図における視点を横方向回転(注視点を中心として視点を回転移動)する。
rate 回転角度（360.0 は 1 回転して元に戻る）
rate＞0 は右回りを示す

void g3SetCameraVerticalRound(double rate);
透視図における視点の縦方向回転（注視点を中心として視点を回転移動）する。
rate 回転角度（360.0 は 1 回転して元に戻る）
rate＞0 は右回りを示す

void g3SetCameraHorizontalPutar(double rate);
透視図の注視点を、視点を中心に横方向回転する（パン)。
rate 回転角度（360.0 は 1 回転して元に戻る）
rate＞0 は右回りを示す

void g3SetCameraVerticalPutar(double rate);
透視図の注視点を、視点を中心に縦方向回転する。
rate 回転角度（360.0 は 1 回転して元に戻る）
rate＞0 は右回りを示す

void g3SetCameraHorizontalShift(double rate);
視点を横にシフト（平行移動）する。
rate シフト率
=0 はシフトなし
＞0 は右または上シフトである
※透視図の場合は視点、注視点を横方向に、視点と注視点の距離を 1.0 としたときの rate 分シフトする。
※オルソの場合は、left,right 間の距離を 1.0 としたときの rate を指定する 。

void g3SetCameraVerticalShift(double rate);
視点を縦シフトする。
rate シフト率
=0 はシフトなし
＞0 は右または上シフトである
※パースの場合は視点、注視点をアップベクトル方向（縦)、視点と注視点の距離を 1.0 としたときの rate 分シフトする。
※オルソの場合は、top,bottom 間の距離を 1.0 としたときの rate を指定する。

void g3SetLightGroup(s3LightGroup *lg);
光源グループを表示画面に対してセットする。
lg 光源グループのアドレス
※ 光源グループ lg 及びその配下の光源ユニットをコピーしたメモリ･ブロックを作成し、セットする。lg が、既に存在する場合には、これを解放した上でコピーをセットする。但し、既存の lg が、新たにセットしようとする lg と同じものであった場合には、解放せずにコピーをセットする。

void g3SetBackImage(s3Image *img);
背景イメージをセットする。
img イメージ情報のアドレス

void g3SetFrontImage(s3Image *img);
前景イメージをセットする。
img イメージ情報のアドレス

void g3SetRootGroup(int count, d3Group **root);
ルートグループをセットする。
count ルートグループ数
root グループ配列のアドレス

int g3CalcBoundingBox();
登録されているルートグループ全体のバウンディングボックス、即ち地物全体の存在範囲（３軸最小最大値）を計算する。結果は選択されている表示画面の属性(da[cd].bbox)として保存される。
返り値 成功時１　エラー時０

※ルートグループ以下が変更された場合（形状変更、配置変更、削除など）は本関数を呼ぶこと。
※バウンディングボックスにより、最適視点設定が可能になる。

void g3SetBoundingBox(double min[3], double max[3]);
バウンディングボックスをセットする。
min[3] 最小値(X,Y,Z)
max[3] 最大値(X,Y,Z)

void g3SetCameramark(s3Camera *cam);
カメラ情報からカメラマークに必要な情報がセットされる
cam カメラ情報のアドレス
※カメラマークは、編集画面上において視点位置を示すための一種のアイコンである。

void g3SetCameramarkFromPers();
パース視点情報からカメラマークに必要な情報がセットされる

int g3SetTime(float tim);
時間をセットする。
tim 日数
返り値 成功時１　エラー時０

void g3SetScene(s3Scene *scn);
指定したシーン構造体から必要な情報を、現在選択されている表示画面にセットする。
scn シーンのアドレス（通常は、シーン配列から選択された１要素）

void g3Light();
現在選択されている表示画面に設定されている光源グループを、OpenGL の関数として表示条件として出力する。光源グループが存在しなければデフォルトの光源をセットする(2.09では非使用)。

int g3SelPoint(float sx, float sy, int seltype);
スクリーン座標上の位置と指定選択方法に基づき、全地物を検査して、結果を格納するg3SelPath 構造体を構築し、表示画面の属性(da[ce].pathList)に格納する。
sx,sy スクリーン座標
seltype　選択方法

G3_SET_GROUP　グループを選択する
G3_SET_FACE　面を選択する
G3_SEL_SAMECOLFACE_IN_GROUP　同色面を選択する
返り値 ヒット数（同じ平面位置に重なり合っている三次元オブジェクト数）
※この関数により作成された情報は、g3GetSelpath 関数により取り出され、利用される。

int g3UpSelPath(g3SelPath *spath);
選択対象のグループのレベルを一つ上(親グループ)に上げる。
spath 選択パス情報のアドレス
返り値　成功時１　エラー時０

int g3DownSelPath(g3SelPath *spath);
選択対象のグループのレベルを一つ下（子グループ）に下げる。
spath 選択パス情報のアドレス
返り値 成功時１　エラー時０

int g3SetSingleHilight(g3SelPath *spath);
１つのオブジェクトをハイライト表示するようにセットする。
spath 選択パス情報のアドレス
返り値 成功時１　エラー時０

int g3AddHilight(g3SelPath *spath);
ハイライト表示するオブジェクトを追加する。
spath 選択パス情報のアドレス
返り値 成功時１　エラー時０

void g3SetSize(int width, int height);
表示エリアのサイズを、表示画面の属性(da[cd].width, height)にセットし、OpenGL の描画条件として設定する。
width 幅
height 高さ
※ユーザーが画面サイズを変更した場合に、OpenGL の表示条件をこれに合わせることにより、適切な表示を行うことができる。

void g3SetGridsize(double interval);
格子間隔をセットする。

interval 格子間隔（0.0以上、デフォルト1.0

void g3SetTexMatrix(d3Matrix texm);
テクスチャマトリクスをセットする。
texm マトリクス

void g3SetOrbitPointHilight(int ix);
軌跡点のハイライトをセットする。画像視点抽出において、計算誤差の大きかった参照点を表示する場合などに使用する。
ix ハイライト表示する軌跡点の通し番号

void g3SetOrbitPoints(int count, double *points, double pointSize);
軌跡点をセットする。×印で表示される。
count 点数
points 点の座標（X,Y,Z･･･の並び）
pointSize 軌跡点（×印）の描画サイズ
pointsはポインタだけが保存される

void g3SetOrbitLine(int count, double *points);
軌跡線をセットする。
count 点数
points 点の座標（X,Y,Z･･･の並び）
pointsはポインタだけが保存される
※曲線補間がされていること

void g3SetVisualArea(double left, double right, double bottom, double top, s3Image * texture);
可視範囲をセットする。
left,right,bottom,top 可視範囲のワールド座標(x,y)
texture 可視分布のテクスチャデータ
※テクスチャファイル名は不要
※アルファを1.0未満にすると半透明になる
※NULLの場合は、可視範囲が線画で表示される

void g3SetAntiAliasCount(int count);
アンチエリアシングの精度をセットする。

count 精度

int g3CalcGroupBoundingBox(d3Group *g);
グループのバウンディングボックスを計算する。
g グループのアドレス
返り値 成功時１　エラー時０

double g3CalcBoundingSphere(d3Group *group);
バウンディングスフィアーを計算する。
group グループのアドレス
返り値 半径

void g3PrintString(char *s);
文字（ラスターフォント）を出力する。縮尺の表示などに用いる。
s 表示する文字

void g3NormalizeTexture4(int *width, int *height, unsigned long **pixels);
テクスチャをノーマライズする。
width 幅の格納先
height 高さの格納先
pixels 処理結果のピクセル配列の格納先アドレス
※幅と高さを２のべき乗の値にする。

int g3AutoTextureFace(d3Face *f, g3TsDetail *detail, d3Matrix pathMat, double uvorigin[3], double udir[3], double uScale, double vScale);
面にテクスチャを自動貼り付けする。
f 面のアドレス
detail g3TsDetail のアドレス
pathMat パスマトリクス
uvorigin[3] uv 原点 (X,Y,Z)
udir[3] uv ベクトル (X,Y,Z)
uScale u スケール
vScale v スケール
返り値 成功時１　エラー時０

int g3AutoTextureGroup(d3Group *g, g3TsDetail *detail, d3Matrix pathMat,

double uvorigin[3], double udir[3], double uScale, double vScale);
グループにテクスチャを自動貼り付けする。
g グループのアドレス
detail g3TsDetail のアドレス
pathMat パスマトリクス
uvorigin[3] uv 原点 (X,Y,Z)
udir[3] uv ベクトル (X,Y,Z)
uScale u スケール
vScale v スケール
返り値 成功時 1　エラー時 0

void g3SetShadowLength(double len)
影計算において作成する影立体の長さの設定値を設定する。
len 影立体の長さ

void g3SetShadowMode(int mode);
影計算における原因物体設定モードを設定する。
mode 設定モード

```
G3_SHADOW_ALL                1  /* 全ての物体*/
G3_SHADOW_OTHER_GROUND       2  /* 地面以外の物体*/
G3_SHADOW_ONE                3  /* 選択した1つの物体*/
```

void g3SetShadowLightMode(int mode);
影計算における光源の設定モードを設定する。
mode 設定モード

```
G3_SHADOW_OTHERLIGHT_OFF    10  LIGHT0以外の光源をOFF
G3_SHADOW_OTHERLIGHT_ON     11  LIGHT0以外の光源をON
```

（５）データ取得関数

int g3GetDrawable(int ix);
指定した表示モードの有効/無効を取得する。
Ix　取得する表示モード
G3_DRA_TEXTURE テクスチャ表示モード (Enable)
G3_DRA_ANTIALIAS アンチエリアシング　(Disale)
G3_DRA_GRID グリッド表示 (Disable)、オルソだけ
G3_DRA_AXIS 座標軸 (Disable)

G3_DRA_LIGHT 光源 (Enable)
G3_DRA_BACKIMAGE 背景画像表示 (Enable)
G3_DRA_FRONTIMAGE 前景画像表示 (Enable)
G3_DRA_CAMERAMARK カメラマーク (Disable)、オルソだけ
G3_DRA_MEASURE
メジャー (Disable)、オルソだけ画面の左隅に横方向の縮尺を表示する。
画面の約 10%の切りのよい長さ(1.5,10,50･･･)を線と数字で表示する。
G3_DRA_ORBIT 軌跡 (Disable)、オルソだけ
G3_DRA_HILIGHT 強調表示 (Disable)
G3_DRA_RECT 矩形表示 (Disable)、オルソだけ
G3_DRA_WIRE 線画表示 (Disable=shading)
G3_DRA_VISUAL 可視範囲表示 (Disable)
オルソの平面(G3_CAM_TOP)だけ
( ) 内はデフォルト値
モードが有効になっていても、g3Set 関数で必要な情報が設定されていない場合は無視される
返り値 有効１　無効０

int g3GetCameraOrtho(int *kind, g3Ortho *ortho);
オルソビューの視点情報を取得する。
kind
G3_CAM_FRONT zx 平面：南立面図（正面図）
G3_CAM_TOP zy 平面：平面図
G3_CAM_SIDE yz 平面：東立面図（側面図）
G3_CAM_UTARA -xz 平面：北立面図
G3_CAM_BARAT -yz 平面：西立面図
ortho オルソのアドレス
返り値 成功時１　エラー時０
保存領域をコピーして返すので、ポインタを示すフィールドを直接変更した場合は、Ｓｅｔ関数を呼ぶ必要がある。

int g3GetCameraPers(s3Camera *pers);
透視図画面のカメラ情報を取得する。
pers カメラ情報の格納先
返り値 成功時１　エラー時０
構造体の内容を格納先にコピーするので、内容を変更した結果を表示に反映させるために

は、g3SetCameraPers 関数を用いる必要がある。

int g3GetBackImage(s3Image **img);
背景イメージを取得する。
img イメージ情報のアドレスを格納するアドレス
返り値 成功時 1　エラー時 0

int g3GetFrontImage(s3Image **img);
前景イメージを取得する。
img イメージ情報のアドレスを格納するアドレス
返り値 成功時 1　エラー時 0

int g3GetRootGroup(int *count, d3Group ***root);
ルートグループを取得する。
count ルートグループ数の格納先
root グループ配列のアドレスの格納先
返り値 成功時 1　エラー時 0

int g3GetBoundingBox(double min[3], double max[3]);
グループの属性として記録されているバウンディングボックスを取得する。
min[3] 最小値(X,Y,Z)
max[3] 最大値(X,Y,Z)
返り値 成功時 1　エラー時 0

int g3GetGroupLocalCenter(d3Group *g, double center[3]);
グループのバウンディングボックスの中心（ローカル座標）を取得する。
g グループのアドレス
center[3] 中心(X,Y,Z)
返り値 成功時 1　エラー時 0

int g3GetGroupWorldCenter(g3SelPath *spath, double center[3]);
グループのバウンディングボックスの中心（ワールド座標）を取得する。
spath 選択パス情報のアドレス
center[3] 中心(X,Y,Z)
返り値 成功時 1　エラー時 0

int g3GetTime(float *tim);
時間を取得する。
tim 時間（日数を浮動小数で表現）を格納するアドレス
返り値 成功時１　エラー時０

int g3GetSelPath(int num, g3SelPath **spath);
選択パス情報を取得する。
num セレクト番号(g3SelPoint の返り値以内の数)を指定
spath 選択パス情報のアドレスの格納先
返り値 成功時１　エラー時０
※セレクト番号は、同じ位置に重なっている複数のオブジェクトのいずれかを選択する。

int g3GetHilight();
ハイライト表示されているオブジェクトの数を取得する。
返り値 ハイライト数(da[cd].hilightCount)

int g3GetSize(int *width, int *height);
表示エリアのサイズを取得する。
width 幅
height 高さ
返り値 成功時１　エラー時０

int g3GetGridsize(double *interval);
格子間隔を取得する。
interval 格子間隔（0.0 以上、デフォルト 1.0）のアドレス
返り値 成功時１　エラー時０

int g3GetTexMatrix(d3Matrix texm);
表示画面に定義されているテクスチャマトリクスを取得する。
texm マトリクス(長さ 16 の倍精度浮動小数の配列)
返り値 成功時１　エラー時０

int g3GetOrbitPoints(int *count, double **points, double *pointSize);
軌跡点のデータを取得する。
count 点の総数の格納先
points 点の座標（X,Y,Z･･･の並び）の配列の格納先

pointSize ×印のサイズの格納先
返り値 成功時１　エラー時０

int g3GetOrbitLine(int *count, double **points);
軌跡線のデータ取得を行う。曲線補間がされていること。
count 点の数の格納先
points 点の座標 (X,Y,Z･･･の並び) の配列の格納先
返り値 成功時１　エラー時０

int g3GetVisualArea(double *left, double *right, double *bottom, double *top,
  s3Image **texture);
可視範囲を取得する。
left, right,　 bottom, top 可視範囲の矩形ワールド座標(x,y)の格納先
texture 可視分布のテクスチャデータのアドレスの格納先
返り値 成功時１　エラー時０

int g3GetAntiAliasCount(int *count, int *maxCount);
アンチエリアシングの精度を取得する
count 精度の格納先
maxCount 最大精度の格納先
返り値 成功時１　エラー時０

double g3AplHeight(g3Ground *grd, double x, double y);
任意の場所の地面の高さを取得する。
grd 地面の高さを記述するメッシュデータのアドレス
x,y 高さを取得したい位置
返り値 高さ
※メッシュ格子点の高さから補間計算を行う

double g3GetGroundHeight(double x, double y);
設定されているメッシュデータを用いて任意の場所の地面の高さを取得する。
x,y 位置
返り値 地面の高さ

double g3GetVisualRate(double eye[3], double ref[3], g3SelPath *spath);
視点と、選択されたグループから可視率を取得する。

eye[3] 視点(X,Y,Z)
ref[3] 注視点(X,Y,Z)
spath 対象グループを特定する選択パスのアドレス
返り値 可視率
※視点・注視点を用いて、可視率計算用画面に、選択対象物のみを描いた場合と、全地物を描いた場合の、対象物に係るピクセル数をカウントし、対象物以外の地物で隠されない部分の比率を計算する。

void g3ReadPixelDrawarea(int sx, int sy, int width, int height, int format, int type, void *pixels);
指定した長方形領域のピクセルを配列に取得する。
sx,sy スタート座標
width 幅
height 高さ
format フォーマット（取得したい色情報の形式：GL_BGRA 等）
type タイプ（読み取ったデータを保存する配列の型：GL_UNSIGNED_BYTE 等）
pixels ピクセルの格納先アドレス
glRead Pixels()に準ずる

void g3GetVisualHeight(d3Group *g, double left, double right, double bottom, double top, double *min_z, double *max_z);
高さ方向の可視範囲を取得する（2.09 では廃止)。
g グループのアドレス t
left, right, bottom, top エリア
min_z 高さの最小値
max_z 高さの最大値

int g3GetPlanePoints(d3Group *g, double top, double bottom, double left, double right, double z, int *count, double ***points);
指定した高さの面の頂点を取得する(2.09 では、land.dll に移管)。
g グループのアドレス t
left, right, bottom, top エリア
z 高さ
count 頂点数のアドレス
points 頂点（X,Y,Z･･･の並び）
返り値 成功時 1　エラー時 0

int g3GetDivision(d3Group *g, double v1[3],double v2[3], int num, int *division);
丸めの分割数を取得する(2.09 では非使用)。
g グループのアドレス
v1[3] 丸め始めの点 (X,Y,Z)
v2[3] 丸め終りの点 (X,Y,Z)
num 円の分割数
division 分割数のアドレス
返り値 成功時 1　エラー時 0

void g3GetLightGroup(s3LightGroup **lg);
表示画面に定義されている光源グループのアドレスを取得する。
lg 光源グループのアドレス格納先

double g3GetShadowLength();
影計算において作成する影立体の長さの設定値を取得する。
返り値 影立体の長さ

int g3GetShadowMode();
影計算における原因物体設定モードを取得する。
返り値 設定モード
G3_SHADOW_ALL　　1　/* 全ての物体*/
G3_SHADOW_OTHER_GROUND　　2　/* 地面以外の物体*/
G3_SHADOW_ONE　　3　/* 選択した1つの物体*/

int g3GetShadowLightMode();
影計算における光源の設定モードを取得する。
返り値 設定モード
G3_SHADOW_OTHERLIGHT_OFF　10　LIGHT0以外の光源をOFF
G3_SHADOW_OTHERLIGHT_ON　11　LIGHT0以外の光源をON

int g3GetStereoPers(double eye, s3Camera *PersSave, s3Camera* leftPers, s3Camera *rightPers);
ステレオ表示のパラメータを取得する。
eye 左右間隔を指定する
PersSave 基準となるカメラ情報（入力）

leftPers 計算した左眼のカメラ情報（出力）
rightPers 計算した右眼のカメラ情報（出力）
返り値　成功時１　失敗時０

void g3ReadEyeParameter();
設定ファイル eye_param.set からステレオ表示に係るパラメータを取得する。

void g3sGetLightColor(float *c);
補助描画領域の光源色を取得する。
c　光源色を格納する配列のアドレス

void g3sGetLightPosition(float *pos);
補助描画領域の光源位置を取得する。
pos 光源位置座標を格納する配列のアドレス

dbMaterial* g3sGetMaterial(void);
補助描画領域のマテリアルを取得する。
返り値　マテリアルを記述する構造体のアドレス

void g3sGetSizeSteelArea(int *width, int *height);
型鋼の見出し画像を描画する領域の幅と高さを取得する。

（６）データ削除関数

void g3DeleteDrawarea(int num);
表示エリアを削除する。

void g3sDeleteDrawarea(int num);
補助表示エリアを削除する。

void g3Clear();
背景色でクリアする。デプスバッファもクリアする。

void g3ClearScene();
カメラ情報、光源、イメージ、ルートグループをクリアする。

void g3ClearHilight();

ハイライト数を 0 にする。
強調表示する場合は、g3Enable(G3_DRA_HILIGHT)を呼ぶこと。

void g3ClearOrbitPointHilight()
軌跡点のハイライトをクリアする。

void g3AplFreeDepth(g3Ground *grd);
地面の高さを求めるためのデプスバッファ（メッシュ情報）をクリアする
grd メッシュデータのアドレス

void g3ReleaseGroundPixel();
現在の地面の高さ情報をクリアする。

void g3Bebas();
表示領域の配列（メモリ・ブロック）を削除する。

void g3BebasPathList();
選択された表示領域の、オブジェクト選択状態を示すデータ（PathList）を削除する。

（7）表示関数

void g3Draw();
全地物を表示する。

void g3sColorSashareaDraw();
グラフィックなマテリアル編集の見出し画像領域を描画する。一つの長方形エリアの中に、
全てのマテリアルを短冊状に表示する。

void g3sLightAreaDraw();
光源ユニットの色を表示する補助画面を描画する。

void g3sMaterialAreaDraw();
マテリアル編集画面（オリジナル）の設定カラーを表示する補助画面を描画する。

void g3sMaterialAreaDraw2();
グラフィックなマテリアル編集画面の設定カラーを表示する補助画面を描画する。

void g3sPlainMaterialAreaDraw();
グラフィックなマテリアル編集画面の選択候補マテリアル群を平面表示する補助画面を描画する。

void g3sSteelSectionDraw(int type);
型鋼の種類別パラメータの説明画面を描画する。
type 型鋼の種類
 G3_HSTEEL　　0
 G3_CSTEEL　　1
 G3_TSTEEL　　2
 G3_LSTEEL　　3

void g3Resize(int width, int height);
表示エリアのリサイズをする
width 幅
height 高さ

void g3ClearRectScreen();
矩形ライン表示をクリアする

void g3DrawRectScreen(float sx, float sy, float ex, float ey);
正規化されたスクリーン座標で矩形ラインを表示する
sx,sy スタート（表示エリアを 1.0 に正規化した座標）
ex,ey エンド（表示エリアを 1.0 に正規化した座標）
フロントバッファに直接描画するので、wg3Swapbuffers は呼んではいけない。
ラバーバンド表示用である

（8）ウインドウ操作関数

int wg3EntryDrawarea(void *w, void *c);
表示エリアの生成
w ウィジェットクラスのポインタ(Windows 系では、HDC を格納する変数のアドレス)
c コンテキスト(Windows 系では HGLRC を格納する変数のアドレス)
返り値 成功時 1　エラー時 0（意味は、制限個数(10)オーバーのエラー）

int wg3ReentryDrawarea(void *oldw, void *neww, void *newc);
表示エリアを作り直した場合、例えば singlebuffer と doublebuffer の切り替えなどで、

旧情報(g3Set 関数)をキープしたままウィジェットだけを交換する
以降は、新しいウィジェット(neww)でアクセス可能になる
oldw 古いウィジェットクラスのアドレス(Windows 系では､HDC を格納する変数のアドレス)
neww 新しいウィジェットクラスのアドレス(同上)
newc 新しいコンテキスト(Windows 系では HGLRC を格納する変数のアドレス)
返り値 成功時 1　エラー時 0（意味は、oldw が未登録のエラー）

void wg3DeleteDrawarea(void *w);
表示エリアを削除する。
w ウィジェットクラスのアドレス(Windows 系では、HDC を格納する変数のアドレス)

int wg3AssignDrawarea(void *w);
表示エリアを指定する。以後の処理はこの表示画面に対して行われる。
w ウィジェットクラスのアドレス(Windows 系では、HDC を格納する変数のアドレス)
返り値 成功時 1　エラー時 0

void wg3Swapbuffers();
バッファをスワップする。

void wg3SetRedrawFlag(int flag);
再表示フラグの ON/OFF をする。
flag フラグ (ON:1/OFF:0)
フラグが ON のとき通常通り再表示し、OFF のとき再表示しない

void g3SetCameraHorizontalPutar(double rate);
視点を水平回転（パン）する。
rate 回転角

int g3sSetDrawarea(int num);
処理対象とする補助描画画面を選択する。
num 対象とする補助描画画面の番号
返り値 成功時 TRUE　失敗時 FALSE

void g3sSetLightColor(float *c);
補助描画領域の光源色を設定する。
c　光源色の配列

void g3sSetLightPosition(float *pos);
補助描画領域の光源位置を設定する。
pos 光源位置座標の配列

void g3sSetMaterial(dbMaterial *m);
補助描画領域のマテリアルを設定する。

void g3sSetTexture(dbTexture *tex);
補助描画領域のテクスチャを設定する。

void g3sSetMaterialFile(char *name);
補助描画領域にマテリアルファイル名をコピーしたメモリ・ブロックを設定する。
name マテリアルファイル名の文字列

g3sSetSize(int width, int height)
選択されている補助描画領域のサイズを設定する。
width 幅
height 高さ

int g3sSetSizeSteelArea(int width, int height);
型鋼の見出し画像を描画する領域の幅と高さを設定する。
width 幅
height 高さ

（9）その他関数

void g3TransformScreen2World(int sx, int sy, double *x, double *y, double *z);
スクリーン座標からワールド座標へ変換する。
sx, sy スクリーン座標（左上を(0,0)、右下を(width-1,height-1）とした整数値
x, y, z ワールド座標のアドレス（奥行き方向の値は0.0で返す）
※オルソのみ

Ｂ－５．インタプリタ（ＩＰ）

（1）データ構築関数

d3Group *i3LoadLssg(const char *gname, const char *file);

ジオメトリファイル(LSS-G 形式)を読み込む。
gname グループ名称
file ファイル名
返り値 グループのアドレス

d3Group *i3LoadTmpGroup(const char *file);
一時的ファイルを読み込む。ファイル中の最上位のグループを返り値とする。
file ファイル名
返り値 グループのアドレス
※最上位グループは、上方リンクが存在しないことをもって識別している

void i3CallCube(char *name, double x, double y, double z, double lx, double ly, double lz);
外部コマンド（直方体）を実行する(2.09 では非使用)。
name グループ名称
x, y, z 物体原点(X、Y、Z)
lx, ly, lz 幅、奥行き、高さ

void i3CallSphere(char *name, double x, double y, double z, double r);
外部コマンド（球）を実行する(2.09 では非使用)。
name グループ名称
x , y, z 物体原点
r 半径

void i3CallCylinder(char *name, double x1, double y1, double z1, double x2, double y2,
  double z2, double r);
外部コマンド（円柱）を実行する(2.09 では非使用)。
name グループ名称
x1, y1, z1 下円中心
x2, y2, z2 上円中心
r 半径

void i3CallCone(char *name, double x1, double y1, double z1, double x2, double y2,
  double z2, double r1, double r2);
外部コマンド（円錐、円錐台）を実行する(2.09 では非使用)。

name グループ名称
x1, y1, z1 下円中心
x2, y2, z2 上円中心
r1, r2 下円半径、上円半径

void i3CallFlatCylinder(char *name, double x1, double y1, double z1, double x2,
  double y2, double z2, double r, int n);
外部コマンド（角柱）を実行する(2.09 では非使用)。
name グループ名称
x1, y1, z1 下面中心
x2, y2, z2 上面中心
r 中心から頂点の長さ
n 角数

void i3CallFlatCone(char *name, double x1, double y1, double z1, double x2, double
y2,
  double z2, double r1, double r2, int n);
外部コマンド（角錐、角錐台）を実行する(2.09 では非使用)。
name グループ名称
x1, y1, z1 下面中心
x2, y2, z2 上面中心
r1, r2 下面から頂点の長さ、上面中心から頂点までの長さ
n 角数

void i3CallCSteel(char *name, double a, double b, double c, double d, double h);
外部コマンド（溝形鋼）を実行する(2.09 では非使用)。
name グループ名称
a, b, c, d, h 高さ , 幅、厚み、縦板の厚み、長さ

void i3CallTSteel(char *name, double a, double b, double c, double d, double h);
外部コマンド（T型鋼）を実行する(2.09 では非使用)。
name グループ名称
a, b, c, d, h 高さ , 幅、厚み、縦板の厚み、長さ

void i3CallHSteel(char *name, double a, double b, double c, double d, double h);
外部コマンド（H型鋼）を実行する(2.09 では非使用)。

name グループ名称
a, b, c, d, h 高さ , 幅、厚み、縦板の厚み、長さ

void i3CallLSteel(char *name, double a, double b, double c, double d, double h, double x, double y);
外部コマンド（Ｌ型鋼）を実行する(2.09 では非使用)。
name グループ名称
a, b, c, d, h 高さ , 幅、厚み、縦板の厚み、長さ
x, y 原点からの離れ

void i3CallSweep1 (char *name, char *face, char *line);
掃引体１面を生成する(2.09 では非使用)。
name グループ名
face 断面ファイル名
line 中心線軌跡ファイル名

void i3CallSweep2 (char *name, char *face1, char *face2);
掃引体２面を生成する(2.09 では非使用)。
name グループ名
face1 断面ファイル名
face2 断面ファイル名
※Ver.2.09 においては、外部関数の起動は、原始図形とユーザー定義のパラメトリック部品を共通化したため、以上の各原始図形専用の形状生成関数は使用しない。

int IP_interpret(char *command);
command を解析し、DML ライブラリを用いてデータを構築・取得する。
command コマンド文字列
※インタプリタ（ＩＰ）の処理ループは、データファイルの一行である。一方、コマンドの実行単位は、セミコロン「;」を区切りとしている。従って、本関数の呼び出しは、複数行にまたがるコマンドの一部である場合（例１）も、また複数のコマンドを含む１行である場合（例２）もある。

例１：（１回目）G001=
（２回目）face(V01,V02,V03,
（３回目）V04,V05,V06);
この場合、１回目と２回目は、コマンドをスタックする。

３回目の';'で初めてコマンドを実行する。

例２：　V01=COORD(0, 0, 0,); V02=COORD(1, 0, 0);
内部ループで２つのコマンドを実行する。

FILE *i3_outputOpen(const char *filename);
出力ファイル(LSS-S または LSS-G)をオープンする。
filename ファイルのパス名
返り値 ファイルのアドレス(NULL は失敗)
※冒頭行にバージョンを表記するコメント行を出力する。
最適化保存が選択されている場合には、テンプレートとなる要素定義を出力する。

void i3_outputClose(FILE *fp);
出力ファイルをクローズする。
fp ファイルポインタ
※最適化保存を行った場合には、末尾にサマリーをコメント行として追加する。

（２）データ定義/更新関数

void i3UpdateParentsAndMe(d3Group *g);
アップデートフラグを親グループと自グループに立てる。
g グループのアドレス
※ファイル参照により地物に追加されたオブジェクトに関して、形状、カラー、テクスチャ等が変更され、元のファイル（既存部品）とは異なるものになった場合にはこのフラグを立て、ファイル保存時点で、その内的構成まで出力する。

void IP_TraverseError(int (*func)(int mesnum, const char *basicmes, const char *var1, const char *var2));
インタプリタのエラー検出時の割り込み関数をデフォルトとは異なるものに設定する。
func 設定するエラー処理関数(NULL の場合は、デフォルトに戻す)。
(以下は、引数 func で設定する関数に渡される引数リストである)
mesnum メッセージ番号
basicmes メッセージ固定文字列
var1, var2 メッセージ可変文字列(NULL は無しを示す)
※１つのコマンド解析でエラーが複数発生する場合がある。処理関数の戻りを TRUE にすると、できる限りコマンド解析を続け、別のエラーも検出するが、FALSE にすると現在のコマンド解析処理を中止する（エラー処理を、逐次メッセージ表示とする場合、エラーが２つ

以上あるとオペレータにとって煩わしさが増幅するため、通常 FALSE が良いと思われる)。
本関数を使わないときは、エラーメッセージ (basicmes) がコンソールに表示される。
現在サポートされているエラーとその可変部は次の通り。
I3_MSGNO_NotFoundCommandName: var1:commandline, var2:line#
I3_MSGNO_UndefCommandName: var1:command, var2:line#
I3_MSGNO_IllegalCommandLine: var1:commandline, var2:line#
I3_MSGNO_NotFoundName: var1:commandtype, var2:line#
I3_MSGNO_DupName: var1:name, var2:line#
I3_MSGNO_InsufParams: var1:name, var2:line#
I3_MSGNO_UndefName: var1:name, var2:line#
I3_MSGNO_CreateError: var1:name, var2:line#
I3_MSGNO_FileOpen: var1:name, var2:line#
I3_MSGNO_IllegalArgNum: var1:arg, var2:line#
I3_MSGNO_IllegalArgWord: var1:arg, var2:line#
I3_MSGNO_NotSupportedCommandName: var1:command, var2:line#

void i3SetAttribute(d3Group *g, const char *str);
グループに属性を追加する。
g グループのアドレス
str 属性文字列(例：“&GROUND”)

void i3SetDefaultEnv(int type);
参照先ディレクトリに係る環境を設定する。
type 環境のタイプ
I3_DEFENV_MASTER 通常編集時のディレクトリ構成
I3_DEFENV_YURYO 景観事例データベースのディレクトリ構成
I3_DEVENV_YOUSO 景観構成要素データベースのディレクトリ構成
I3_DEFENV_ZAIRYO 景観材料データベースのディレクトリ構成

void i3SummitStack(d3Group *group);
サミットグループを通知する (2.09 では廃止)
group サミットグループのアドレス

int i3_outputFile(const char *filename, d3Group *group);
グループを指定のファイルに出力する。
filename 出力ファイル名 (フルパスで指定する)

group グループのアドレス
返り値 成功時1　エラー時0

int i3_outputScenesDirect(const char *file);
シーン(LSS-S 形式)をセーブする。
file ファイル名（フルパスで指定する）
返り値 成功時1　エラー時0

int i3_outputGroup(FILE *fp, d3Group *group);
オープン済の出力ファイルにグループを追加出力する。
fp ファイルポインタ
group グループのアドレス
返り値 成功時1　エラー時0

int i3_outputGroupsDirect(const char *filename, d3Group **groups, int grpcount, int summitDel);
全てのグループをファイル(LSS-G)に保存する。
filename ファイル名
groups グループ配列のアドレス
grpcount ルートグループ数
summitDel 0:サミットグループを削除する　-1:サミットグループをしない
返り値 成功時1　エラー時0

int i3_outputScenes(i3 *ip, i3Parts *parts);
OUTPUT_SCENE コマンドを実行し、全てのシーンをセーブする。
ip 解析中のファイルに関する情報を格納した構造体
parts 引数リスト（この場合、保存ファイル名を指定）
返り値 成功時1　エラー時0

int i3_outputScene(FILE *fp, s3Scene *scn);
一つのシーンをファイル出力する。
fp ファイルポインタ
scn シーン構造体のアドレス
返り値 成功時1　エラー時0

char* i3_allocCom(i3* ip);

LSS ファイルを解析し、一つのコマンド単位を抽出する。コマンドは、「;」を区切りとして、余分な余白、タブコード等を除き、複数行に跨るものは連結して一つのコマンドとする。
ip 解析中のファイルに関する情報を格納した構造体
返り値 コマンド単位文字列を格納したメモリ・ブロックのアドレス

char* i3_allocCopyStr (const char *str,char delim);
コマンド文字列から、指定したデリミタまたは「¥0」までの部分文字列を抽出する。
str 元の文字列
delim デリミタ
返り値 抽出された文字列を格納した新たなメモリ・ブロックのアドレス

char* i3_allocGroupName(const char *name, const char *gprefix);
接頭辞を付加したグループ名称を作成する。
name グループの本文
gprefic 接頭辞
返り値 作成した名称を格納する新たなメモリブロックのアドレス

int i3_allocParts(i3 *ip,const char *com,i3Parts *parts);
コマンド文字列を、コマンドと引数に分解する。
ip 解析対象のファイルに関する情報
com コマンド文字列（入力）
parts 結果を格納する文字列のアドレス配列のアドレス
返り値 成功時：TRUE 失敗時：FALSE

int i3_camera(i3 *ip,i3Parts *parts);
カメラ情報（パース表示のためのパラメータ）を解析し、S3 構造体を構築した上で、ip に追加する。
ip 解析対象のファイルに関する情報
com コマンド文字列（入力）
返り値 成功時：TRUE 失敗時：FALSE

i3Parts* i3_copyParts(i3Parts *parts);
コマンドと引数をコピーする。
parts コマンドを構成要素に分解した文字列アドレスの配列のアドレス
返り値　複製された文字列アドレスの配列のアドレス

int i3_delete(i3 *ip, i3Parts *parts);
DELETE コマンドを実行する。
ip 解析対象のファイルに関する情報
parts コマンドを構成要素に分解した文字列アドレスの配列のアドレス
返り値　成功時：TRUE、失敗時：i3_setErrMsg 関数の返り値

int i3_effect(i3 *ip, i3Parts *parts);
EFFECT コマンドを実行する。
ip 解析対象のファイルに関する情報
parts コマンドを構成要素に分解した文字列アドレスの配列のアドレス
返り値　成功時：TRUE、失敗時：i3_setErrMsg 関数の返り値

int i3_effectGroup(i3 *ip, i3Parts *parts);
EFFECTGROUP コマンドを実行する。
ip 解析対象のファイルに関する情報
parts コマンドを構成要素に分解した文字列アドレスの配列のアドレス
返り値　成功時：TRUE、失敗時：i3_setErrMsg 関数の返り値

int i3_execCom(i3 *ip, const char *com, const char *gprefix);
コマンドを識別し、それぞれのコマンド処理に分岐・実行する。
ip 解析対象のファイルに関する情報
com コマンド文字列（入力）
gprefix 接頭辞（新たなグループを生成する場合、グループ名称を修飾する）
返り値　成功時：TRUE、失敗時：それぞれの処理の実行結果

int i3_file(i3 *ip, i3Parts *parts, const char gprefix)
FILE コマンドを実行し、外部ファイルまたは外部関数を使用する。
ip 解析対象のファイルに関する情報
com コマンド文字列（入力）
gprefix 接頭辞（新たなグループを生成する場合、グループ名称を修飾する）
返り値　成功時：TRUE、失敗時：それぞれの処理の実行結果

void i3_freeParts(i3Parts *parts);
コマンド解析の中間データを解放する。
parts コマンドを構成要素に分解した文字列アドレスの配列のアドレス

void i3_freeUserInfo(d3Group *g);
グループに定義された、属性などのユーザー情報を解放する。
g 処理対象とするグループ構造体のアドレス

i3Record* i3_getEntryRecord(i3 *ip, int typ, const char *name);
名称(面、グループ、リンク等)を参照するコマンドを実行し、名称をタイプ別に辞書にから検索する。未登録の場合には新たに登録する。
ip 解析対象のファイルに関する情報
typ データの種類
name 名称
返り値　辞書への登録内容

i3ExtCommand *i3_getExtCommand(const char *name);
外部関数登録ファイル(ext.tab)を検索し、登録内容（引数リスト）を取得する。
name 外部関数名
返り値　登録内容を記述した構造体のアドレス

i3ExtTable *i3_getExtTable();
外部関数登録ファイル(ext.tab)を開き、内容をメモリ上にロードする。
返り値　関数リスト（配列）と登録数を内容とする構造体のアドレス

int i3_getoptim()
LSS-G ファイルの保存に際して適用する最適化のレベルを取得する。
返り値　最適化レベル

int i3_group(i3 *ip, i3Parts *parts, const char *gprefix);
GOURP コマンドを実行する。
ip 解析対象の LSS-G ファイルに関する情報
parts コマンドを構成要素に分解した文字列アドレスの配列のアドレス
gprefix 接頭辞（新たなグループを生成する場合、グループ名称を修飾する）
返り値　成功時：TRUE、失敗時：i3_setErrMsg 関数の返り値

int i3_image(i3 *ip, i3Parts *parts);
IMAGE コマンドを実行する。
ip 解析対象のファイルに関する情報
parts コマンドを構成要素に分解した文字列アドレスの配列のアドレス

返り値　成功時：TRUE、失敗時：i3_setErrMsg 関数の返り値

int i3_initip(i3 *ip);
解析情報を初期化する。
ip 解析対象のファイルに関する情報
返り値　成功時：TRUE、失敗時：FALSE(ip が NULL の場合)

int i3_interpret(i3 *ip, const char *com_line, const char *gprefix);
コマンド行を解析し実行する。
ip 解析対象のファイルに関する情報
com_line コマンド行
gprefix 接頭辞（新たなグループを生成する場合、グループ名称を修飾する）
返り値　成功時：TRUE、失敗時：i3_setErrMsg 関数の返り値

void i3_jounal(i3 *ip, const char *com);
解析経過をコンソール出力する（2.09 では何もしない）
ip 解析対象のファイルに関する情報
com コマンド

int i3_lf(i3 *ip, i3Parts *parts, int typ);
FACE コマンドから面の頂点リストを構築する。
ip 解析対象のファイルに関する情報
parts コマンドを構成要素に分解した文字列アドレスの配列のアドレス
返り値　成功時：TRUE、失敗時：i3_setErrMsg 関数の返り値

int i3_lfAttr(i3 *ip, i3Parts *parts, int typ int atyp);
FACE_COLOR、FACE_NORMAL、FACE_MATERIAL、FACE_TEXTURE、LINE_COLOR、LINE_NORMAL、LINE_MATERIAL および LINE_TEXTURE の各コマンドを実行し、カラー、法線、マテリアル及びテクスチャの属性を、面または線に定義する。
ip 解析対象のファイルに関する情報
parts コマンドを構成要素に分解した文字列アドレスの配列のアドレス
typ 属性を付ける対象の種類(面または線)
atyp 属性の種類(カラー、法線、マテリアル及びテクスチャ)
返り値　成功時：TRUE、失敗時：i3_setErrMsg 関数の返り値

int i3_lfGroup(i3 *ip, i3Parts *parts, int typ, const char *gprefix, int concave);

GROUP_FACE コマンドを実行しグループに面群を定義する。
ip 解析対象のファイルに関する情報
parts コマンドを構成要素に分解した文字列アドレスの配列のアドレス
typ 名称の種類（この場合、面）
gprefix 接頭辞（新たなグループを生成する場合、グループ名称を修飾する）
concave 凹ポリゴンであることを示すフラグ
返り値　成功時：TRUE、失敗時：i3_setErrMsg 関数の返り値

int i3_light(i3 *ip, i3Parts *parts);
LIGHT コマンドを実行し、光源ユニットを構築する。
ip 解析対象のファイルに関する情報
parts コマンドを構成要素に分解した文字列アドレスの配列のアドレス
返り値　成功時：TRUE、失敗時：i3_setErrMsg 関数の返り値

int i3_lightGroup(i3 *ip, i3Parts *parts);
LIGHTGROUP コマンドを実行する（光源グループ）。
ip 解析対象のファイルに関する情報
parts コマンドを構成要素に分解した文字列アドレスの配列のアドレス
返り値　成功時：TRUE、失敗時：i3_setErrMsg 関数の返り値

int i3_link(i3 *ip, i3Parts *parts, const char *gprefix);
LINK コマンドを実行する。
ip 解析対象のファイルに関する情報
parts コマンドを構成要素に分解した文字列アドレスの配列のアドレス
gprefix 接頭辞（新たなグループを生成する場合、グループ名称を修飾する）
返り値　成功時：TRUE、失敗時：i3_setErrMsg 関数の返り値

int i3_linkxform(i3 *ip, i3Parts *parts);
LINK_XFORM コマンドを実行する。
ip 解析対象のファイルに関する情報
parts コマンドを構成要素に分解した文字列アドレスの配列のアドレス
返り値　成功時：TRUE、失敗時：i3_setErrMsg 関数の返り値

int i3_model(i3 *ip, i3Parts *parts);
MODEL コマンド(シーンへのモデルの読み込み)を実行する。
ip 解析対象のファイルに関する情報

parts コマンドを構成要素に分解した文字列アドレスの配列のアドレス
返り値　成功時：TRUE、失敗時：i3_setErrMsg 関数の返り値

int i3_mt(i3 *ip, i3Parts *parts, int typ);
MATERIAL 及び TEXTURE コマンドを実行する。
ip 解析対象のファイルに関する情報
parts コマンドを構成要素に分解した文字列アドレスの配列のアドレス
typ マテリアルかテクスチャかの区別
 I3_TYP_MATERIAL 100
 I3_TYP_TEXTURE 110
返り値　成功時：TRUE、失敗時：i3_setErrMsg 関数の返り値

int i3_mtGroup(i3 *ip, i3Parts *parts, int typ, const char *gprefix);
GROUP_MATERIAL 及び GROUP_TEXTURE コマンドを実行する。
ip 解析対象のファイルに関する情報
parts コマンドを構成要素に分解した文字列アドレスの配列のアドレス
typ マテリアルかテクスチャかの区別
 I3_TYP_MATERIAL 100
 I3_TYP_TEXTURE 110
gprefix 接頭辞（新たなグループを生成する場合、グループ名称を修飾する）
返り値　成功時：TRUE、失敗時：i3_setErrMsg 関数の返り値

i3Record* i3_newList(i3 *ip, int typ, const char *name);
 インタプリタの辞書に新たなエントリーを作成する。
ip 解析対象のファイルに関する情報
typ エントリのタイプ
name エントリの名称
返り値　成功時：TRUE、失敗時：i3_setErrMsg 関数の返り値

int i3_nextEntryRecord(i3 *ip, int typ, int startix);
インタプリタの辞書において指定した番号以降の、指定したタイプの最初のエントリを返す(ver.2.09 では非使用)。
ip 解析対象のファイルに関する情報
typ エントリのタイプ
startix 検索開始番号
返り値　該当するエントリ番号　または I3_NULL_INDEX（該当無き場合）

int i3_output(i3 *ip, i3Parts *parts, const char *gprefix);
OUTPUT コマンドを実行する。パーツの名称のファイルを作り、内容を出力する。
ip 解析対象のファイルに関する情報
parts コマンドを構成要素に分解した文字列アドレスの配列のアドレス
gprefix 接頭辞（新たなグループを生成する場合、グループ名称を修飾する）
返り値　成功時：TRUE、失敗時：i3_setErrMsg 関数の返り値

int i3_outputFace(FILE *fp, d3Face *f, int type);
一つの面をファイル出力する。条件設定により最適化出力を行う。
fp 出力先ファイル
f 出力する面
type 面(FACE)または線(LINE)の区別
 I3_EXT FACE 5
 I3_EXT_LINE 6
返り値　成功時：TRUE、失敗時：FALSE

int i3_outputGroupFace(FILE *fp, const char *gname, d3Face *f, int type);
何もしない
返り値：TRUE
※外部関数の引数がメモリ上にしか存在しない FACE 等となっていた場合に、ファイル保存に際して、この FACE を独立したファイルに出力する必要がある。このようなタイプの外部関数がまだ存在しないため、実装していない。

int i3_reset(i3 *ip);
インタープリタをリセットし、ＤＭＬメモリ空間を初期化する。
ip 解析対象のファイルに関する情報
返り値 d3Initialize の処理結果

void i3_resetIp(i3 *ip);
インタープリタの解析結果を解放する。
ip 解析対象のファイルに関する情報

int i3_scene(i3 *ip, i3Parts *parts);
SCENE コマンドを実行する。
ip 解析対象のファイルに関する情報

parts コマンドを構成要素に分解した文字列アドレスの配列のアドレス
返り値　成功時：TRUE、失敗時：i3_setErrMsg 関数の返り値

int i3_searchEntryRecord(i3 *ip, int typ, const char *name);
インタプリタの解析結果から、所定のラベル名、タイプのエントリを検索する。
ip 解析対象のファイルに関する情報
typ エントリのタイプ（グループ、面、頂点、・・・）
name エントリの名称
返り値 エントリの番号または I3_NULL_INDEX(存在しない場合：-1)

int i3_setErrMsg(i3 *ip, int mesnum, const char *var);
インタープリタのエラーメッセージを保存する。
ip 解析対象のファイルに関する情報
mesnum メッセージ番号
var 補足情報（パラメータ等）
返り値 TRUE（デフォルトの出力先の場合）または指定出力関数（IP_TraverseError 関数により出力先の変更が行われた場合）の処理結果

int i3_setFace(i3 *ip, int ix);
インタプリタが解析した面に関する情報から、d3Face 構造体を生成する。
ip 解析対象のファイルに関する情報
ix インタープリタの解析結果における処理対象面の登録番号
返り値 成功時 TRUE　失敗時 FALSE

void i3_setoptim(int i);
LSS-G ファイル保存における最適化のレベルを設定する。
int i 最適化のレベル(ビット単位のスイッチとして表現する)
 I3_OPT_3 1 三角形の連続出力
 I3_OPT_4 2 四角形の連続出力
 I3_OPT_C 4 カラーのソート
 I3_OPT_M 8 マテリアルのソート
 I3_OPT_N 16 法線のソート
 I3_OPT_T 32 テクスチャのソート

void i3_setPath(char *pathname, e3Param *envdir, const char *filename);
環境設定ファイルの指定ディレクトリと、ファイル名からフルパスのファイル名文字列を

合成する。
pathname 結果を格納する文字列
envdir 環境設定ファイルで指定されたディレクトリ名称
filename ファイル名

void i3_stackCom(i3 *ip, const char *com_line);
入力ファイルから取得した１行分の文字列をコマンド列に追加する。コメント行は無視する。
ip 解析対象のファイルに関する情報
com_line ファイルから入力した１行分のテキスト

int i3_time(i3 *ip, i3Parts *parts);
TIME コマンドを実行する。
ip 解析対象のファイルに関する情報
parts コマンドを構成要素に分解した文字列アドレスの配列のアドレス
返り値　成功時：TRUE、失敗時：i3_setErrMsg 関数の返り値

int i3_vertex(i3 *ip, i3Parts *parts);
VERTEX コマンドを実行する。
ip 解析対象のファイルに関する情報
parts コマンドを構成要素に分解した文字列アドレスの配列のアドレス
返り値　成功時：TRUE、失敗時：i3_setErrMsg 関数の返り値

int i3_vntc(i3 *ip, i3Parts *parts, int typ);
COORD, NORMAL, TCOORD, COLOR のコマンドを実行する。
ip 解析対象のファイルに関する情報
parts コマンドを構成要素に分解した文字列アドレスの配列のアドレス
typ コマンドのタイプ
 I3_TYP_V 10 頂点座標
 I3_TYP_N 20 法線ベクトル
 I3_TYP_T 30 テクスチャ座標
 I3_TYP_C 40 カラー値(R,G,B,A)
返り値 TRUE

（３）データ取得関数

char *i3GetAttribute(d3Group *g, int num);

グループに定義されている属性を取り出す。
g グループのアドレス
num 属性の番号(0,1,2,・・・)
返り値 成功時 属性を記述する文字列のアドレス　エラー時 NULL

d3Face *IP_getface(const char *lssgfile);
ジオメトリファイル(LSS-G 形式)の最初に出てくる面を取得する。
lssgfile ファイル名
返り値 成功時 面のアドレス　失敗時 NULL
※この時、目的とする面を擁するグループも生成するが、2.09 では、別プロセスである外部関数の引数の解析に使用しているため、処理終了後は全て解放され、メイン画面の表示に現れることはない。

char* i3GetGroupNamePtr(d3Group *group);
グループ名を取得する(2.09 では廃止)。
group グループのアドレス
返り値 グループ名のグループのアドレス

int i3GetDefaultEnv();
参照先ディレクトリに係る環境設定を取得する。
返り値
 I3_DEFENV_MASTER 通常編集時のディレクトリ構成
 I3_DEFENV_YURYO 景観事例データベースのディレクトリ構成
 I3_DEVENV_YOUSO 景観構成要素データベースのディレクトリ構成
 I3_DEFENV_ZAIRYO 景観材料データベースのディレクトリ構成

int i3GetGeoDefaultEnv();
現在の環境設定下における LSS-G ファイルの参照先ディレクトリを取得する。
返り値：
 E3_FILE_PATH_MASTER_GEOMETRY 通常編集
 E3_FILE_PATH_JIREI_GEOMETRY 景観事例
 E3_FILE_PATH_YOUSO_GEOMETRY 景観構成要素
 E3_FILE_PATH_ZAIRYO_GEOMETRY 景観材料

int i3GetSceneDefaultEnv
現在の環境設定下における LSS-S ファイルの参照先ディレクトリを取得する。

返り値：
E3_FILE_PATH_MASTER_SCENE 通常編集
E3_FILE_PATH_JIREI_SCENE 景観事例
E3_FILE_PATH_YOUSO_SCENE 景観構成要素
E3_FILE_PATH_ZAIRYO_SCENE 景観材料

（4）問い合わせ関数

int i3IsAttribute(d3Group *g, const char *keyword);
グループに指定の属性があるかチェックする。
g グループのアドレス
keyword 属性文字列(例：“&GROUND”)
返り値 ある時1　ない時0

int i3IsGeometry(const char *file);
ジオメトリファイルかチェックする(2.09 では廃止)。
file ファイル名
返り値 成功時1　エラー時0

int i3_checkRootGroup(d3Group *g);
グループがルートグループかどうかをチェックする。
g グループのアドレス
返り値 ルートである時1　ルートではない時0
※上方リンクの有無を調べる。

Ｂ－６．環境設定ライブラリ（ＥＮＶ）

（1）データ構築関数

void e3SetDataPath(int type);
データベースのパスをセットする(2.09 では廃止)。
type データベースタイプ

void e3ReadSettingFile(char *filename);
環境設定ファイル(デフォルト名：kdbms.set)を読み込む。
filename ファイル名

（2）初期化関数

int e3Initialize();
ENV を初期化する(2.09 では廃止)。
返り値 成功時１　エラー時０

（３）データ定義/更新関数

void e3SetKeywordDefault();
デフォルトパラメータで環境設定を行う。
※環境設定ファイル(kdbms.set)の読み込みに失敗した場合や、定義が欠如に項目に備える。

void e3SetKeyword(char *buf);
文字列（環境設定ファイルの１行に相当）を解析し、当該項目にパラメータをセットする。
buf 文字列

void e3SetPath(int index);
ディレクトリ指定項目をセットする。
index 環境設定の番号
※指定した項目が、ホーム・ディレクトリからの相対アドレスで指定されていた場合に、
これをフルパスの記述に変換する。

int e3SaveFile(char *filename);
環境ファイルを保存する。
filename ファイル名
返り値 成功時１　エラー時０
※コメント行などは、上書き保存で削除されるので注意を要する。

void e3SimpangKerja(char *tempat);
作業用ディレクトリを登録する。
tempat 作業用ディレクトリ

void e3GantiTitle(int i, char* t);
環境変数の表示タイトルを変更する。表示言語を切り替える際に使用する。
i 環境変数の ID 番号
t 新たなラベル名

（４）データ取得関数

char *e3GetDataPath(int type);

データベースのパスを取得する(2.09 では非使用)。
type データベースタイプ
返り値 パスの文字列
※初期のバージョンにおけるセットアップのディレクトリに対応していた。

int e3GetType(int index);
キーワードのタイプを取得する
index 環境設定項目の番号
返り値 タイプ

E3_TYPE_HOME_PATH	0 ホームディレクトリ（フルパス）
E3_TYPE_FILE_PATH	1 ディレクトリ（フルパスまたはホームからの相対）
E3_TYPE_FILE_NAME	2 ファイル名
E3_TYPE_SELECT	3 強調表示タイプ
E3_TYPE_LONG_1	4 一つの整数
E3_TYPE_LONG_2	5 二つの整数
E3_TYPE_LONG_3	6 三つの整数
E3_TYPE_LONG_4	7 四つの整数
E3_TYPE_FLOAT_1	8 一つの浮動小数
E3_TYPE_FLOAT_2	9 二つの浮動小数
E3_TYPE_FLOAT_3	10 三つの浮動小数
E3_TYPE_FLOAT_4	11 四つの浮動小数
E3_TYPE_STRING	12 文字列

char *e3GetTitle(int index);
環境設定項目の表示タイトルを取得する。
index 項目の通し番号
返り値 表示タイトルの文字列のアドレス
※環境設定の編集ダイアログで表示する、各言語でのタイトル(例：ホームディレクトリ)

char *e3GetName(int index);
環境設定項目のエントリ名を取得する。
index 項目の通し番号
返り値 エントリ名称の文字列
※環境設定ファイルに用いるエントリ名称(例：“E3_HOME_PATH”)

char *e3GetSelectParam(int index);

環境設定項目の選択肢を取得する。
index　環境設定項目の通し番号
返り値 選択肢の文字列

int e3GetFile(FILE *fp, char *buf, int size);
ファイルの１行分の文字列を取得する。
fp ＦＩＬＥポインタ
buf 文字列のアドレス
size 取得するサイズ（バッファ長以下）
返り値 成功時１　エラー時０

e3Param *e3GetEnvParam(int index);
指定した 環境設定項目のパラメータ部を取得する。
index　環境設定項目の通し番号
返り値 パラメータ構造体のアドレス

char *e3GetItemString(int itemno);
パラメータを文字列で取得する。
itemno　環境設定項目の通し番号
返り値 文字列

const char e3AmbilKerja();
作業用ディレクトリを取得する。
返り値 文字列（メモリ・ブロックのアドレス）

（５）データ削除関数

void e3ResetEnvParam();
環境設定ファイル(kdbms.set)データが読み込み済みであることを示すフラグを降ろす(多重読み込み防止用)。

void e3Clear();
ＥＮＶライブラリに関連したメモリ・ブロックを解放する。

void e3BebaskanKerja();
作業用ディレクトリを解除する。

（６）データ比較関数

int e3CheckDirName(char *path, char *dname);
指定したパスに指定したディレクトリがあるかチェックする。
path パス
dname ディレクトリ名
返り値 ある:1　ない:0

Ｂ－７．ユーティリティライブラリ（Ｕ３）

（１）データ構築関数

void *u3SaveGroupFamily(d3Group *g);
グループ以下全てを一時的ファイルに保存（バックアップ）する。
g グループのアドレス
返り値 成功時:u3SaveGroup 構造体のアドレス　エラー時:NULL

int u3B3Spline(int loop, int pCount, double *points, int interp, double *outPoints);
３次の B スプライン補間をする。
loop True は面、False は線
pCount 軌跡点数
points 軌跡点列(X,Y,Z･･･の並び)
interp 補間点数（１以上)
outPoints 出力座標(X,Y,Z･･･の並び)
返り値 成功時１　エラー時０

d3Group *u3RestoreGroup Family(int load, void *sginfo);
保存してある一時的ファイル（バックアップ）からグループを復元する。
load load＞0 なら復元
load≦0 なら領域の解放のみ
sginfo u3SaveGroup 構造体のアドレス（u3SaveGroupFamily の返り値)
返り値 グループのアドレス

int u3PosAlongLine(int pCount, double *points,double interval, int random, double shiftUMax, double shiftVMax, double **outPoints);
線上の配置位置を求める。
pCount 入力線の点数
points 入力線の座標列(X,Y,Z･･･の並び)

interval 間隔
random 位置の揺らぎに関する乱数の適用方法を指定
U3_RAND_NONE 0 なし
U3_RAND_FLAT 1 一定乱数 (u3RandomOffset関数の第一引数)
U3_RAND_LINEAR 2 リニア乱数 (同上)
shiftUMax 線方向の最大ずれ
shiftVMax 線に XY 面垂直方向の最大ずれ
outPoints 出力点群の座標(alloc される)
返り値 出力点数

int u3PosInXYArea(int pCount, double *points, double origin[3], double uVec[3], double vVec[3], int random, double shiftUMax, double shiftVMax, double **Points);
多角形エリア内の配置位置を求める。
pCount 入力線の点数
points 入力線の座標列(X,Y,Z･･･の並び)
origin[3] 基準点座標
uVec[3] u 軸方向と間隔
vVec[3] v 軸方向と間隔
random 位置の揺らぎに関する乱数の適用方法を指定
U3_RAND_NONE 0 なし
U3_RAND_FLAT 1 一定乱数 (u3RandomOffset関数の第一引数)
U3_RAND_LINEAR 2 リニア乱数 (同上)
shiftUMax u 方向の最大ずれ(uVec を 1)
shiftVMax v 方向の最大ずれ(vVec を 1)
outPoints 出力点群の座標(alloc される)
返り値 出力点数
※多角形のエリア内に origin を基準とし、uVec と vVec 方向に格子配置する。エリア内のチェックは XY 投影図で行う。

double u3RandomOffset(int random, double shiftMax);
ランダムな位置ずれを求める。
random ランダムタイプ
U3_RAND_NONE 0 なし (常に0.0を返す)
U3_RAND_FLAT 1 一定乱数 (0を中心に+-shiftMaxの範囲で一様)
U3_RAND_LINEAR 2 リニア乱数 4(r-0.5)|r-0.5|shiftMax, 確率密度は$|x|^{-1}$に比例)
shiftMax 最大のずれ量

返り値 乱数値（平均0.0、最小-shiftMax、最大+shiftMax）

double u3RandomRate(int random, double rateMax);
ランダムなスケールを求める。
random ランダムタイプ
rateMax 最大スケール量
返り値 平均1.0、最小1/rateMax、最大rateMax

int u3RandomNumber(int count, double *rates);
ランダムな種類を求める。
count 種類数
rates 各種類の割合
返り値 0～countまでの値
※countの場合は、どれも選ばれなかったことを示す(ratesの合計が1.0に満たない場合)。

void u3Offsetline(int pCount, double *points, int whichSide, double offsetXY, double offsetZ, double *outPoints);
オフセットした線を求める。
pCount 基準線の点数
points 基準線の座標列（X,Y,Z･･･の並び）
whichSide 右か左か（進行方向）
offsetXY　XYのオフセット
offsetZ　Zのオフセット
outPoints オフセットされた線の座標列（X,Y,Z･･･の並び）

int u3SlantPolygon(int count, double *points, double **outPoints);
スプラインによりポリゴンのスムージングを行う。
count 点数
points 入力座標（X,Y,Z･･･の並び）
outPoints 出力座標（X,Y,Z･･･の並び）
返り値 点数

int u3ClipByPolygon(int pCount, double *points, int cpCount, double *cPoints, int inside, int *newCount, int **newVcounts, double **newPolygons);
ポリゴンの切断を行う。
pCount 軌跡点の点数

points 軌跡点の座標列 (X,Y,Z･･･の並び)
cpCount 被切断面の点数
cPoints 被切断面の座標列 (X,Y,Z･･･の並び)
inside TRUE:内側の切断 FALSE:外側の切断
newCount 切断後のポリゴン数
newVcounts 切断後のポリゴン頂点数
newPolygons 切断後のポリゴン座標列 (X,Y,Z･･･の並び)
返り値 TRUE:正常　FALSE:切断なし

void u3CalcTcoord(int opCount, double *oPoints, double *oTcoords,int pCount, double *points, double *tcoords);
テクスチャ座標を計算する。
opCount 元の面の頂点数
oPoints 元の面の頂点列 (X,Y,Z･･･の並び)
oTcoords 元の面のテクスチャ座標列
pCount 新しい面の頂点数
points 新しい頂点列 (X,Y,Z･･･の並び)
tcoords 新しい面のテクスチャ座標

int u3CrossCheck2d(double *sv, double *ev, double *csv, double *cev, double *opnt ,int *scross, int *ccross);
交わりの状態をチェックする。
sv 線分１の始点
ev 線分１の終点
csv 線分２の始点
cev 線分２の終点
opnt 交点
scross 線分１の交わり状態
ccross 線分２の交わり状態
返り値 交わりあり:1　交わりなし:0

int u3MakeUVMatrix(double uvOrg[3], double u[3], double v[3], d3Matrix m);
ＵＶマトリクスを作成する。
uvOrg[3] ｕｖ原点
u[3] ｕベクトル
v[3] ｖベクトル

m マトリクス
返り値 成功時１　エラー時０

int u3CrossPoint2d(double p1[2], double p2[2], double p3[2], double p4[2], double op[2], double *s, double *t);
線分と線分の交点を求める（２次元)。
p1[2] 線分１の始点
p2[2] 線分１の終点
p3[2] 線分２の始点
p4[2] 線分２の終点
op[2] 交点
s 線分１の長さに対する始点から交点までの距離の割合
t 線分２の長さに対する始点から交点までの距離の割合
返り値 交点あり:1　交点なし:0

int u3CrossPoint(double l1s[3], double l1e[3], double l2s[3], double l2e[3], double point1[3], double point2[3], double *rate1, double *rate2);
線分と線分の交点を求める（３次元)。
l1s[3] 線分１の始点
l13[3] 線分１の終点
l2s[3] 線分２の始点
l2e[3] 線分２の終点
point1[3] 線分１の交点
point2[3] 線分２の交点
rate1 線分１の長さに対する始点から交点までの距離の割合
rate2 線分２の長さに対する始点から交点までの距離の割合
返り値 交点あり:1　交点なし:0

int u3DecomposePolygonWithWin2d(u3Polygon2d *polygon, int win, u3Polygon2d *winPolygon[], int *tri, u3Poygon2d **triPolygons);
ポリゴンの分割
polygon 被分割ポリゴン
win 分割モード（内/外)
winPolygon[] 窓ポリゴン
tri 分割後のポリゴン数
triPolygons 分割後のポリゴン配列のアドレス

返り値 成功時１　エラー時０

int u3CutPolygon2d(u3Polygon2d *tobeCut, u3Polygon2d *cutPolygon, int win, int *tri,
u3Polygon2d **triPolygons);
ポリゴンの切断を行う。
tobeCut 被切断ポリゴン
cutPolygon 切断ポリゴン
win 切断モード（内/外）
tri 切断後のポリゴン
triPolygons 切断後のポリゴン配列のアドレス
返り値 成功時１　エラー時０

u3Polygon3d *u3PolygonAlloc();
u3Polygon3 構造体の領域を確保する。

int u3DivideTriangle3d(u3Polygon3d *polygon, double area, int *tri,
  u3Polygon3d **triPolygons);
ポリゴンを三角形分割する（2.09 では land.dll に移管)。
polygon 分割するエリア
area エリア
tri 分割後のポリゴン数
triPolygons 分割後のポリゴンのアドレスのアドレス
返り値 成功時１　エラー時０

int u3CombineTriangle3d(int count, u3Polygon3d **polygons, float d_value,
  float r_ratio, int *tri, u3Polygon3d **triPolygons);
ポリゴンを結合する（2.09 では land.dll に移管)。
count ポリゴン数
polygons 結合するポリゴン配列のアドレス
d_value 結合するレートを表す係数
r_ratio 結合共用範囲角度
tri 結合後のポリゴン数
triPolygons 結合後のポリゴン配列のアドレス
返り値 成功時１　エラー時０

int u3Round(int innum, u3Poygon3d **inpoly, double radius, int division_no,

int ednum, u3Edge3d *smoothing_edge, int *outnum, u3Polygon3d **outpoly);
丸め処理をする (2.09 には含めない)。
innum 頂点数
inpoly ポリゴン配列のアドレス
radius 半径
division_no 分割数
ednum 頂点数 (線データ)
smoothing_edge エッジのアドレス
outnum 点数
outpoly 出力ポリゴン配列のアドレス
返り値 成功時 1　エラー時 0

（2）データ定義/更新関数

int u3CheckPointInAreaXY(double point[3], int pCount, double *points);
点のエリア内をチェックする
point[3] 点
pCount エリア線の点数
points エリア線の座標列(X,Y,Z･･･の並び)
返り値 エリア内:1　エリア外 0

void u3StartRandom(int restart);
ランダム計算を開始する。
restart 1:前回の乱数初期値と同　0:新規乱数 (呼び出し時刻を用いて SEED を更新)

int u3OptimizeQuadXY(double points[12], double texCoord[8]);
四角形の矛盾を除く。
points[12] 頂点
texCoord[8] テクスチャ座標
返り値 正常:4
ＸＹから見て右回り:-4
ＸＹから見て８の字なら左回りの部分を残した三角形にする:3
同一点、同線上の点がある場合は削除する:3～0

int u3TransGrid(int width, int height, double left, double rigth, double bottom,
  double top, double zNear, double zFar, int count, int *ipoint2d, float *depth,
  double *points);

二次元ポリゴンとデプスバッファを用いた地面の高さから三次元ポリゴンを生成する(nori.dll に移管)。
width バッファの横サイズ
height バッファの縦サイズ
left, rigth X 方向の min、max
bottom, top Y 方向の min、max
zNear, zFar Z 方向の min、max
count 点数
ipoint2d ２次元のポリゴンの座標列 (X,Y,Z･･･の並び)
depth Z 値列 (3×width×height)
points 出力座標
返り値 点数

int u3CrossCheck(int cnt1, ouble *pnt1, int ix1, int cnt2, double *pnt2, int ix2,
    double *opnt);
被切断面ポリゴンと切断面ポリゴンの交点チェックを行う。
cnt1 被切断ポリゴンの点数
pnt1 被切断ポリゴンの座標列 (X,Y,Z･･･の並び)
ix1 被切断ポリゴンの線分の通し番号
cnt2 切断ポリゴンの点数
pnt2 切断ポリゴンの座標列 (X,Y,Z･･･の並び)
ix2 切断ポリゴンの線分の通し番号
opnt 交点があった場合の座標列 (X,Y,Z)
返り値 TRUE:交点あり
FALSE:交点なし

int u3FaceCrossCheck(int out, int cnt1, double *pnt1, int ix1, int cnt2, double *pnt2,
  double **opnt);
被切断ポリゴンと切断ポリゴンの包含チェックを行う。
out TRUE:交点座標列を出力する
FALSE:出力しない
cnt1 被切断歩 ori ゴンの点数
pnt1 被切断ポリゴンの座標列 (X,Y,Z･･･の並び)
ix1 被切断ポリゴンの線分の通し番号
cnt2 切断ポリゴンの点数
pnt2 切断ポリゴンの座標列 (X,Y,Z･･･の並び)

opnt 交点があった場合の座標列 (X,Y,Z)
返り値 交点数

int u3CheckSamePoint(int count, double *inPoints, double ** outPoints);
同一点をチェックし、不要な頂点を削減する。
count 点数
inPoints 入力座標列 (X,Y,Z･･･の並び)
outPoints 出力座標列 (X,Y,Z･･･の並び)
返り値 出力点数

（3）データ取得関数

void u3GetQuadShape(int ix, double *leftPoints, double *rightPoints,
  double points[12]);
２本の点列のそれぞれの ix 番目と ix+1 番目をつないだ四角形を取り出す
ix 点の通し番号
leftPoints 左側の点列 (X,Y,Z･･･の並び)
rightPoints 右側の点列 (X,Y,Z･･･の並び)
points[12] 四角形の座標

void u3GetQuadTcoord(int ix, int uvType, double *centerPoints, double offsetXY,
  double uStart, double vStarts, double uScale, double vScale, double texCoord[8]);
２本の点列の ix 番目と ix+1 番目をつないだ四角形のテクスチャ座標を取り出す
ix 点の通し番号
uvType 縦／横
centerPoints 線 (X,Y,Z･･･の並び)
offsetXY オフセット値
uStart 開始点の u 座標
vStarts 開始点の v 座標
uScale u スケール
vScale v スケール
texCoord[8] テクスチャ座標

int u3Get2dGridPolygon(int width, int height, unsigned char *grids, int xStart,
  int yStart, int *ipoints2d);
ステンシルバッファから島状図形の輪郭線を抽出する。
width バッファの横サイズ

height バッファの格子の縦サイズ
grids バッファのメッシュの値(0/1)(width×height)
xStart, yStart 着目する島の内部点の位置を示す
ipoints2d ポリゴン（格子）座標（2×width×height）の格納先
返り値 ポリゴンの頂点数

int u3GetZLine(d3Face *f, double z, double s[3], double e[3]);
面の高さ Z の水平面との交線を調べる。
f 面のアドレス
z 高さ
s[3] 交線の始点
e[3] 交線の終点
返り値 交差がある 1　無い 0

Ｂ－８．メッセージライブラリ（Ｚ３ＥＲＲ）

（1）メッセージ定義ファイル

E3_BIN_PATH　で設定されているディレクトリ上にある“ERR_MSG.txt”というメッセージ定義ファイルにあるメッセージフォーマットを使用する。
その“ERR_MSG.txt”ファイルのメッセージのフォーマットを以下に示す。

＜ERR_MSG.txt＞

```
E  1001  エラー(%d)
E  1002  アプリケーションエラー_%s
  ・・・・・・・・・・・・
W  3001  %s がありません
W  3002  %d の%s が見つかりません
  ・・・・・・・・・・・・
I    5001  %s ができません
  ・・・・・・・・・・・・
C  7001  %s の削除をしてもいいですか？
```

１カラム目：メッセージタイプを表すアルファベット大文字１文字：
　　Ｅ：エラー(error) Ｗ：警告(warning) Ｉ：情報(info) Ｃ：確認(confirm)
　　これら以外は無視される。
　　基本的な仕分け：

E はシステムに起因するエラー：ユーザーに対する謝罪（バグ、リソース不足など）
W はユーザーに起因するエラー：ユーザーに対する訓示（禁じ手、操作ミスなど）
I はユーザーへの安心感の増進（操作成功の祝辞など、うるさくない範囲で）
C はユーザーに判断を求める場合（疑問手や重大な操作の事前再確認、後始末選択）

２カラム目：スペースを入れる。

３カラム目から４桁の数字：メッセージNo.

1000 番台：エラー　　3000 番台：警告

5000 番台：情報　　7000 番台：確認

※これは整理のためのものであって、表示方法はメッセージタイプが決める。

例えば、5000 番台のメッセージでも、タイプを W とすれば警告

７カラム目：スペース

８カラム目：メッセージ

%d、%f、%s に対応し、%.2f などのフォーマットも指定可能で個数の制限は特にない。

また、スペースを入れたい時は半角アンダーバー（_）で置き換えて入力すること。

制限事項：１行 512byte を越えてはならない。

（２）関数

void z3LoadMessage();

初版において、メッセージ定義ファイルをロードする。

一番最初に呼ぶ。

void z3LoadMessageML(char* filename);

多言語版において、メッセージ定義ファイルをロードする。多言語版において、起動時または言語切換時に呼ぶ。

filename 言語別のエラーメッセージファイル名をフルパスで指定

（例：″c:¥@keikan¥ksim¥Language¥ja¥ERR_MSG.ja.txt″）

void z3CloseMessage();

メモリー上のメッセージテーブルを解放する。プログラム終了時に呼ぶ。

void z3StoreMessage(int no, ...);

メッセージをバッファにセットする。

no メッセージ番号

... メッセージ可変引数部

※表示系（G3DRL）のエラーは、直ちに表示すると多重エラー無限ループに陥りがちである。

int z3FlushMessage(int display);
バッファにセットされているメッセージを表示(非表示)して、メッセージをクリアする 。
display TRUE ならセットしてあるメッセージを表示
FALSE なら非表示
返り値 TRUE ならＯＫ（はい）ボタンが押された
FALSE ならいいえボタンが押された
（メッセージタイプが確認モードのときのみ意味がある）

void z3Message(int no, ...);
no（エラー、警告、情報モード）のメッセージを（直ちに）表示する。
no メッセージ番号
... メッセージ可変引数部

int z3Confirmation(int no, ...);
no（確認モード）のメッセージを（直ちに）表示する。
no メッセージ番号
... メッセージ可変引数部
返り値 TRUE ならＯＫ（はい）ボタンが押された　FALSE ならいいえボタンが押された

void z3GetMessage(char *type, int *no, char *msg);
現在ヒットされているメッセージ情報を取得する。
type,msg 共に呼ぶ側で領域（type[2]、msg[512])を確保しておくこと
type E（エラー）かW（警告）かI（情報）かC（確認）
no メッセージ番号
msg （可変部が埋められている）メッセージ

int z3ShowWindow();
メッセージウインドウを表示する。
中身は、UNIX と WindowsNT とで異なる
ライブラリ以外から呼ばないこと

Ｂの２．アプリケーション・ライブラリ関数

（１）システム管理機能

①初期化　　（従前の（１）初期化関数）

システムの起動時、データの新規作成、あるいは新たなファイルを読み込む前に、システムの全体または一部の初期化を行う。メモリ管理の上では、②の終了処理と対になる。

void InitStruct();
内部データ(ms 構造体)を初期化する。ms.scene, ms.geometry 等は NULL とする。

void InitSceneStruct();
内部シーンデータ(ms.scene)にメモリ・ブロックを割り当て、内容を初期化する

void InitGeometryStruct();
内部ジオメトリデータ(ms.geometry) にメモリ・ブロックを割り当て、内容を初期化する

void InitImageStruct();
内部イメージデータ(ms.image) にメモリ・ブロックを割り当て、内容を初期化する

void InitCameraStruct();
内部パースデータ(ms.cam) にメモリ・ブロックを割り当て、内容をを初期化する

void InitOrthoStruct();
内部オーソデータ(ms.ortho) にメモリ・ブロックを割り当て、内容をを初期化する

void InitGeometryLightStruct();
内部ジオメトリ用光源グループ(ms.geometry->lg)にデフォルト光源を設定する。

void InitGeometryEffectStruct();
内部ジオメトリ用効果データを空に初期化する（何もしない)。

void InitTextureParam();
テクスチャマッピングのためのパラメータにデフォルト値を設定する 。

void InitTextureMapping(char *texname);

適用対象が選択されている場合、初期値を用いてテクスチャ・マッピングを実行する。
選択されていない場合、テクスチャのロードだけを行う。
texname テクスチャファイル名

void InitPrimitiveParam(char *name);
原始図形パラメータを初期化する。
name 原始図形名称

void InitSteelParam(char *name);
形綱パラメータを初期化する。
name 形鋼名称

void InitBridgeParam(char *name);
橋パラメータを初期化する。
name 橋名称

②終了処理（初版の（７）データ削除関数）
void FreeStruct();
内部データを解放する。

void FreeFileName(char *buf);
メモリブロックを解放する。
buf 文字列を格納したメモリブロック

int DeleteSummitGroup(d3Group ***root, d3Group **summit);
サミットグループを削除し、サミット直下のグループをルートグループとする。
root ルートグループ群を指すポインタ配列へのポインタ
summit サミットグループを指すポインタのアドレス
返り値
ルートグループ数（成功時）
-1（単位行列でないリンクをもつルートグループがあり、削除できなかった場合）

void DeleteScene(int width, int height);
シーンを削除し、指定した幅と高さの画面にする。
width 表示エリアの幅
height 表示エリアの高さ

void ClearDataLssS(int num, s3Scene **scns);
シーンをクリアする（Ver.2.09 では使用していない）
num シーン数
scns シーンポインタへのポインタ

void ClearDataLssG(int num, d3Group **root);
ジオメトリをクリアする（Ver.2.09 では使用していない）
num ルートグループ数
root ルートグループポインタへのポインタ

int DeleteFrontAndBackImage(s3Scene *scn, int type);
前景イメージまたは背景イメージのいずれかを削除する
scn 操作対象となるシーンへのポインタ
type　前景と背景の区別
　K_IMAGE_BACK 背景
　K_IMAGE_FRONT 前景
返り値 成功時:TRUE　エラー時:FALSE

（２）システム状態のモニタリングと、問い合わせへの回答

アプリケーション・ライブラリは、システム全体で唯一のスタティック変数に様々のシステム状態を記録しておき、異なる編集ダイアログの間でデータやシステム状態を共有することを可能としている。また、①に列挙された関数（SetXXX）を用いたこのようなスタティック変数への書き込みと、②に列挙された関数(GetXXX)を通じてのそこからのデータの読み出しは対になっている。

システムの開発、デバッグに際しては、これらの関数にブレークポイントを設けることにより、システム状態の変遷を確実に監視することができる。

以下の①と②は、この他に、データのバックアップと復原に用いる関数も含んでいる。

①システム状態の記録　（初版の（５）データ定義関数）

void SetLssFileType(int type);
編集対象とするLSSファイルタイプをセットする
type LSSファイルタイプ
　K_LSS_S：LSS-Sファイル
　K_LSS_G：LSS-Gファイル

void SetFileSelectMode(int mode);

ファイル入出力に際してファイル選択モードをセットする
mode ファイル選択モード
  K_LSS_S シーン読み込み
  K_LSS_G ジオメトリ読み込み
  K_LSS_G_CHILD ジオメトリの追加読み込み
  K_READ_MODEL シーンモデルの読み込み
  K_IMAGE イメージモデルの読み込み
  K_SAVE_SGI_IMAGE SGI ファイルの保存
  K_SAVE_JPEG_IMAGE JPEG ファイルの保存
  K_SAVE_AS ファイルの保存
  K_SAVE_MODEL モデルの保存

void SetViewType(int type);
表示画面（ビュー）のタイプをセットする。
type ビュータイプ（g3drl.h で定義）
  G3_CAM_FRONT 南立面図（正面図）
  G3_CAM_TOP 平面図
  G3_CAM_SIDE 東立面図（側面図）
  G3_CAM_UTARA 北立面図
  G3_CAM_BARAT 西立面図
  G3_CAM_PERS 透視図（パース）

void SetSceneFileName(char *name);
シーンファイル名をセットする
name ファイル名

void SetSceneCurrentIndex(int index);
表示および編集の対象とするシーンの番号をセットする
index シーン番号（０～シーン総数－１）

void SetScene(int num, s3Scene **scn_array);
シーンをセットする　(Ver.2.09 では廃止)
num シーン数
scn_array シーンポインタへのポインタ

void SetGeometryFileName(char *name);

LSS-G ファイルの読み込み、保存を行った際にファイル名を記録する。
name ファイル名

void SetGeometryGroup(int num, d3Group **rgrp_array);
ms.geometry 構造体にルートグループの配列をセットする
num ルートグループ数
rgrp_array ルートグループ配列（メモリ・ブロック）のアドレス

void SetPers(s3Camera *cam);
ms.cam 構造体に透視図のパラメータ情報をセットする
cam カメラ構造体（メモリ・ブロック）へのポインタ

void SetOrtho(int type, g3Ortho *ortho);
ms.ortho[type-1]構造体に平行投影表示に関するパラメータをセットする
type 図面タイプ（平面図、南立面図、・・・）
  G3_CAM_FRONT 南立面図（正面図）
  G3_CAM_TOP 平面図
  G3_CAM_SIDE 東立面図（側面図）
  G3_CAM_UTARA 北立面図
  G3_CAM_BARAT 西立面図
ortho パラメータを格納したメモリ・ブロックのアドレス

void SetGeometryLightGroup(s3LightGroup *lg);
ms.geometry 構造体に、LSS－G 編集モードにおける光源グループをセットする
lg 光源グループ（メモリ・ブロック）のアドレス

void SetGeometryEffectGroup(s3EffectGroup *eg);
ms.geometry 構造体に、LSS-G 編集モードにおける効果グループをセットする
eg 効果グループ（メモリ・ブロック）のアドレス

void SetGeometryTime(float date);
ms.geometry 構造体に、時間を設定する。
date 日数

void SetImage(s3Image *img);
ms.image 構造体に、画像情報を格納したメモリ・ブロックのアドレスをセットする。

img イメージポインタ

int ChangeTime(float date);
LSS-G、LSS-S 共通に、現在編集中のシーンに関して現在時間を変更すると共に、時間依存する各種条件を更新する。
date 日数
返り値 成功時:TRUE　エラー時:FALSE

void SetPointersInSceneToLss();
表示・編集の対象とするシーン情報を格納したメモリ・ブロックのアドレスを表内部データに反映させる。これには、モデル、前景・背景、光源グループ、効果グループ、時間が含まれる。Ver.2.09 においては、内部処理ロジックの変更に伴い、比較のみ行い、相違があればメッセージを表示する。

void SetSceneToLss();
現在表示されている最新のシーン情報を、内部データに反映させる。

void SetSceneInitWinSize(int width, int height);
メイン画面の表示サイズをセットする。その際に、幅の最小値は保つようにし、要求された幅(width)がこれよりも小さかった場合には最小値とし、縦横比を保つように高さも調整する。背景・前景が指定されている場合には、この画像の縦横比を保つように高さを調整する。背景と前景が異なっていた場合には背景を優先する。
width 表示エリアの幅
height 表示エリアの高さ

void SetInputMainDrawareaMode(int mode);
メイン表示エリアのモードをセットする。画面上でマウス左ボタンがクリックされた場合の動作を、状況に応じて切り換えるために使用する。
mode モード (TRUE/FALSE)

int SelectingGroup(int x, int y, int width, int height);
グループの選択を行う。指定された x,y 座標に係るグループを、画面縦横範囲の中から探索する。結果は、g3SelPath 構造体に格納され、別途 g3GetSelpath(int,g3SelPath*)関数により取得できる。選択されたグループを強調表示する。
x, y ウインドウの座標
width 表示エリアの幅

height 表示エリアの高さ
返り値 成功時:TRUE　エラー時:FALSE

void SetSelectCount(int count);
同一場所に重複して存在しているグループの数をスタティック変数に記憶する 。
count 選択数

void SetSelCurrentIndex(int index);
選択対象の通し番号をスタティック変数に記憶する。
index 戸押し番号

void SelectNext();
同じポイントに複数のグループが重なって存在する場合、選択されたグループの番号を次に移す。最後（最奥）のグループの次は最初（最前）のグループとする。

void SetSelectType(int seltype);
画面選択における選択タイプを設定する。
seltype 選択タイプ
  G3_SEL_GROUP グループ
  G3_SEL_FACE 面
  G3_SEL_SAMECOLFACE_IN_GROUP グループの同色面

void SetLayoutMode(int mode);
移動・回転・スケールの編集状態の設定（Ver.2.09 では意味を失っている)。
mode レイアウトモード
  K_LAYOUT（移動・回転・スケール)

void SetSimMode(int mode);
景観シミュレータの動作モードを記憶する。
mode モード
 K_VIEWER　　メイン画面（ビューワ）
 K_LAYOUT　　移動・回転・スケール
 K_BRIDGE　　橋
 K_OPTION　　ユーザー定義のパラメトリック部品

void SetCheckWinMode(int mode);

チェックボックスの状態等を記憶する(Ver.2.09 では使用していない)。

void SetMaterialSelectionMode(int flag);
数種類あるマテリアル編集画面マテリアルのどれが現在開いているか、状態を記録する。
ビット毎に画面を区別する。
　オリジナルのマテリアル編集画面：１
　オリジナルのテクスチャ編集画面：２
　オリジナルのマテリアルファイル画面：４
　オリジナルのマテリアル選択画面：８
　グラフィックなマテリアル編集:１６
　グラフィックなテクスチャ編集：３２
　テクスチャの貼り方：６４
　カラーセット編集：１２８
　各編集画面が開いた時点で値をセットする。閉じた時点で負値（例えばグラフィックなテクスチャ編集ならば３２）を引数とすることにより、そのダイアログに対応したビットがクリアされる。

void SetMultiLayoutCopyMode(int flag);
配置/コピーの編集であることを示すフラグをセットする。
flag ON:TRUE　OFF:FALSE

void SetCalledDBFlag(int flag);
データベースからの起動であることを示すフラグをセットする。
flag TRUE/FALSE

void SetNameStr(char *name);
ファイル名（固定文字列のアドレス）をスタティック変数（配列）に保存する。
name ファイル名
 K_CALL_COMMU_FILE：tmp001.txt（データベースから選択された項目）
 K_CALLED_COMMU_FILE：tmp002.txt（確認表示するファイル）
 K_SAVED_FILE：tmp003.txt（編集中のデータのファイル名）

void SetNewCameraName(s3Camera *cam);
古い視点名称があればこれを解放した上で、新しい視点名称（メモリ・ブロック）をセットする。
cam 構造体（メモリ・ブロック）

void SetBackupCameraParam(s3Camera *cam);
視点情報のバックアップをする。スタティック変数にパラメータが代入されるが、名称に関しては、メモリ・ブロックのコピーが作成される。
cam バックアップする視点情報（メモリ・ブロック）のアドレス

void SetBackupGridParam(int mode, double interval);
グリッド情報のバックアップをする。スタティック変数に値を代入する。
mode モード
interval 間隔

void SetInitLayoutParam();
移動/回転/スケールのパラメータに初期値をセットする 。平行移動、回転、スケール、移動回転の中心の４項目について、デフォルト値をセットする。

void SetLayoutParamTranslate(double *tran);
移動/回転/スケールのパラメータの内、平行移動量をセットする。デフォルト値は、(0,0,0)である。
tran 移動量 (X,Y,Z)

void SetLayoutParamRotate(double *rot);
移動/回転/スケールのパラメータの内、回転をセットする。デフォルト値は(0,0,0)である。
rot 回転 (X,Y,Z)

void SetLayoutParamScale(double *scl);
移動/回転/スケールのパラメータの内、スケールをセットする。デフォルト値は(1,1,1)である。
scl スケール (X,Y,Z)

void SetLayoutParamCenter(double *center);
移動/回転/スケールのパラメータの内、回転の中心をセットする。デフォルト値は(0,0,0)である。
center 回転中心 (X,Y,Z)

void BackupInitLayoutMatrix(g3SelPath *spath);
選択されたオブジェクトのリンクマトリクスをバックアップする。

spath セレクトパス

void SetInitLayoutGroupLink(g3SelPath *spath);
グループのリンクにレイアウトパラメータの初期値をセットする
spath セレクトパス

int SetLayoutGroupLink(g3SelPath *spath, double *tran, double *rot, double *scl, double *center);
配置操作に際して指定された配置位置と、移動、回転、スケールおよび移動回転の中心の各座標から、リンクマトリクスを計算し、配置するグループの上方リンクのマトリクスにセットする。
spath セレクトパス
tran 移動量 (X,Y,Z) へのポインタ
rot 回転 (X,Y,Z) へのポインタ
scl スケール (X,Y,Z) へのポインタ
center 回転中心 (X,Y,Z) へのポインタ
返り値 成功時:TRUE　エラー時:FALSE

void SetExitFlag(int flag);
終了フラグをセットする
flag ０～２

void BackupLightGroupLight(s3Light *l);
光源ユニットの各数値をスタティック変数に代入する。光源名称に関してはメモリ・ブロックのコピーを作成する。
l 光源ユニットポインタ

void BackupLightGroup(s3LightGroup *lg);
光源グループのバックアップをする。引数の光源グループを構成する全てのメモリ・ブロックに関して、コピーを作成する。引数が NULL である場合、バックアップを構成する全てのメモリブロックを解放して再初期化する。
lg 光源グループポインタ

void BackupSceneLightGroupAll(int scnnum, s3Scene **scns);
シーン数と同じ長さの配列を作り、各シーンの光源グループのバックアップを作る。引数の scns が NULL であった場合、バックアップを構成する全てのメモリ・ブロックを解放す

る。
scnnum シーン数
scns シーンポインタへのポインタ

void SetSelectLightIndex(int index);
選択した光源ユニットの戸押し番号を記憶する。
index 通し番号

void SetNewLightFlag(int flag);
新しい光源ユニットを作成するフラグをセットする。編集操作がキャンセル終了した場合にバックアップデータ処理の場合分けに用いる。
flag TRUE/FALSE

int AddLightGroupLight(char *name, s3LightGroup *lg);
光源グループ lg に、名称 name の光源ユニットを追加する。lg に既に同名の光源ユニットがあれば何もせず、FALSE を返す。なければ、全てのシーンから name と同名の光源ユニットを探し、これを lg に加え TRUE を返す。無ければ FALSE を返す。
name 光源ユニット名称
lg 光源ユニットポインタ
返り値 成功時:TRUE　エラー時:FALSE

int ChangeLightGroup(char *name, s3LightGroup **lg);
光源グループを変更する。シーンに定義されている光源グループの名称を初めから順に調べ、最初に一致した名称の光源グループのアドレスを第二引数のポインタに代入し、TRUE を返す。名称が一致する光源グループが無ければ、FALSE を返す。
name 光源グループ名称
lg 光源グループを指すポインタのアドレス
返り値 成功時:TRUE　エラー時:FALSE

void SetInitBackupMaterial();
バックアップ用のマテリアル情報（カラー値及び反射率を記録するスタティック変数）に初期値をセットする。

void BackupMaterial(float *rgba, float *specular);
マテリアルのバックアップをする
rgba カラー値(R,G,B,A)

specular 反射率(R,G,B,A)

int SetFaceMaterialToRGBAandSpecular(float *rgba, float *specular, d3Face *f);
面に定義されたマテリアルの RGBA と反射率を取得する
rgba カラー値(R,G,B,A)
specular 反射率(R,G,B,A)
f 参照する面（メモリ・ブロック）のアドレス
返り値 成功時:TRUE　エラー時:FALSE（マテリアルが定義されていない場合）

void SetInitBackupMaterialLab();
バックアップ用 Lab 情報(スタティック変数)に初期値をセットする

void BackupMaterialLab(float *lab);
Lab のバックアップをする
lab Lab 値

void BackupSelectingObject(int type, g3SelPath *spath);
選択したオブジェクトのバックアップをする
type セレクトタイプ（グループ、面、同色面）
spath セレクトパス

void BackupGroup(d3Group *g);
グループのバックアップをする。グループのメモリ・ブロック、及びこれを構成する各面のメモリ・ブロックを複製する。
g グループ（メモリ・ブロック）のアドレス

void BackupGroupFree();
バックアップしたグループのメモリ・ブロックを解放する

void BackupFace(d3Face *f);
面のバックアップをする。ユニークな面の配列として情報を管理し、既存の配列に無い新たな面が指定された場合には、末尾に追加する。
f 面ポインタ

void BackupFaceFree();
バックアップした面を解放する

void SetTextureParam(double *org, double *horizontal, double *vertical, double *soutai, double *scale);
テクスチャパラメータをスタティック変数に記録する。
org テクスチャ原点 (X,Y,Z)
horizontal u ベクトル (X,Y,Z)
vertical v ベクトル (X,Y,Z)
soutai 相対移動量 (X,Y)
scale スケール (X,Y)

void MappingTexture();
選択されているオブジェクト（グループまたは面）に対して、テクスチャ座標を設定する。SetTextureParam 関数により設定されているパラメータを用いる。適用するテクスチャの指定はこの関数の中では行わない。

void UnmappingObjectTexture(int type, g3SelPath *spath);
選択されているオブジェクトのテクスチャを解除する。適用するテクスチャを解除するのみで、設定済みのテクスチャ座標は温存される。
type 選択タイプ（面またはグループ）
spath セレクトパス（処理対象を特定する）

void SetTextureFileName(char *file)
テクスチャファイル名を記録する。
file ファイル名

void SetAutoTexList(g3TsTop *ts);
自動貼り付けテクスチャのリストをスタティック変数に記憶する。
ts g3TsTop 構造体へのポインタ

void AutoTextureMappingGroup(g3SelPath *spath, g3TsTop *ts, int level1, int level2, double uscale, double vscale);
グループにテクスチャを自動貼り付けする。
spath セレクトパス
ts g3TsTop へのポインタ
level1 リストのレベル 1 のインデックス
level2 リストのレベル 2 のインデックス

uscale u スケール
vscale v スケール

void AutoTextureMappingFace(g3SelPath *spath, g3TsTop *ts, int level1, int level2, double uscale, double vscale);
面にテクスチャを自動貼り付けする。
spath セレクトパス
ts g3TsTop へのポインタ
level1 リストのレベル１のインデックス
level2 リストのレベル２のインデックス
uscale u スケール
vscale v スケール

void BackupTime(float date);
時間をスタティック変数に記録する。
date 日数（浮動小数）で記述した経年

void BackupAntialias(int flag, int count);
アンチエリアシングの設定をスタティック変数に記録する。
flag ON:TRUE、OFF:FALSE
count 精度

void SetSummitGroup(int *num, d3Group ***root, d3Group *summit);
サミットグループをルートグループ配列の最初の要素にセットする。もし num が２以上であれば、ルートグループを解放し、新たに１の要素だけを持つルートグループ配列を割り当てた上で以上の処理を行う。
num ルートグループ数
root ルートグループ配列を指すポインタのアドレス
summit サミットグループポインタ

void SetPrimitiveParam(double *toppoint, double *bottompoint, double topradius,
  double bottomradius, int n, char *name);
原始パラメータをセットする(Ver.2.09 では廃止)
toppoint 上部原点 (X,Y,Z)
bottmpoint 下部原点 (X,Y,Z)
topradius 上部半径

bottomradius 下部半径
n 角数
name グループ名称
タイプによりセットしないパラメータがある

void SetSteelParam(double *org, double *param, double *sft, char *name);
形鋼パラメータをセットする(Ver.2.09 では廃止)
org 物体原点 (X,Y,Z)
param パラメータ (a,b,c,d,h)
sft シフト (X,Y)
G3_LSTEEL のみ有効
name グループ名称

void SetBridgeParam(double *org, double *size, char *name);
橋パラメータをセットする(Ver.2.09 では廃止)
org 橋原点
size 幅、長さ、高さ
name グループ名称

void SetModifyFlag(int flag);
変更フラグをセットする。セットされている場合、終了時に保存するかどうかを尋ねる。
flag TRUE/FALSE

void SetNewType(int type);
新規作成のタイプ
type タイプ
  K_LSS_S シーン
  K_LSS_G ジオメトリ

void SetNewDataFlag(int flag);
新規データフラグをセットする
flag TRUE/FALSE

void SetOpenFlag(int flag);
ファイルを開く操作を記録する。
flag TRUE/FALSE

void SetPickCoordMode(int mode);
座標ピッキングモードをセットする。メイン画面が平行投影（オルソ系）である場合に、画面クリックした点の座標を、他の編集画面に伝達する処理を行うかどうかを決める。
mode TRUE/FALSE

void SetLayoutPickCoordMode(int mode);
移動・回転・スケール編集画面において、メイン画面から届くクリック位置に対応した座標値を、平行移動、回転、回転中心どのパラメータに用いるかを指定する
mode：座標値の適用先（ K_LAYOUT_MOVE, K_LAYOUT_ROTATE, K_LAYOUT_ROT_CENTER)

void SetGenshiPickCoordMode(int mode);
原始図形生成において座標ピッキングの適用先の設定(Ver.2.09 では廃止)
mode ピッキングモード
  K_GENSHI_CENTER_BOTTOM 下部中心
  K_GENSHI_CENTER_TOP 上部中心
  K_GENSHI_LENGTH_BOTTOM 下部長さ
  K_GENSHI_LENGTH_TOP 上部長さ

void SetSteelPickCoordMode(int mode);
形鋼生成における座標ピッキングの適用先の設定(Ver.2.09 では廃止)
mode
  K_STEEL_ORIGIN 物体原点
  K_STEEL_SHIFT 原点からの離れ

void SetBridgePickCoordMode(int mode);
橋生成の座標ピッキングの適用先の設定
Mode
  K_BRIDGE_ORIGIN 橋原点
  K_BRIDGE_SIZE 橋サイズ

void SetLightDefaultParam(s3Light *l);
光源ユニットにデフォルト値をセットする
l 光源ユニットポインタ

void SetCameraDefaultParam(s3Camera *cam);

透視図の視点等にデフォルト値をセットする
cam カメラ構造体（メモリ・ブロック）のアドレス

void SetSceneDefaultParam(s3Scene *scn);
シーンにデフォルト値をセットする
scn シーン構造体のアドレス

void SetColorGroupFaceAll(d3Group *g, float *rgba);
グループ内の全ての面にカラーをセットする
g グループポインタ
rgba RGBA
（子グループがあれば、そのグループにもセットする）

void SetColorFace(d3Face *f, float *rgba);
面にカラーをセットする
f 面構造体のアドレス
rgba RGBA

void SetColorSameColorFace(d3Face *f, d3Group *g, float *rgba);
グループ内の同色面にカラーをセットする
f 面(メモリ・ブロック)のアドレス
g グループ(メモリ・ブロック)のアドレス
rgba カラー値(R,G,B,A)

int SetNewMaterial(char *name);
指定された名称のマテリアルをリストに登録し、その具体的内容をファイルから読み込む。
name マテリアル名称
返り値 マテリアル ID

void SetMaterialGroup(d3Group *g, int id);
グループ、配下の面、および子グループにマテリアルを設定する。
g グループポインタ
id マテリアル ID

void SetMaterialFace(d3Face *f, int id);
面にマテリアルを設定する。

f 面ポインタ
id マテリアル ID

void SetPickWorldCoord(double x, double y, double z);
ワールド座標値をスタティック変数に記録する。
x, y, z ワールド座標

int SetNew(int type, int width, int height);
新規作成時の初期化を行う。
type: 編集するデータのタイプ　(K_LSS_S：シーン、K_LSS_G：ジオメトリ)
width 表示エリアの幅
height 表示エリアの高さ

int SetGroupChildlen(d3Group *parent, int num, d3Group **childlen);
グループに複数の子グループをセットする。
parent 親グループポインタ
num 子グループ数
childlen 子グループのアドレスの配列
返り値 成功時:TRUE　エラー時:FALSE

int SetSectionFileToList(char *file, int *count, char ***list);
断面リストファイルを読み込み断面ファイルリストにセットする。
file 断面リストのファイル名
count 項目数を格納する変数のアドレス
list リスト文字列の配列を指すポインタのアドレス
返り値 成功時:TRUE　エラー時:FALSE

void SetNoriMode(int flag);
道路法面編集ダイアログを使用中であることを示すフラグをセットする
flag TRUE(使用中)/FALSE(非使用)

②問い合わせへの回答　　(初版の（６）データ取得関数の一部)

int GetFileType();
LSS ファイルタイプを取得する(Ver.2.09 では廃止)
返り値 ＬＳＳファイルタイプ

int GetFileSelectMode();
ファイル選択モードを取得する。
返り値 ファイル選択モード
  K_LSS_S シーン読み込み
  K_LSS_G ジオメトリ読み込み
  K_LSS_G_CHILD ジオメトリの追加読み込み
  K_READ_MODEL シーンモデルの読み込み
  K_IMAGE イメージモデルの読み込み
  K_SAVE_SGI_IMAGE SGI ファイルの保存
  K_SAVE_JPEG_IMAGE JPEG ファイルの保存
  K_SAVE_AS ファイルの保存
  K_SAVE_MODEL モデルの保存

int GetViewType();
表示画面（ビュー）のタイプを取得する
返り値 ビュータイプ
  G3_CAM_FRONT 南立面図（正面図）
  G3_CAM_TOP 平面図
  G3_CAM_SIDE 東立面図（側面図）
  G3_CAM_UTARA 北立面図
  G3_CAM_BARAT 西立面図
  G3_CAM_PERS 透視図（パース）

char *GetSceneFileName();
シーン(LSS-S)ファイル名を取得する
返り値 ファイル名

int GetSceneCurrentIndex();
現在表示・編集中のシーン番号を取得する。
返り値 シーン番号

int GetScene(s3Scene ***scn_array);
シーン配列を取得する（Ver.2.09 では廃止）
scn_array シーン配列へのポインタのアドレス
返り値 シーン数

char *GetGeometryFileName();
ジオメトリファイル名を取得する
返り値 ファイル名

int GetGeometryGroup(d3Group ***rgrp_array);
ジオメトリのグループを取得する。
rgrp_array ルートグループのアドレスの配列を指すポインタのアドレス
返り値 ルートグループ数

s3Camera *GetPers();
透視図表示に関する情報を取得する
返り値 カメラ構造体（メモリ・ブロック）のアドレス

g3Ortho *GetOrtho(int type);
オルソ系画面に関する情報を取得する。
type タイプ
  G3_CAM_FRONT 南立面図（正面図）
  G3_CAM_TOP 平面図
  G3_CAM_SIDE 東立面図（側面図）
  G3_CAM_UTARA 北立面図
  G3_CAM_BARAT 西立面図
返り値 オルソ構造体（メモリ・ブロック）のアドレス

s3LightGroup *GetGeometryLightGroup();
ジオメトリの光源グループを取得する。
返り値 光源グループ(メモリ・ブロック)のアドレス

s3EffectGroup *GetGeometryEffectGroup();
ジオメトリの効果グループを取得する。
返り値 効果グループ(メモリ・ブロック)のアドレス

float GetGeometryTime();
ジオメトリの時間を取得する。
返り値 日数（浮動小数）

void GetImage(s3Image **img);
イメージを取得する。
img イメージ構造体(メモリ・ブロック)へのポインタのアドレス

int GetInputMainDrawareaMode();
メイン表示エリアのモードを取得する。オルソ系編集ダイアログが開いている場合、FALSE。
返り値 モード(TRUE/FALSE)

int GetSelectCount();
画面でのオブジェクト選択で、クリック地点に重なり合うオブジェクト数を返す。
返り値 選択数

int GetSelCurrentIndex();
重なり合うオブジェクトから現在選択されているオブジェクトの番号を返す。
返り値 インデックス

int GetSelectType();
選択タイプを返す。
返り値 選択タイプ
  G3_SEL_GROUP グループ
  G3_SEL_FACE 面
  G3_SEL_SAMECOLFACE_IN_GROUP グループの同色面

int GetLayoutMode();
移動・回転・スケールの編集モードを返す。
返り値 編集モード(K_LAYOUT)

int GetSimMode();
景観シミュレータの動作を返す
返り値 モード
 K_VIEWER　　メイン画面（ビューワ）
 K_LAYOUT　　移動・回転・スケール
 K_BRIDGE　　橋
 K_OPTION　　ユーザー定義のパラメトリック部品

int GetCheckWinMode();

チェックボックスの状態等を返す(Ver.2.09 では使用していない)。
返り値 モード

int GetMaterialSelectionMode();
マテリアルの選択モードの取得
返り値 ON:TRUE　OFF:FALSE
ビット毎に画面を区別する。
　オリジナルのマテリアル編集画面：１
　オリジナルのテクスチャ編集画面：２
　オリジナルのマテリアルファイル画面：４
　オリジナルのマテリアル選択画面：８
　グラフィックなマテリアル編集:１６
　グラフィックなテクスチャ編集：３２
　テクスチャの貼り方：６４
　カラーセット編集：１２８
　複数開いている場合には、OR の値を返す。

int GetMultiLayoutCopyMode();
配置/コピーの編集中であることを示すフラグを返す。
返り値 ON:TRUE OFF:FALSE

int GetCalledDBFlag();
データベースからの起動フラグを取得する。
返り値 TRUE/FALSE

int GetCalledDB(int width, int height);
データベースからの起動モードで各情報をセットする(2.09 では使用しない)
width 表示エリアの幅
height 表示エリアの高さ
返り値 呼ばれたデータベースの種類

char *GetNameStr();
内部通信ファイル名を取得する。
返り値 ファイル名

char *GetFileName();

内部通信ファイル名をフルパスで取得する。メモリ・ブロックを作成しこの上に文字列を作成するため、使用後は解放する必要がある。
返り値 フルパスのファイル名(メモリ・ブロック)のアドレス

void GetGridPoints(double *pnts, double xgrid, double ygrid, double *opnts);
参照点に最も近いグリッド上の点を取得する。
pnts 参照点の座標値[X,Y,Z]のアドレス
xgrid 横のグリッド間隔
ygrid 縦のグリッド間隔
opnt 最も近いグリッド上の点の座標値[X,Y,Z]を格納するアドレス

void GetBackupCameraParam(s3Camera *cam);
バックアップした視点情報を返す。
cam カメラ情報を返すための構造体のアドレス。
 cam が NULL である場合には、バックアップされた情報を解放する。
 cam->name が NULL ではない場合、これを解放した上で、新たなメモリ・ブロックを割り当て、バックアップされたカメラ名称のコピーを作成する。バックアップのカメラ名称は解放する。

void GetBackupGridParam(int *mode, double *interval);
バックアップしたグリッド情報を取得する
mode モードを格納する変数のアドレス
interval 間隔を格納する変数のアドレス

void GetLayoutParam(double *tran, double *rot, double *scl, double *center);
移動/回転/スケールのパラメータを取得する。
tran 移動量 (X,Y,Z) を格納する配列のアドレス
rot 回転 (X,Y,Z) を格納する配列のアドレス
scl スケール (X,Y,Z) を格納する配列のアドレス
center 回転中心 (X,Y,Z) を格納する配列のアドレス

void GetInitLayoutMatrix(d3Matrix m);
バックアップしたレイアウトマトリクスを取得する
m マトリクスを格納する配列のアドレス

int GetExitFlag();

終了フラグを取得する
flag TRUE/FALSE

void GetBackupLightGroupLight(s3Light *l)
バックアップされた光源ユニットを取得する。
l 光源ユニット構造体を格納するアドレス

void GetBackupLightGroup(s3LightGroup **lg);
バックアップした光源グループを取得する
lg 光源グループ構造体を格納するメモリ・ブロックを指すポインタのアドレス

void GetBackupSceneLightGroupAll(s3LightGroup ***lgs);
バックアップしたシーンの全ての光源グループを取得する
lgs 光源グループを指すポインタ配列のアドレスを格納するポインタのアドレス

int GetSelectLightIndex();
選択した光源ユニットのインデックスを取得する。
返り値 インデックス

int GetNewLightFlag();
新しい光源ユニットを作成するフラグを取得する。
返り値 TRUE/FALSE

void GetBackupMaterial(float *rgba, float *specular);
バックアップしたマテリアルの内容を取得する。
rgba カラー値[R,G,B,A]を格納するアドレス
specular 反射率[R,G,B,A]を格納するアドレス

int SetGroupMaterialToRGBAandSpecular(float *rgba, float *specular, d3Group *g);
グループマテリアルの RGBA とスペキュラーを取得する
rgba カラー値[R,G,B,A]を格納するアドレス
specular 反射率[R,G,B,A]を格納するアドレス
g グループを指すポインタ
返り値 成功時:TRUE　エラー時:FALSE

void GetMaterialDrawAreaMaterial(int seltype, g3SelPath *spath, float *rgba, float

*specu, dbMaterial *mtl);
マテリアル編集画面の色球に表示するためのマテリアル情報を取得する。選択したオブジェクトにマテリアルが定義されている場合にはこれを用いる。
seltype セレクトタイプ（面、グループ）
spath セレクトパス
rgba カラー値（マテリアルが定義されていなかった場合に使用する）
specu 反射率 （マテリアルが定義されていなかった場合に使用する）
mtl マテリアル情報の格納先のアドレス

void GetMaterialDrawAreaColor(int seltype, g3SelPath *spath, float *rgba, float *specu, dbMaterial *mtl);
サンプル色球に表示するカラーとテクスチャを取得する。
seltype セレクトタイプ（面、グループ）
spath セレクトパス
rgba 適用するカラー値
specu 適用する反射率
mtl 結果を格納するマテリアル構造体のアドレス

void GetBackupMaterialLab(float *lab);
バックアップしたLabを取得する。
lab Lab情報の格納先のアドレス

void GetBackupGroup(d3Group **g);
バックアップファイルしたグループを取得する。
g グループを指すポインタのアドレス

void GetBackupFace(d3Face **f);
バックアップした面を取得する。
f 面を指すポインタのアドレス

void GetTextureParam(double *org, double *horizontal, double *vertical, double *soutai, double *scale);
テクスチャパラメータを取得する。
org テクスチャ原点（X,Y,Z）を格納する配列のアドレス
horizontal uベクトル（X,Y,Z）を格納する配列のアドレス
vertical vベクトル（X,Y,Z）を格納する配列のアドレス

soutai 相対移動量 (X,Y) を格納する配列のアドレス
scale スケール (X,Y) を格納する配列のアドレス

char *GetSelObjectTextureName(int type, g3SelPath *spath);
選択されているオブジェクトのテクスチャを取得する。
type セレクトタイプ (面、グループ)
spath セレクトパス
返り値 テクスチャファイル名

char *GetTextureFileName();
記憶してあるテクスチャファイル名を返す。

void GetAutoTexList(g3TsTop **ts);
自動貼り付けテクスチャのリストを返す。
ts g3TsTop を格納するメモリ・ブロックを指すポインタのアドレス

float GetBackupTime();
バックアップした時間を取得する
返り値 経過日数 (浮動小数)

int GetBackupAntialias(int *count);
バックアップしたアンチエリアシングの状態を取得する
count 精度を格納する変数のアドレス
返り値 ON:TRUE OFF:FALSE

int GetChildGroupCount(d3Group *g);
子グループ数を取得する
g 検査するグループ構造体のアドレス
返り値 子グループ数

void GetPrimitiveParam(double *toppoint, double *bottompoint, double *topradius,
double *bottomradius,int *n, char *name);
原始図形パラメータを取得する(2.09 では使用していない)
toppoint 上部原点 (X,Y,Z) を格納する配列のアドレス
bottmpoint 下部原点 (X,Y,Z) を格納する配列のアドレス
topradius 上部半径 を格納する変数のアドレス

bottomradius 下部半径 を格納する変数のアドレス
n 角数を格納する配列のアドレス
name グループ名称(メモリ・ブロック)を指すポインタのアドレス

void GetSteelParam(double *org, double *param, double *sft, char *name);
形鋼パラメータを取得する(2.09 では使用していない)
org 物体原点 (X,Y,Z) を格納する配列のアドレス
param パラメータ (a,b,c,d,h) を格納する配列のアドレス
sft シフト (X,Y) を格納する配列のアドレス (G3_LSTEEL の場合のみ有効)
name グループ名称(メモリ・ブロック)を指すポインタのアドレス

void GetBridgeParam(double *org, double *size, char *name);
橋パラメータを取得する
org 橋原点 (X,Y,Z) を格納する配列のアドレス
size 幅、長さ、高さ を格納する配列のアドレス
name グループ名称(メモリ・ブロック)を指すポインタのアドレス

int GetModifyFlag();
編集操作の有無を示すフラグを取得する。終了時に保存の確認を行うために参照する。
返り値 TRUE/FALSE

int GetNewType();
新規作成のタイプを取得する
返り値 タイプ
  K_LSS_S シーン
  K_LSS_G ジオメトリ

int GetNewDataFlag();
「新規作成」が選択されたことを示すフラグを取得する
返り値 TRUE/FALSE

int GetOpenFlag();
「ファイルを開く」が実行されたことを示すフラグを取得する
返り値 TRUE/FALSE

int GetPickCoordMode();

座標ピッキングモードを取得する。メイン画面が平面・立面系の表示モードの時に、画面クリックに対応した座標値を別のダイアログに送出する必要があるかどうかを判定する。
返り値 TRUE/FALSE

int GetLayoutPickCoordMode();
移動・回転・スケールの編集で、メイン画面クリックにより取得された座標値をどのパラメータに適用するかに関する情報を取得する。
返り値
  K_LAYOUT_MOVE：平行移動
  K_LAYOUT_ROTATE：回転
  K_LAYOUT_SCALE：拡大縮小(使用しない)
  K_LAYOUT_ROT_CENTER：回転の中心

int GetGenshiPickCoordMode();
原始図形生成の座標ピッキングモードを取得する
返り値 ピッキングモード
  K_GENSHI_CENTER_BOTTOM 下部中心
  K_GENSHI_CENTER_TOP 上部中心
  K_GENSHI_LENGTH_BOTTOM 下部長さ
  K_GENSHI_LENGTH_TOP 上部長さ

int GetSteelPickCoordMode();
形鋼生成の原始図形成の座標ピッキングモードを取得する
返り値
  K_STEEL_ORIGIN 物体原点
  K_STEEL_SHIFT 原点からの離れ

int GetBridgePickCoordMode();
橋生成の原始図形成の座標ピッキングモードを取得する
返り値
  K_BRIDGE_ORIGIN 橋原点
  K_BRIDGE_SIZE 橋サイズ

void GetPickWorldCoord(double *x, double *y, double *z);
ピッキングしたワールド座標値を取得する
x, y, z ワールド座標の各格納先

void GetReadChildGroupPath(d3PickPath **path);
何もしない（非使用）

int GetSameSceneBackImageCount(int scnnum, s3Scene ** scns, s3Image *img);
シーンの背景イメージが他のシーンでも使用されている数を取得する(2.09では非使用)
scnnum シーン数
scns シーンポインタへのポインタ
img 背景イメージポインタ
返り値 0:なし
1:自分のシーンのみ
2～:使用されているシーン数

int GetSameSceneFrontImageCount(int scnnum, s3Scene ** scns, s3Image *img);
シーンの前景イメージが他のシーンで使用されている数を取得する(2.09では非使用)
scnnum シーン数
scns シーンポインタへのポインタ
img 前景イメージポインタ
返り値 0:なし
1:自分のシーンのみ
2～:使用されているシーン数

int GetRoadSectionList(char *file, int *count, char ***list);
道路断面リストを取得する
file 道路断面リストを保存しているファイル名
count 項目数の格納先
list リスト文字列を指すポインタの配列を指すポインタのアドレス
返り値 成功時:TRUE　エラー時:FALSE

int GetRiverSectionList(char *file, int *count, char ***list);
河川断面リストを取得する
file 河川断面リストを保存しているファイル名
count 項目数の格納先
list リスト文字列を指すポインタの配列を指すポインタのアドレス
返り値 成功時:TRUE　エラー時:FALSE

int GetNoriMode();
道路法面モードを取得する
返り値 TRUE/FALSE

（３）ライブラリ支援機能

① OpenGL 画面の初期化

(pixel.c)
void set_pixel(HDC hDC);
hDC で指定されたデバイス・コンテキスト（画面等）に適したピクセル・フォーマット（色数など）を見つけ、設定する。

(参考)
wg3.c にある、
int wg3EntryDrawarea(void *w, void *c);
void wg3DeleteDrawrarea(void *w);
int wg3AssignDrawarea(void *w);
int wg3Redraw(void *w);
等は、ほぼ全ての OpenGL 画面を有するウインドウを表示するダイアログで用いる重要な関数であり、定義は、wg3.h に記述されている。しかし、Ver.2.09 ではこれらのソースコードは、Ｇ３ＤＲＬライブラリの中に含まれているので、アプリケーション・ライブラリではない。

② OpenGL による画面の印刷処理

(B3Bitmap.c)
int ColorPrintDIB(RECT rect);
rect で指定された画面領域を、環境設定ファイルで指定されたプリンターに出力する
戻り値　TRUE(成功)/FALSE(失敗)

HDIB BitmapToDIB(HBITMAP hBitmap, HPALETTE hPal);
hBitmap で指定される画面のビットマップを DIB 画像に変換する
戻り値　変換結果の DIB(Device Independent Bitmap)へのハンドル

(Print.c)
int PrintDIB(HDIB hDib, WORD fPrintOpt, WORD wXScale, WORD wYScale, LPSTR szJobName);
hDib で指定される DIB をプリンターに出力する。
戻り値：0（成功）またはエラーコード(dibapi.h で定義)

（４）ダイアログ・ハンドラ支援機能

①ファイル入力　　（初版の（２）データ読み込み関数）

int LoadSceneFile(char *file, int width, int height);
シーンファイル(LSS-S)を読み込んで、指定したサイズの初期画面を開く。
file ファイル名
width 表示エリアの幅
height 表示エリアの高さ
返り値 成功時:TRUE　エラー時:FALSE

int LoadGeometryFile(char *file, int width, int height);
ジオメトリファイル(LSS-G)を読み込んで、指定したサイズの初期画面を開く。
file ファイル名
width 表示エリアの幅
height 表示エリアの高さ
返り値 成功時:TRUE　エラー時:FALSE

int LoadImageFile(char *file, int width, int height);
イメージファイルを読み込む(2.09 では非使用)
file ファイル名
width 表示エリアの幅
height 表示エリアの高さ
返り値 成功時:TRUE　エラー時:FALSE

int ImageFileSelect(char *file);
何もしない(2.09 で非使用)
file ファイル名
返り値 成功時:TRUE　エラー時:FALSE

int FrontAndBackImageFileSelect(char *file, int scnnum, s3Scene **scns, s3Scene *scn, int type);
前景/背景イメージファイルを読み込み、シーンの背景または前景にセットする。
file イメージファイル名
scnnum シーン数(2.09 では使用しない)
scns シーンポインタへのポインタ(2.09 では使用しない)
scn 現在のシーンのポインタ(2.09 では使用しない)

type　前景か背景かの区別
 前景　K_IMAGE_FRONT
 背景　K_IMAGE_BACK
返り値 成功時:TRUE　エラー時:FALSE

int LoadSceneModelFile(char *file);
シーンにモデルを読み込む。
file ジオメトリ(LSS-G)ファイル名
返り値 成功時:TRUE　エラー時:FALSE

void ReadFromFile(int *seletype, int *filetype, char *filename);
内部通信ファイルを読み込む
seletype データベースタイプの格納先
filetype ファイルタイプの格納先
filename ファイル名の格納先

②ファイル出力　(初版の（３）データ書き込み関数)
void WriteCommuFileDrawingData();
終了時に編集結果に関する情報を内部通信ファイル(tmp003.txt)に書き込む。

void WriteToFile(int seletype, int filetype, char *filename);
内部通信ファイルに書き込む ファイルの種類は、予め SetNameStr 関数で指定する。基幹部分とデータベース検索機能の間でファイル名等の情報を通信する。
seletype データベースタイプ
filetype ファイルタイプ
filename ファイル名

void OutputGeometryFile(int summitdelflag, int num, d3Group **root, char *file);
サミットグループを考慮して、ジオメトリをファイル(LSS-G)に保存する。
summitdelflag ≧0 ならサミットグループが削除されている
≧-1 ならサミットグループが削除されていない
num ルートグループ数
root ルートグループポインタへのポインタ
file ファイル名
SAVE Geometry File()よりコールされる

void SaveGeometryFile(char *file, int num, d3Group **root);
ジオメトリをファイル(LSS-G)に保存する。
file ファイル名
num ルートグループ数
root ルートグループを指すポインタ配列のアドレス

void OutputSceneFile(int num, s3Scene **scns, char *file);
シーンをファイル(LSS-S)に保存する。
num シーン数
scns シーンを指すポインタ配列のアドレス
file ファイル名

void SaveSceneModelFile(char *file, int num, d3Group **root);
シーンのモデルを保存する
file ファイル名
num ルートグループ数
root ルートグループを指すポインタ配列のアドレス

③システム関数の起動　（初版の（４）データ構築関数）
void CallExtCommand(int type);
外部コマンドをコールする(2.09 では非使用)
type 外部コマンドのタイプ
  K_CUBE 直方体
  K_SPHERE 球
  K_CYLINDER 円柱
  K_FLAICYLI 円すい、円すい台
  K_CONE 角柱
  K_FLATCONE 角すい、角すい台
 ※本関数をコールする前に SetPrimitiveParam()でパラメータをセットしておくこと

void CallSteelCommand(int type);
形鋼コマンドをコールする(2.09 では非使用)
type 形鋼タイプ
  G3_HSTEEL H 形鋼
  G3_CSTEEL 溝形鋼
  G3_TSTEEL T 形鋼

G3_LSTEEL L 形綱
※本関数をコールする前に SetSteelParam()でパラメータをセットしておくこと

int CallBridgeCommand(int type);
橋コマンドをコールする
type 橋タイプ
K_BRIDGE_KETA 桁橋
K_BRIDGE_ARCH アーチ橋
K_BRIDGE_TRAS トラス橋
K_BRIDGE_TURI 吊橋
K_BRIDGE_SHACHO 斜張橋
返り値 成功時:TRUE　エラー時:FALSE
※本関数をコールする前に SetBridgeParam()でパラメータをセットしておくこと

int CreateRoadGroup(double width, double height, char *sectionfile, int pnum,
double *locus, d3Group **grps);
道路グループを生成する
width 道路幅
heigth 道路高
sectionfile 断面ファイル名
pnum 点数
locus 軌跡線の座標列 (X,Y,Z･･･の並び)
grps 生成した道路を記述するグループを指すポインタのアドレス
返り値 成功時:TRUE　エラー時:FALSE

d3Group *CreatePlaneGroup(double left, double right, double bottom, double top,
double z);
面を生成する
left, right, bottom, top 面のサイズ
z 高さ
返り値 生成したグループ構造体のアドレス

d3Group *CreateCutGroup(int count, double *points, int inside);
切断したグループを生成する (land プラグイン DLL に移管)
count 切断面の頂点数
points 切断面座標 (X,Y,Z･･･の並び)

inside 0:外側
1:内側
返り値 生成したグループ構造体のアドレス

int CreateDivtriGroup(double area);
グループを三角形分割する(land プラグイン DLL に移管)
area 三角形のエリアサイズ
返り値 成功時:TRUE　エラー時:FALSE

int CreateCombineGroup(float value, float ratio);
グループ内の面を結合する (land プラグイン DLL に移管)
value 値
ratio レート
返り値 成功時:TRUE　エラー時:FALSE

int CreateTunnel(int tCnt, double *tPoints, double *startP, double *endP)
トンネルを生成する (tunnel プラグイン DLL に移管)
tCut 断面頂点数
tPoints 断面頂点列 (X,Y,Z･･･の並び)
startP 始点 (X,Y,Z)
endP 終点 (X,Y,Z)
返り値 成功時:TRUE　エラー時:FALSE

（５）特殊演算機能

①データ形式変換

void CnvColLabtoRGB(float l, float a, float b, float *R, float *G, float *B);
カラーを記述する形式を Lab から RGB に変換する
l, a, b Lab 値
R, G, B RGB 値へのポインタ

void CnvColRGBtoLab(float R, float G, float B, float *l, float *a, float *b);
カラーを記述する形式を RGB から Lab に変換する
R, G, B RGB 値へのポインタ
l, a, b Lab 値

XYZ matahari(double Id, double Kd, int Bulan, int Hari, int Jam, int Mun);

太陽方位を求める
ld 緯度
Kd 経度
Bulan 月
Hari 日
Jam 時
Mun 分
返り値 太陽方位（座標）

char *CheckColorValue(char *string, int *setflag);
色の値を示す浮動小数文字列を整形する。
string 処理前の文字列
setflag 整形の有無を格納する（変更あり:TRUE、変更なし:FALSE）
返り値 新しい文字列(static char[])

void GetClampZeroToOneDouble(double *value);
double 型の value を 0≦≦1 にクランプする
value 値（入力と出力）

void GetClampZeroToOneFloat(float *value);
float 型の value を 0≦≦1 にクランプする
value 値（入力と出力）

void GetClampMaxZeroToZeroOneFloat(float *val1, float *val2, float *val3);
float 型の 3 つの値の内最大のものを 1 として、3 つの値を 0≦≦1 にクランプする
val1 値 1（入力と出力）
val2 値 2（入力と出力）
val3 値 3（入力と出力）

char *ReformStringDouble(char *str);
str を double の値にし、余分な「0」を取る
str 文字列
返り値 新しい文字列

char *ReformStringFloat(char *str);
str を float の値にし、余分な「0」を取る

str 文字列
返り値 新しい文字列

char *ReformStringInt(char *str);
str を int の値にし、余分な「0」を取る
str 文字列
返り値 新しい文字列

char *ReformStringColorValue(char *str);
str を色の値にし、余分な「0」を取る
str 文字列
返り値 新しい文字列

char *GetFileNameFromPath(char *path);
パスからファイル名を取得する
path パス
返り値 ファイル名

②画面設定等

void GetMaxWindowSize(int w, int h, int wa, int ha, int *maxw, int *maxh);
表示可能なウインドウの最大サイズを取得する
w 表示エリアの幅
h 表示エリアの高さ
wa ウインドウフレームの幅の合計
ha ウインドウフレームの高さ
maxw ウインドウの最大幅
maxh ウインドウの最大高さ

int GetKeepAspectHeight(double aspect, int width);
アスペクト比が変わらない高さを取得する
aspect アスペクト比
width 幅
返り値 高さ

int GetKeepAspectWidth(double aspect, int height);
アスペクト比が変わらない幅を取得する

aspect アスペクト比
height 高さ
返り値 幅

int GetClampMinus1To1(double n, double max, double *clamp);
n を-1≦≦1 にクランプする
n 値
max クランプする幅
clamp クランプされた値
返り値 成功時:TRUE　エラー時:FALSE

③データの複写　（初版の（８）データ複写関数）

void SubstitutionGroupMaterialAndColor(d3Group *g, d3Group *org);
グループのマテリアルと配下の面のマテリアル及びカラーをコピーする
g コピー先グループ構造体のアドレス
org コピー元グループ構造体のアドレス

void SubstitutionFaceMaterialAndColor(d3Face *f, d3Face *org);
面のマテリアル及びカラーをコピーする
f コピー先の面構造体のアドレス
org コピー元の面構造体のアドレス

void SubstitutionGroupTexture(d3Group *g, d3Group *org);
グループと配下の面のテクスチャをコピーする
g コピー先グループ構造体のアドレス
org コピー元グループ構造体のアドレス

void SubstitutionFaceTexture(d3Face *f, d3Face *org);
面のテクスチャをコピーする
f コピー先の面構造体のアドレス
org コピー元の面構造体のアドレス

s3LightGroup *CopyLightGroup(s3LightGroup *lg);
光源グループをコピーする。光源グループ、配下の光源ユニット、それらの名称は、新たなメモリ・ブロックを割当て、内容をコピーしたものである。

lg 光源グループポインタ
返り値 コピーされた光源グループポインタ

④データの検査　（９）問い合わせ関数：
int CheckIsDigit(int len, char *s);
0～9 の値かどうかチェックする
len 文字数
s 文字列
返り値 TRUE/FALSE

int CheckIsComma(int len, char *s);
文字列 s の先頭から len 文字の中に小数点（.）があるかどうかチェックする。
len 文字数
s 文字列
返り値 TRUE：ある/FALSE：ない

int CheckIsMinus(int len, char *s);
文字列 s の先頭から len 文字の中にマイナス（－）があるかどうかチェックする。
len 文字数
s 文字列
返り値 TRUE：ある/FALSE：ない

int CheckIsDigitComma(int len, char *s);
文字列 s の先頭から len 文字が全て 0～9 の値と小数点（.）かどうかチェックする。
len 文字数
s 文字列
返り値 TRUE：全て合格/FALSE：違う文字がある

int CheckIsDigitCommaMinus(int len, char *s, int ins, char *ss);
文字列 s の先頭から len 文字が 0～9 の値、小数点（.）またはマイナス(－)かどうかチェックする。2.09 では非使用。
len 文字数
s 文字列
ins 文字列のインデックス
ss インデックスの文字列
返り値 TRUE/FALSE

int CheckIsAlpha(int len, char *s);
A～z、0～9 の値、マイナス(－)、またはアンダーバー（_）かどうかチェックする
len 文字数
s 文字列
返り値 TRUE/FALSE

int ChangeString(char **str1, char **str2);
文字列の変化を査定する。
str1 文字列 1 へのポインタ
str2 文字列 2 へのポインタ
返り値 strcmp に準ずる

int IsIdentMatrix(d3Matrix m);
単位行列かチェックする。
m マトリクス
返り値 TRUE/FALSE

int IsIdentGroupLinkMatrixAll(d3Group *g);
グループの下方リンクマトリクスが全て単位行列かどうかチェックする。
g グループ構造体のアドレス
返り値 TRUE：全て単位/FALSE：違うものあり

int IsFile(int type, char *file);
ファイルの存在をチェックする。
type 探索先 (E3_FILE_PATH_XXX)
file ファイル名
返り値 ファイルが存在する:TRUE　ファイルが存在しない:FALSE

⑤異常への対応（初版の（１０）その他）
void ErrorInfo(int err_type, char *info);
コンソールにエラーメッセージを表示する。2.09 においては、メッセージをｚ３ライブラリの関数 z3Message(1326,str)を用いて出力する。
err_type エラータイプ
info 文字列

void KsimExit();
シミュレータを終了させる。

void KdbmsExit();
データベースをクローズする。」

⑥その他の特殊な処理

dbImage* ambilLOD(char* name)
環境設定ファイル kdbms.set で指定した画像格納用ディレクトリの下のサブディレクトリ LOD の中に格納されている縮小画像ファイルをロードする。ファイルが存在しない場合には NULL を返す。
Name:画像ファイル名

void AppendLog(char*mes)
所定のログファイルの末尾に、文字列 mes と改行を追加する。
mes:追加するメッセージ文字列

void BebasGeometryLightStruct()
LSS-G 編集モードで使用している光源グループを全て解放する。

void BebasGeometryStruct()
LSS-G 編集モードで編集しているグループおよびそれらの配下の全ての子グループを解放する。

void BebasOrthoStruct()
LSS-G 編集モードで使用する平面図表示のためのパラメータを解放する。

void BebasSceneStruct()
LSS-S 編集モードで使用している全てのシーンのデータを解放する。

dbImage BuatCitrakecil(dbImage* image)
既存の画像データから、６４×６４ピクセルの縮小画像を作成する(dataope.c)。
image:元のイメージ

void CheckKerja(char* file)
要求されたディレクトリが、現在の作業環境と一致するか調べ、異なっていた場合には、

作業環境を引数で指定されたディレクトリに変更する。
file:新たに設定する作業環境（ディレクトリ）のディレクトリ

int CountFaces(d3Group *g)
指定したグループを含む配下のグループの総数を数える。
g:操作対象となるグループへのポインタ

int CountChosts(d3Group *g)
指定したグループとその配下のグループにある、幽霊グループ（表示する面も、配下のグループも持たないグループ）の総数を数える。
g:操作対象となるグループ

int CountFaces(d3Group *g)
指定したグループとその配下のグループの面の総数を数える
g:操作対象となるグループ

int DeleteGhosts(d3Group *g)
指定したグループとその配下のグループから、幽霊グループ（表示する面も、配下のグループも持たないグループ）を検出して削除する。削除の結果、新たに幽霊グループとなったグループも削除する。削除した総数を返す。
g:操作対象となるグループ

void FaceInvert(d3Face *f)
面の頂点列を逆順にする。法線等は元のままである。
f:操作対象となる面

void File2DirectSaveAllGroups()
ルートグループ以下の、外部ファイル参照を含む全てのグループを一つのファイルとして保存するようにデータ形式を変換する。ファイル参照された子グループのファイル属性は外されるが、これに代わり「&FILE:ファイル名」の形の属性を新たに付して、元は独立したファイルであったという情報は残す。

char* GetFileName()
予め SetNameStr コマンドで保存してあるファイル名の前に環境設定ファイルで定義されたテンポラリ・ディレクトリを加えたフルパスを、新たに作成したメモリ・ブロック上に作成し、このアドレスを返す。

char* GetFileNameFromPath(char *path)
フルパスで記述したファイル名から、ディレクトリ名等を除いたファイル名のみを検出して返す。なお、返す値は、元の文字列中の、ファイル名の先頭のアドレスであるため、これを用いたメモリ解放、元の文字列が無効化した後のこのポインタの使用は障害の原因となる。
path 元のフルパスで記述したファイル名

s3EffectGroup* GetGeometryEffectGroup()
LSS-G 編集モードにおける効果グループへのポインタを返す。

char* GetGeometryFileName()
LSS-G 編集モードにおけるファイル名を返す。このファイル名は、ファイルを読み込んだ際、または編集結果をファイル保存した際に設定される。

void GetIngat()
SetIngat の結果を返す

char* GetLangsung()
SetLangsung の結果を返す

int GetLobangMode()
SetLobangMode の結果を返す

int GetLssFileType()
編集中のファイルの種類を返す
LSS-S ならば、K_LSS_S 1
LSS-G ならば、K_LSS_G 2

void GetMateFromPath(g3SelPath*spath, int mode, float* rgba, float* specu)
選択されているオブジェクトからマテリアルの内容に関する情報を取得する。
spath:オブジェクトの選択状態を示す構造体
mode:情報取得方法（現在は何を指定しても結果は同じ）
rgba:取得されるカラー情報
specu:取得される鏡面反射率

int GetMaterialColorMode 廃物
char* GetNameStr()
保存してあるファイル名(アドレス)を返す

int GetOptionZ(double **array)
X,Y 座標が同じスナップ点に高さの異なる頂点が重複している場合に、Z 値の配列を array に格納し、重複数を返す。

g3Ortho *GetOrtho(int type)
オルソ系画面、つまり平行投影する平面図、４方向からの立面図に関して、パラメータを返す。図面の種類を type で指定する。g3Ortho 構造体は、表示する左右・上下・前後範囲を定義する。

s3Camera *GetPers()
透視図のパラメータを返す。パラメータの内容は名称、視点、注視点、天頂ベクトル、焦点距離、縦横比、前後範囲である。

int GetPrimitiveMode()
保存してある原始図形の種類を返す。
0:CUBE 1:SPHERE 2:CYLINDER 3:CONE 4:FLATCYLI 5:FLATCONE 7:SWEEP1 8:SWEEP2

char* GetSceneFileName()
編集中のシーンのファイル名を返す。

char *GetSelObjectTextureName(int type, g3SelPath *spath)
現在選択されているオブジェクトに適用されているテクスチャの名称(ポインタ)を返す。テクスチャが無ければ、NULL を返す。
type: 選択モード (グループ、面)
spath: 現在の選択状態を示す構造体

int GetSnapPoints(double *pnts, double *opnts)
pnts で示された座標に XY 平面的に最も近い既存頂点を探し､その座標を opnts に格納する。頂点が存在しなければ、pnts と同じ座標値を opnts に格納し、ゼロを返す。成功した場合の戻り値は、同じ XY 位置にある高さの異なる頂点数である。

char* GetTextureFileName()

保存してあるテクスチャのファイル名(ポインタ)を返す。
void GroupFaceInvert(d3Group *g)
引数で指定したグループおよび配下のグループの面について、頂点の順序を逆回りに変更する。法線ベクトルは変更しない。

int InitIpSalah(int mode)
インタープリターのエラー処理方法（０～４）を指定する。
０：何もしない　１：直ちに停止してメッセージを出す
２：あとでまとめてメッセージを出す（デフォルト）
３：直ちに停止するが報告しない　４：ログファイルに出力する
戻り値：以前に設定されていたモード
引数を負値とすることにより、現在設定されているモードを調べることができる。

void MelengkapiNama(char *name)
引数で示された名称に”.geo”の拡張子を追加した上で、全体を“”で囲む。元々拡張子がある場合には何もしない。引数が示すバッファにはこの処理を行うだけの余裕がなければならない。

int mstrcmp(char* s1, char* s2)
古いWindowsにおいて、長いファイル名が「XXX~1」等の形に縮約されている場合を考慮した比較を行う。
s1, s2:比較する二つの文字列

void OpenLog(char* message)
ログファイルを新たに作成した上で、文字列を出力する。
message：メッセージ

char* ReformStringColorValue(char *str)
文字列として表現されたカラー値を、0.0<=value<=1.0の範囲に整形しスタティック変数へのポインタとして返す。

char* ReformStringDouble(char *str)
文字列として表現された倍精度浮動小数を整形し、スタティック変数へのポインタとして返す。

char* ReformStringFloat(char *str)

文字列として表現された単精度浮動小数を整形し、スタティック変数へのポインタとして返す。

char* ReformStringInt(char *str)
文字列として表現された整数を整形し、スタティック変数へのポインタとして返す。

void SetFrontAndBackImageType(int type)
画像が前景か背景かの区別を指定する

void SetIngat(d3Group *g)
グループを記憶する

void SetLangsung(char*)
コマンドライン文字列を記憶する。

void SetLobangMode(int flag)
ルバング島の穴あけモードをセットする。
０：グループ
１：面

int simpanLOD(dbImageData *img)
ロードされているメモリ上の画像(img)を、縮小したJPEG画像に変換し、LODサブディレクトリに「.jpg」の拡張子を有するファイルとして保存する。

（６）ヒストリー機能

以前の枝分かれバージョン(2.5,3.2,4.0)において、ユーザーの操作を記録する（ヒストリー）と共に、このデータを利用して、エラー・リカバリー、オートデモ、リモート協調動作等の機能を実現するために開発されたライブラリである。2.09 においては、それぞれの操作におけるエラー処理等が充実したため、必要性が低くなり、またエラー処理が行われるような場合の記録方法に関しても再検討が必要であることから、ビルドから外している（９－１(1)参照)。但し、このような思想・機能が将来再び必要となる可能性もある。

①データ読み込み関数

void CheckRecoveryFile();
リカバリーファイルがあるかチェックする
あれば、確認ウインドウを表示する

②データ構築関数

char *hisCom(int cmd_type, void *detail);
コマンド文字列の生成
cmd_type コマンドタイプ
detail コマンドの内容へのポインタ
返り値 コマンド文字列
返り値 成功時:TRUE　エラー時:FALSE

char *cutTailZero(double d);
「0」を取った文字列の生成
d 数値
返り値 文字列

char *hisStrCat(char *str1, char *str2, int *cmdLen);
文字列を連結する
str1 連結先
str2 連結元
cmdLen 文字列の長さ
返り値 連結後の文字列

int ReadRecoveryFile(int *num, char ***cmd) ;

リカバリーファイルを読み込む
num コマンド数
cmd コマンド文字列ポインタへのポインタへのポインタ
返り値 成功時:TRUE　エラー時:FALSE

int ReadAutoDemoFile(int *num, char ***cmd);
オートデモファイルの読み込み
num コマンド数
cmd コマンドポインタへのポインタへのポインタ
返り値 成功時:TRUE　エラー時:FALSE

int WriteRecoveryFile(char *cmd);
リカバリーファイルに書き込む
cmd コマンド
返り値 成功時:TRUE　エラー時:FALSE

int WriteAutoDemoFile(char *cmd);
オートデモファイルに書き込む
cmd コマンド
返り値 成功時:TRUE　エラー時:FALSE

void i3RdhMallocNVec(I3RdhR2Tensor* tensor, int num);
tensor を num 分、領域を確保する
tensor I3RdhR2Tensor ポインタ
num 座標数

void i3RdhMallocHaichi(I3RdhHaichiParam** haichiparameter, int num);
haichiparam を num 分、領域を確保する
haichiparameter I3RdhHaichiparam ポインタへのポインタ
num 個数

void i3RdhMallocTexture(I3RdhTexture** texture);
texture の領域を確保する
texture I3RdhTexture ポインタへのポインタ

void i3RdhMallocMaterial(I3RdhMaterial** material);

material の領域を確保する
material I3RdhMaterial へのポインタへのポインタ

void i3RdhMallocLight(I3RdhLight** light);
light の領域を確保する
light I3RdhLight のポインタへのポインタ

void i3RdhMallocCameraPos(I3RdhCameraPos** abstract, int num);
abstract を num 分、領域を確保する
abstract I3RdhCameraPos へのポインタへのポインタ
num 個数

void i3RdhMallocCube(I3RdhCube** cube);
cube の領域を確保する
cube I3RdhCube へのポインタへのポインタ

void i3RdhMallocBall(I3RdhBall** ball);
ball の領域を確保する
ball I3RdhBall へのポインタへのポインタ

void i3RdhMallocColumn(I3RdhColumn** column);
column の領域を確保する
column I3RdhColumn へのポインタへのポインタ

void i3RdhMallocCone(I3RdhCone** cone);
cone の領域分を確保する
cone I3RdhCone へのポインタへのポインタ

void i3RdhMallocPillar(I3RdhPillar** pillar);
pillar の領域を確保する
pillar I3RdhPillar へのポインタへのポインタ

void i3RdhMallocPilCone(I3RdhPilCone** pilcone);
pilcone の領域を確保する
pilcone I3RdhPilCone へのポインタへのポインタ

void i3RdhMallocPlane(I3RdhPlane** plane,int num);
plane を num 分領域を確保する
plane I3RdhPlane へのポインタへのポインタ
num 個数

void i3RdhMallocSteel(I3RdhSteel** steel);
steel の領域を確保する
steel I3RdhSteel へのポインタへのポインタ

void i3RdhMallocBridge(I3RdhBridge** bridge);
bridge の領域を確保する
bridge I3RdhBridge へのポインタへのポインタ

void i3RdhMallocShutter(I3RdhShutter** shutter);
shutter の領域を確保する
shutter I3RdhShutter へのポインタへのポインタ

void i3RdhMallocSelPick(I3RdhSelPick** selpick);
selpick の領域を確保する
selpick I3RdhSelPick へのポインタへのポインタ

void i3RdhMallocNoriParam(I3RdhNoriParam** normalplane, int num);
normalplane を num 分、領域を確保する
normalplane I3RdhNoriParam へのポインタへのポインタ
num 個数

void i3RdhMallocRiver(I3RdhRiver** river, int num);
river を num 分、領域を確保する
river I3RdhRiver へのポインタへのポインタ
num 個数

③データ定義関数

void ChangeCmdToSndFmt(char *cmd, int *num, char ***data);
コマンド文字列をネットワーク送信フォーマットに変換する
cmd コマンド文字列
num 分割された送信文字列数

data 送信文字列へのポインタへのポインタへのポインタ

void SetAutoDemoCommand(int num, char **cmd);
オートデモコマンドをセットする
num コマンド数
cmd コマンドポインタへのポインタ

void NextAutoDemoCommand();
オートデモコマンドのカレントインデックスを１増やす

void CloseRecoveryFile();
リカバリーファイルを閉じる

void SetAutoDemoFile(char *file);
オートデモファイルをセットする
file ファイル名

void CloseAutoDemoFile()
オートデモファイルを閉じる

void i3RdhInputVec(I3RdhD3Vec* vector, int index, double value);
vectorに値をセットする
vector I3RdhD3Vec ポインタ
index 0:x 1:y 2:z
value 値

void i3RdhInputCol(I3RdhRgb* color, int index, double value);
colorに値をセットする
color I3RdhRgb ポインタ
index 0:R 1:G 2:B 3:A
value 値

void AddIPAddress(char *ip);
IPアドレスを追加する
ip IPアドレス文字列

void BackupIPAddress();
IPアドレスをバックアップする

void RestoreIPAddress();
IPアドレスをレストアする

void SetDemoMode(int mode);
デモモードをセットする
mode デモモード
  0:リカバリー
  1:リモートデモ
  2:オートデモ

void SetRemoteDemoMode(int mode);
リモートデモモードをセットする
mode TRUE/FALSE

void SetAutoDemoMode(int mode);
オートデモモードをセットする
mode TRUE/FALSE

void RDHPutCommandOne(int mode, char *cmd);
コマンドを送る
mode デモモード
GetdemoMode()の値
cmd コマンド文字列

④データ取得関数
int GetCurrentAutoDemoCommand(int *num, char *cmd);
カレントのオートデモコマンドを取得する
num コマンド数
cmd コマンド
返り値 インデックス

char *GetAutoDemoFile();
オートデモファイル名を取得する

返り値 ファイル名

double i3RdhOutputVec(I3RdhD3Vec *vector, int index);
vector の値を取得する
vector I3RdhD3Vec ポインタ
index 0:x 1:y 2:z
返り値 値

double i3RdhOutputCol(I3RdhRgb *color, int index);
color の値を取得する
color I3RdhRgb ポインタ
index 0:R 1:G 2:B 3:A
返り値 値

void GetIPAddress(
  int *num, char ip[K_NET_CONNECT_MAX][K_NET_IP_MAX_LEN]);
IP アドレスを取得する
num 数
ip[K_NET_CONNECT_MAX][K_NET_IP_MAX_LEN]
 IP アドレス文字列配列

int GetDemoMode();
デモモードを取得する
返り値 デモモード
  0:リカバリー
  1:リモートデモ
  2:オートデモ

int GetRemoteDemoMode();
リモートデモモードを取得する
返り値 TRUE/FALSE

int GetAutoDemoMode();
オートデモモードを取得する
返り値 TRUE/FALSE

⑤データ削除関数

void FreeAutoDemoCommand();
オートデモコマンドを解放する

void DeleteRecovery File();
リカバリーファイルを削除する

void DeleteAutoDemoFile();
オートデモファイルを削除する

void i3RdhFreeNVec(I3RdhR2Tensor* tensor);
tensor の領域を解放する
tensor I3RdhR2Tensor ポインタ

void i3RdhFreeHaichi(I3RdhHaichiParam** haichiparameter);
haichiparam の領域を解放する
haichiparameter haichiparam ポインタへのポインタ

void i3RdhFreeTexture(I3RdhTexture** texture);
texture の領域を解放する
texture I3RdhTexture ポインタへのポインタ

void i3RdhFreeMaterial(I3RdhMaterial** material);
material の領域を解放する
material I3RdhMaterial へのポインタへのポインタ

void i3RdhFreeLight(I3RdhLight** light);
light の領域を解放する
light I3RdhLight のポインタへのポインタ

void i3RdhFreeCameraPos(I3RdhCameraPos** abstract);
abstract の領域を解放する
abstract 個数

void i3RdhFreeCube(I3RdhCube** cube);
cube の領域を解放する

cube I3RdhCube へのポインタへのポインタ

void i3RdhFreeBall(I3RdhBall** ball);
ball の領域を解放する
ball I3RdhBall へのポインタへのポインタ

void i3RdhFreeColumn(I3RdhColumn** column);
column の領域を解放する
column I3RdhColumn へのポインタへのポインタ

void i3RdhFreeCone(I3RdhCone** cone);
cone の領域分を解放する
cone I3RdhCone へのポインタへのポインタ

void i3RdhFreePillar(I3RdhPillar** pillar);
pillar の領域を解放する
pillar I3RdhPillar へのポインタへのポインタ

void i3RdhFreePilCone(I3RdhPilCone** pilcone);
pilcone の領域を解放する
pilcone I3RdhPilCone へのポインタへのポインタ

void i3RdhFreePlane(I3RdhPlane** plane);
plane の領域を解放する
plane 個数

void i3RdhFreeSteel(I3RdhSteel** steel);
steel の領域を解放する
steel I3RdhSteel へのポインタへのポインタ

void i3RdhFreeBridge(I3RdhBridge** bridge);
bridge の領域を解放する
bridge I3RdhBridge へのポインタへのポインタ

void i3RdhFreeShutter(I3RdhShutter** shutter);
shutter の領域を解放する

shutter I3RdhShutter へのポインタへのポインタ

void i3RdhFreeSelPick(I3RdhSelPick** selpick);
selpick の領域を解放する
selpick I3RdhSelPick へのポインタへのポインタ

void i3RdhFreeNoriParam(I3RdhNoriParam** normalplane);
normalplane の領域を解放する
normalplane I3RdhNoriParam へのポインタへのポインタ

void i3RdhFreeRiver(I3RdhRiver** river);
river の領域を解放する
river I3RdhFreeRiver へのポインタへのポインタ

void DeleteIPAddress(char *ip);
IP アドレスを削除する
ip IP アドレス文字列

（7）マルチメディア関連

①データ読み込み関数

int MltLoadSettingFile();
ポート設定ファイルを読み込む
返り値 成功時:TRUE　エラー時:FALSE

②データ取得関数

int MltCalcImgDataSegs(int piece, int width, int height);
イメージデータを分割したときの分割数を求める
piece イメージ数
width イメージ幅
height イメージの高さ
返り値 分割数

void MltGetStartEndTime(struct _timeb *start, struct _timeb *end, long *sec, long *micro_sec);
開始から終了までの時間を取得する
start 開始時刻

end 終了時刻
sec 秒
micro_sec マイクロ秒

long MltGetStartEndMicroSecTime(struct _timeb *start, struct _timeb *end);
開始から終了までの時間（マイクロ秒）を取得する
start 開始時刻
end 終了時刻
返り値 マイクロ秒

void GetClampZeroToOneFloat(float *value);
Value を 0≦≦１２クランプする
value 値（入力と出力）

void ChangeLongMicroToCharMilli(long micro, char *milli);
マイクロ秒をミリ秒の文字列に変換する
micro マイクロ秒
milli ミリ秒の文字列

int MltGetPortSettingData(char *hostip, KPortSet **ps);
ポート設定を取得する
hostip ホストＩＰアドレス文字列
ps KPortSet へのポインタへのポインタ
返り値 ＩＰアドレス数

void MltKeepAspect(int orgwidth, int orgheight, int *width, int *height);
アスペクト比を保ったサイズを取得する
orgwidth オリジナルの幅
orgheight オリジナルの高さ
width 入力:現在の幅
出力:新しい幅
height 入力:現在の高さ
出力:新しい高さ

（８）基幹部分の多言語機能

①class CMultiLang のメンバ関数

bool IsKanjiDlg();
現在選択されている言語が２バイト系かどうかを調べる。
戻り値：TRUE：２バイト系、FALSE：１バイト系

CString GetLanguage();
戻り値：現在使用されている言語を表す２文字のコードを返す。

void MultiLangMenuConv_RC(CMenu *pMenu, CString IDname, int depth, int popupcounter);
メニューの言語を変換する。
pMenu　変換するメニューのアドレス
IDname　変換するメニューの ID（文字列）　（例：”ID_MENU_POPUP”）
depth 変換する階層の深さ
popupcounter ポップアップカウンター

void MultiLangDialogConv(char* IDNAME, CDialog* dlg);
IDNAME ダイアログの ID（文字列で表現したもの）　（例：”IDD_DLG_HAI_SEL”）
dlg ダイアログ・ハンドラのクラス　（例：this）

char* FindReplaceText(char* text);
kanji.txt で定義された文字列を取得する。
text 文字列のキーとなるコード（文字列）　（例：”KANJI_147”）
戻り値：変換された文字列

bool IsLang(CString lang);
現在選択されている言語かどうか調べる
lang 検証したい言語のコード（２文字の文字列）
戻り値：TRUE：一致／FALSE:不一致

CFont* FontManager(long size, CString name);
フォントをロードし使えるようにする。size を 0 とした場合には、全てのロード済みのフォントを解放する(終了処理)。
name 選択するフォント名
size フォントのサイズ
戻り値：ロードした CFont のアドレス

②それ以外の関数

bool teLang();
現在選択されている言語が２バイト系かどうか検査する。
戻り値　TRUE：２バイト系／FALSE：１バイト系

char *LangConvert(char* code);
文字列を現在選択されている言語に変換する
code 変換テーブルのＩＤコード（文字列）
戻り値：変換結果
int IsMultLangOK();
多言語機能が利用可能かどうかを打診する。
戻り値：０：不可　１：可

bool PembantuX(char* filename);
現在選択されている言語でヘルプを開く。
filename　ヘルプ・ファイル名称。
　xxxx.txt というファイル名が指定された場合、これを、xxxx.yy.txt（yy は言語コード）というファイル名に変更した上で、言語に対応するディレクトリに存在するこのファイルを表示する。
戻り値：０（成功した場合)、ERROR_CODE（失敗した場合）

（９）外部関数の多言語機能

（以下は、（１）の関数とは同名であっても、別の実体である。LocalLang.h、LocalLang.cpp で定義され、locallang.lib により、各外部関数のビルドに対して提供される）
int IsKanjiDlg(char*fn);
選択されている言語が２バイト系かどうか打診する。
fn 外部関数の名称　（例：”cone”）
戻り値：TRUE（２バイト系）／FALSE（１バイト系）

int IsMultLangOK();
多言語機能が利用可能かどうかを打診する。
戻り値 TRUE（使用可）／FALSE（使用不可）

void MultiLangDialogConv(char* IDNAME, CDialog *dlg);
IDNAME ダイアログの ID（文字列で表現したもの）　（例：”IDD_CONE”）

dlg ダイアログ・ハンドラのクラス　　(例：this)

void MultiLangMenuConv(CMenu* pMenu, CString IDname, int depth, int popupcounter);
メニューの言語を変換する。
pMenu　変換するメニューのアドレス
IDname　変換するメニューの ID（文字列）　　(例：”ID_MENU_POPUP”)
depth 変換する階層の深さ
popupcounter ポップアップカウンター

void MultiLangEnd();
終了処理を行う。

CFont* FontManager(long size, CString name);
フォントをロードし使えるようにする。size を 0 とした場合には、全てのロード済みのフォントを解放する(終了処理)。
name 選択するフォント名
size フォントのサイズ
戻り値：ロードした CFont のアドレス

void Bantu()
選択されている言語のためのディレクトリに格納された関数名.言語.txt　というヘルプ・ファイルを開く。

CString Kanji(char *ID);
選択されている言語のリテラル定数を取得する。主にプラグインＤＬＬで使用する。
ID　キー（文字列）　(例：”Kanji_5”)

name 選択するフォント名
size フォントのサイズ
戻り値：ロードしたCFontのアドレス

void Bantu()
選択されている言語のためのディレクトリに格納された関数名.言語.txt　というヘルプ・ファイルを開く。

CString Kanji(char *ID);
選択されている言語のリテラル定数を取得する。主にプラグインＤＬＬで使用する。
ID　キー（文字列）　（例：”Kanji_5”）

（以上）

付録C　参考文献一覧

（1）バックグラウンドとなる研究

1.小林英之「住環境形成シミュレーション」日本建築学会・第 16 回情報・システム・利用・技術シンポジウム論文集(1993.12)

2.小林英之・狩野勝重「依怙都市・二本松」調査報告

(1)　序論 日本建築学会大会梗概 1992.8

(2)　字別に見たフローとストック 1993.9

3.小林英之「歴史的観点からみた住宅のライフサイクル」

あらか 12、建設省建築研究所 1994.10

（2）システムの研究開発について直接紹介したもの

4.小林英之・丹羽薫「景観シミュレータ・景観データベースの研究開発について」(JACIC 情報 No.34,日本建設情報総合センター1994.4)

5.Hideyuki Kobayashi : GIS and Landscape Simulation (One-day workshop on historical heritage, ministry of Public Works, Indonesia, 1996.5 Yogyakarta, Indonesia

6.小林英之「景観シミュレータ」(公共建築 36/3 No.141;公共建築協会 1994.7)

7.小林英之「景観シミュレータの研究開発」(測量 Vol.45,No.5, 日本測量協会 1995.5)

8.小林英之「３次元 CG による土木建築施設のための景観検討システムープロトタイプ版(Ver.1.0)ー」(建設省建築研究所・建築研究資料 No.85,1995.9)

9.小林英之ほか「景観デザインにおけるシミュレーション・評価・プレゼンテーションの活用とその実際」(工業技術会,1996.5)

10.小林英之「景観シミュレータができるまで」(ランドスケープ・デザイン,マルモ出版 1997.6)

11.小林英之「建設省版景観シミュレータ・操作自習の手引き(Ver.2.03)」(建設省建築研究所・建築研究資料 No.92,1997.11 絶版、但しその内容の殆どは文献 16, 17 に含まれている)

12.小林英之「景観シミュレータで見る地域の将来像」あらか１５、建設省建築研究所 1997.11

13.小林英之「建築與都市發展電腦視覺模擬(Computer Graphic Simulation for Building and Urban Development)」中日工程技術検討會 建築研究組 論文集(1997.11.4 繁字体中文)

14.일본건설성 건축연구소 제 6 연구부 도시개발엔구실
고바야시 히데유끼 "경관시뮬레이터기술의 지역개발에
의 응용" (농어촌진흥공사 농어촌연구원 1998.8)

15.Hideyuki KOBAYASHI, "Urban Simulation Technologies for Planning – from synchronic to diachronic-", Proceedings of International Symposium on City Planning

1999.9, Tainan, Taiwan.
16.小林英之「成熟都市シミュレータ Ver.1.0＋景観シミュレータ Ver.2.05 実務マニュアル」建設省建築研究所・建築研究資料 No.96,2000.7
17.小林英之「まちづくりのためのコミュニケーションシステムの開発」国総研アニュアルレポート No.1,2002.3
18.Hideyuki KOBAYASHI “Development of Communication System for Town Planning”, Annual Report of NILIM 2002
19.小林英之「まちづくりのためのコミュニケーションシステムの開発」平成１４年度国土交通省国土技術研究会　自由課題（論文集,No.47, pp.185-188）
20.小林英之「まちづくりのためのコミュニケーション・システムの開発」土木研究センター、土木技術資料 Vol.45 No.3 2003.3 表紙及びグラビア
21.小林英之「まちづくり・コミュニケーション・システム　操作・運用マニュアル」国土技術政策総合研究所資料 No.134, 2003.9 (290p)
22.国土技術政策総合研究所高度情報化研究センター・環境研究部緑化生態研究室「国土交通省版・景観シミュレータの活用実績と、今後の応用」平成 15 年度国土技術研究会ポスター 2003.11
23. Hideyuki KOBAYASHI,”Development of Communication System for Town Planning”, International Conference on Construction Information Technology(Incite 2004):World IT for Design & Construction, Langkawi, Malaysia:18-21 February 2004(8p)
24. Hideyuki Kobayashi “Configuration Process of Landscape in Japanese Settlements –Regional Context, Historical Background and Future Scope- Proceedings of International Symposium on City Planning 2004, City Planning Institute of Japan, 2004.9
25. 小林英之「電子納品データ（SXF）と、５ｍメッシュ数値地図を用いた景観検討用データの生成」情報・システム・利用・技術シンポジウム論文集 No.27 2004.12, pp.245-249
26. 小林英之・小栗ひとみ「まちづくりのための景観シミュレーションの活用」日本造園学会造園技術報告集 2005, No.3 pp.138-141
27. 小林英之「まちづくり・コミュニケーション・システム－ネットワークを用いた景観シミュレーション－」測量 Vol.55 No.5 pp.37-40 2005.5
28. Hideyuki Kobayashi “Analysis of Satellite Images for Measuring Urban Green Coverage Ratio in Bandung and Cirebon – As Basis for Planning Future Urban Form regarding Climate Change” Second International Symposium on Sustainable Humanosphere 2007.7, Bandung, Indonesia
29. Hideyuki Kobayashi “Monitoring CO2 Emission in Indonesian Planned Housing Complexes and Designing Alternative Future Images”, Technical Note of Nilim No.440, 2008.3 (56p)

30 小林英之「データ活用による景観シミュレーションの試み」平成 20 年度国土技術研究会ポスター 2008.10
31. 小林英之「データ活用による景観シミュレーションの試み」土木研究センター、土木技術資料 2009 No.2 pp.22-27, 2009.2
32. 小林英之「情報が生産方式を変える　地域づくりに ICT 活用－景観シミュレーションの展開－」日刊建設工業新聞 2009.3.27
33. 小林英之「国土交通省版・景観シミュレータのヒストリー(1993-2009)－ソフトウェアの完成を目指して」情報・システム・利用・技術シンポジウム論文集 、日本建築学会 2009.12.3, pp.111-114

（３）OpenGL について

34.OpenGL Architecture Review Board
"OpenGL Programming Guide(日本語版)"
アジソン・ウェスレイ(1993.12)

（４）写真の解析（画像視点抽出）について

35.高木幹雄・下田陽久監修「画像解析ハンドブック」
東京大学出版会(1991.1)

（５）アルゴリスムについて

36.奥村晴彦「アルゴリズム事典」技術評論社(1991.2)

（６）電子納品データについて（三次元形式）

37.金澤文彦ほか「道路中心線形データ交換標準（案）基本道路中心線形編 Ver.2.0」
国土技術政策総合研究所資料 371, 2007.1

（７）画像およびベクトルの各種外部ファイル形式について

38. James D Murray et-al. "Encyclopedia of Graphics File Formats for PC, Macintosh, and Unix Platforms", O'Reilly & Associates, Inc.1994

（８）国総研ホームページ等からの最新情報の取得

本稿に記載された内容、CD-ROM に収録されたインストーラ、サンプル・データ等に加え、システムのソース・コードその他の情報を、国土技術政策総合研究所のホームページからダウンロードすることができます。

39. 国土技術政策総合研究所、まちづくり・コミュニケーションに関する WEB サイト
http://sim.nilim.go.jp/MCS

その他の関連サイトを以下に紹介します。

40.インドネシア公共事業省研究開発総局人間居住研究所
http://puskim.pu.go.id

41.大韓民国農漁村公社農漁村研究院
http://rri.ekr.or.kr

付録D　プログラマ一覧

関塚 亨（IP、G3DRL ライブラリ）
加納 裕（DML ライブラリ）
高橋 雅史（メイン画面）
塩田 剛志（多くのダイアログ）
高芝 克彰（画像視点抽出）
大峯 美智代（SML,ENV、DBIL ライブラリ）
安元 正博（Z3 ライブラリ）
加藤 久宜（U3 ライブラリ）
朱 慧敏（二次元画像処理）
任 昌榮（日韓共同研究、WEB 関連）
高旗 進（情報配信、デバッグ情報交換用サーバー）
中島 寛（多言語機能）
小林 英之（図形演算、全体とりまとめ）

付録 E　CD-ROM の構成

付録 CD-ROM は、報告の原稿、ソースコード、及びセットアップ一式等を含む。

１．報告の原稿(doc)

３１のファイルに分かれている。

２．ライブラリ関数一覧（html）

解説の付録 B を、html 形式で記述した資料である。関数の検索などに使用する。

３．ソースコード(program)

ソースコードは、VS2005 上のビルドとして構成している。

４．セットアップ作成用ソースコード一式(setup)

セットアップ用ディスクを作成するためのプログラム、バッチファイル等である。

209 ディレクトリに、セットアップすべきファイル群を用意した上で、作成する。

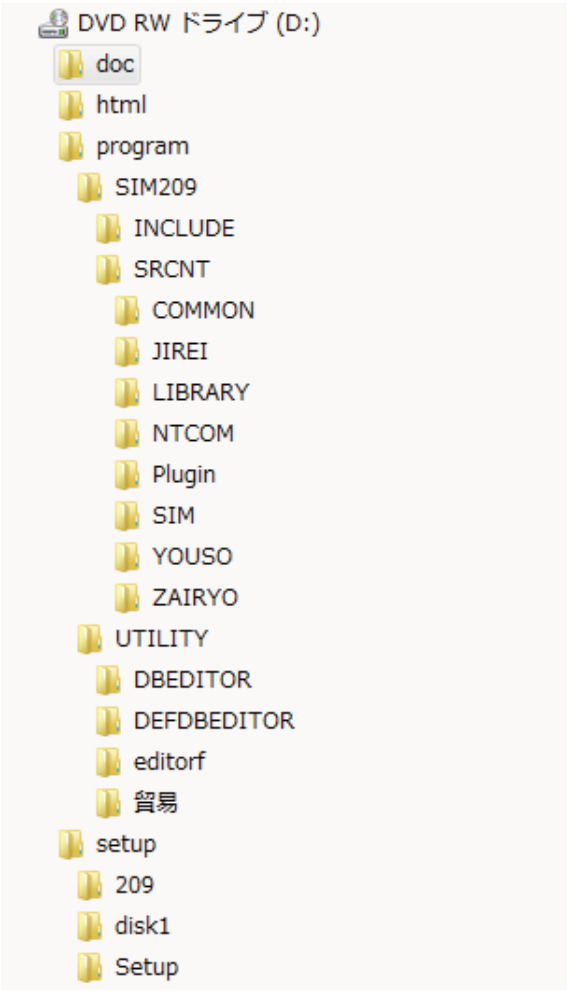
５．セットアップ一式(setup/disk1/setup.exe 以下)

デバッグ、改良した実行形式をテストするための実行環境を構築する。

６．サンプル・データ(セットアップ後の@keikan/kdb/@samples)

セットアップ一式に含まれている。セットアップを実行すると、利用できる。

CD-ROM のディレクトリ構成図



国土技術政策総合研究所報告第 42 号
平成２３年３月１４日
REPORT OF NILIM
No.42 March 14, 2011

本報告の転載・複写のお問い合わせは
〒305-0804 茨城県つくば市旭１番地
企画部研究評価・推進課　TEL 029-864-2675