# 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
                        IsKanjiDlg("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){
                        IsKanjiDlg("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。