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);
形綱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形鋼
※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”)