第2009-02号

OpenGLを用いた3次元グラフィックスアプリケーションの研究開発

関西大学　総合情報学部　関西大学先端科学技術推進機構
教授　田中　成典

平成22年11月

助成研究者紹介

たなか しげのり

田中 成典

現職 ： 関西大学総合情報学部 教授(博士(工学))

主な著書・論文:

【3 次元関連】

- デジカメ活用によるデジタル測量入門（森北出版，平成 12 年）.
- DirectX8 & VC++ －3Dの基礎とゲームの作り方－（工学社，平成 14 年）
- 建設業界のための 3 次元情報（山海堂，平成 18 年）
- できる！使える！バーチャルリアリティ（建通新聞社，平成 18 年）
- 基礎からわかる画像処理（工学社，平成 20 年）

【建設 CALS 関連】

- 建設 CALS/EC に向けた電子国土の動向を探る－CAD/CG/GIS/GPS の統合－（山海堂，平成 13 年）
- e-Japan 電子政府の実現に向けて建設業界のための XML（工学社，平成 14 年）
- e-Japan 電子政府の実現に向けて建設業界のためのデータモデル（工学社，平成 15 年）
- e-Japan 電子政府の実現に向けて建設情報の利活用（工学社，平成 16 年）
- 基礎からわかる GIS（森北出版，平成 17 年）
- Logical Smart for SXF Ver.2.0（建通新聞社，平成 17 年）

【査読論文】

- ステレオビデオカメラを用いた交通量算出システムに関する研究開発（情報処理学会論文誌，平成 18 年）
- SXF の同一性判別コンポートネントの実装研究（情報処理学会論文誌，平成 19 年）
- 特徴点追跡による 3D モデルの自動生成に関する研究（日本知能情報ファジィ学会誌，平成 19 年）
- SXF Ver.3.0 対応の同一性判別システムの展開研究（情報処理学会論文誌 データベース，平成 20 年）
- 建設業界のための SXF ビューアの開発（情報処理学会論文誌 データベース，平成 20 年）
- 高速道路事業におけるプロダクトモデルの研究開発（情報処理学会論文集，平成 20 年）
- 動画像による個人識別技術を用いた勤怠管理に関する研究（映像情報メディア学会誌，平成 21 年）

共同研究者紹介

しばさき りょうすけ

柴崎 亮介

現職 ： 東京大学空間情報科学研究センター センター長・教授(工学博士)

主な著書・論文:

- GIS 入門（日本測量協会，平成 4 年）
- 多様な観測データや知識を用いた地物の時空間変化の再構成手法（地理情報システム学会 GIS-理論と応用，平成 15 年）
- 三次元 GIS の基礎技術（写真測量とリモートセンシング，平成 16 年）
- Simulation-based Estimation of Multipath Mitigation Using 3D-GIS and Spatial Statistics (Proceedings of ION GNSS，平成 18 年)
- 建設分野における地理空間情報基盤の構築に向けた地名辞典に関する研究（土木情報利用技術論文集，平成 19 年）
- 工事完成図を利用した GIS データの整備を支援する CAD-GIS 連携の手引き書の作成(地理情報システム学会講演論文集，平成 19 年)

きたがわ えつじ

北川 悦司

現職 ： 阪南大学経営情報学部 准教授(博士(情報学))

主な著書･論文:

- 写真測量技術を用いた2Dデジタル画像からの3Dモデル空間の創出に関する基礎研究(土木情報システム論文集, 平成 12 年)
- 2D デジタル画像を用いた Web/3D モデルハウスの構築に関する研究（土木情報利用技術論文集, 平成 15 年）
- デジタルビデオカメラを用いた 3 次元モデル自動生成システムの研究開発（土木情報利用技術論文集, 平成 16 年）
- 3 次元衛星電波経路シミュレーションに関する研究開発（土木情報利用技術論文集, 土木学会, 平成 16 年）
- 特徴点追跡による 3D モデルの自動生成に関する研究, 日本知能情報ファジィ学会論文集, 平成 19 年）

くぼた さとし

窪田 諭

現職 ： 岩手県立大学ソフトウェア情報学部 講師(博士(工学))

主な著書･論文:

- コンクリート橋における維持管理業務の To-be モデルの構築に関する研究（土木情報利用技術論文集, 平成 16 年）
- 道路管理における空間基盤データの利活用システムと運用モデル（土木情報利用技術論文集, 平成 18 年）
- 四次元情報を整備した道路マネジメントシステムの構築に関する研究（土木情報利用技術論文集, 平成 18 年）
- 空間基盤データの整備と活用における官民協働の実証研究（土木学会論文集, 平成 19 年）

ものべ かんたろう

物部 寛太郎

現職 ： 宮城大学事業構想学部 助教(博士(情報学))

主な著書･論文:

- 建設業における 3 次元情報の活用に関する文献調査（土木情報利用技術論文集, 平成 16 年）
- WWW 自動探索による電子地図の属性情報自動抽出システムの研究開発（土木情報利用技術論文集, 平成 16 年）
- SXF の同一性判別コンポーネントの実装研究（情報処理学会論文誌, 平成 19 年）
- SXF Ver.3.0 対応の同一性判別システムの展開研究（情報処理学会論文誌, 平成 20 年）
- 高速道路事業におけるプロダクトモデルの研究開発（情報処理学会論文誌, 平成 20 年）

なかむら けんじ

中村 健二

現職 ： 立命館大学情報理工学部 助手(博士(情報学))

主な著書･論文:

- Web リンク構造解析と自然言語処理による組織関係の抽出についての研究（情報処理学会論文誌, 平成 18 年）
- カテゴリ分類と時系列情報に基づくブログスパム判定手法の提案（情報処理学会論文誌, 平成 20 年）
- GA を用いた Web ニュースの時系列情報を考慮したトピック抽出に関する研究(情報処理学会論文誌, 平成 20 年)
- セキュアライフの創出を目指した安全知の獲得に関する研究－電子掲示板からの犯行予告の抽出－(土木情報利用技術論文集, 平成 21 年)

目次

1　序論

1.1　研究の背景

我が国の建設業界においては，CAD データ交換標準フォーマットである SXF の開発が行われ，各社の商用製品への実装や，国土交通省直轄事業の電子納品に使用される等，2 次元の CAD が主流となっている. 一方, 国土交通省 CALS/EC アクションプログラム 2005 では，3 次元 CAD の利活用に関する構想が提案されている．しかし，現在，利用されている 3 次元 CAD エンジンは，AutoCAD や SolidWorks をはじめ，外国の 3 次元 CAD が大半であり，国産は皆無である．安価な 3 次元 CAD エンジンが存在しないため，建設事業の設計・施工フェーズで 3 次元 CAD データが利活用されていない．そのため，国産の安価な 3 次元 CAD エンジンの早急な開発が切望されている．

1.2　研究の目的

国土交通省 CALS/EC アクションプログラム 2005 では，3 次元 CAD の利活用に関する構想が提案されている．しかし，現状では，安価な 3 次元 CAD エンジンが存在しないため，建設事業の設計・施工フェーズで 3 次元 CAD データが利活用されていない．そのため，国産の安価な 3 次元 CAD エンジンの早急な開発が切望されている．この背景を受けて，筆者らは，「平成 20 年度（財）日本建設情報総合センター研究助成（指定研究）」において, OpenGL によるグラフィックスアプリケーションの開発のための調査を行い，調査報告書を作成した．

本研究では，前述の先行研究で作成した調査報告書を基に，実用化に向けての試験研究として，OpenGL を用いた 3 次元 CAD エンジンのパイロットシステムの開発を行う．本研究の意義としては，OpenGL を用いた 3 次元 CAD エンジン開発の実現可能性を理解・掌握できることである．また，本研究の成果は，わが国の CAD ベンダが独自の 3 次元 CAD エンジンの開発に着手する場合の参考資料となる．その結果，CAD ベンダが独自に自社のアイデアを活かした 3 次元 CAD エンジンを開発することが可能になると考えられる．

1.3　本資料の構成

本研究では，3 次元 CAD の開発時において，OpenGL が利用可能であるかどうかの把握を助けることを目的として，平成 20 年度の研究で調査した内容をもとに，3 次元グラフィックアプリケーションの開発を行った．

本章の構成については，以下の通りである．

2 章　画面仕様

平成 20 年度の研究では，市販される 3 次元 CAD エンジンを調査することで，3 次元グラフィックスアプリケーションに実装すべき機能を「平成 20 年度報告書　第 4 章　3 次元 CAD エンジンのプロトタイプの開発に向けた機能調査」として取りまとめた．そこで，本研究では，これらの各機能を実現するための必要となる画面仕様の定義を行った．

3 章　クラス設計

3 次元グラフィックスアプリケーションの実装に必要なクラス構造の定義を行った．クラスの定義においては，システム間における円滑なデータ交換を実現するために，「平成 20 年度報告書　第 5 章 OpenGL による 3 次元モデルの表現方法」で取りまとめた ISO10303 規格の Entity 定義や SXF のフィーチャ仕様を参考として設計を行った．

4 章　実装仕様

3 次元グラフィックスアプリケーションの各機能の実装仕様について定義を行った．実装仕様の定義においては，平成 20 年度の研究で取りまとめた「報告書　第 5 章　OpenGL による 3 次元モデルの表示方法」「報告書　第 6 章　OpenGL による CAD 機能の実装方法」の内容に加えて，3 次元 CAD として必須となる 2D-3D 機能（回転面やスイープ面等の作成），モデル編集機能（平行移動・回転や複写等）や CAD 機能（グループ化やレイヤ切り替え等）の実装仕様を新たに定義した．また，GUI を介したユーザとクラス構造間での情報の受け渡し方法を含めた一連の実装方法について定義した．

5 章　結論

本研究では，2 章から 4 章で示した画面仕様，クラス設計，実装仕様をもとに，3 次元グラフィックスアプリケーションの開発を行った．その成果として，OpenGL を用いた 3 次元 CAD の開発の実現可能性について取りまとめる．

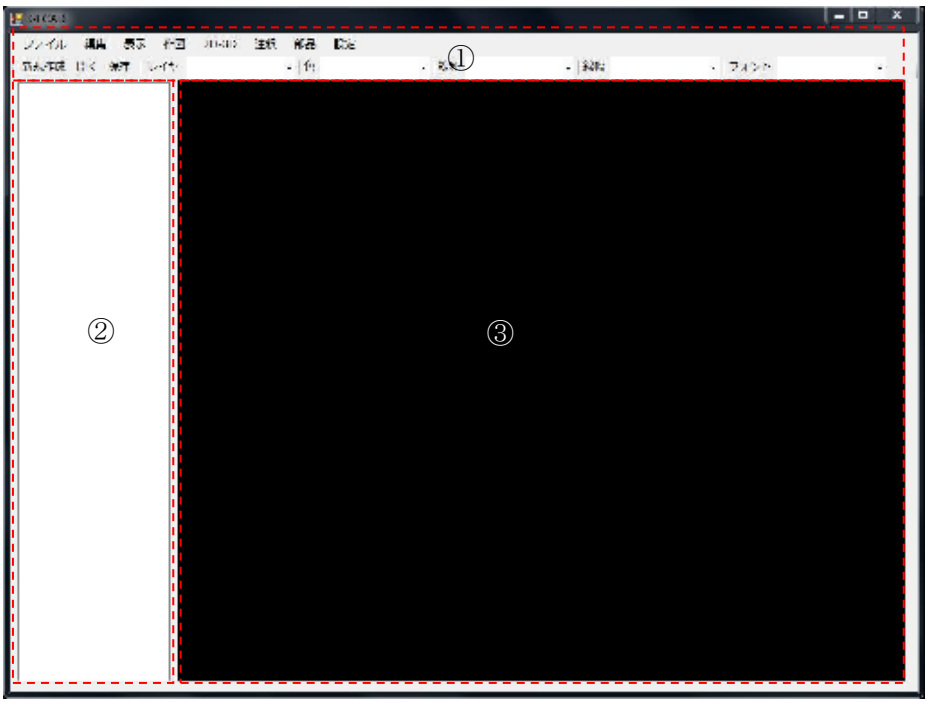
2　画面仕様

2.1　画面仕様の定義

本章では，3 次元グラフィックスアプリケーションの画面仕様について定義する．本研究で定義する画面仕様については，「平成 20 年度報告書　第 4 章　3 次元 CAD エンジンのプロトタイプの開発に向けた機能調査」で定義した機能の実現に必要な最低限のものを定義した．

2.2　メイン画面

3 次元グラフィックスアプリケーションのメイン画面を以下に示す．



メイン画面は，3 つの領域で構成される．まず，実行する操作を選択するメニューバー，ツールバー領域（①）である．つぎに，ユーザが作成した 3 次元モデルの情報を一覧表示するためのツリー領域（②）である．最後に，3 次元モデルの作成時のマウス操作，作成した 3 次元モデルを表示するためのモデル空間領域（③）である．

2.3　メニューバー・ツールバー領域

メニューバー・ツールバー領域では，3 次元グラフィックスアプリケーションで提供する機能一覧を提供する．メニューバーとツールバーで提供する機能をそれぞれ以下に示す．

2.3.1　メニューバー

メニュバーで提供する機能一覧を以下に示す．

(1) ファイル

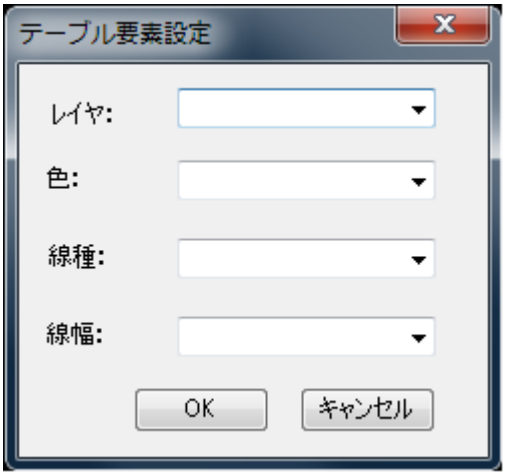
「ファイル」では，ファイルの読込みや保存などの機能を提供する．「ファイル」で提供する機能一覧を以下に示す．

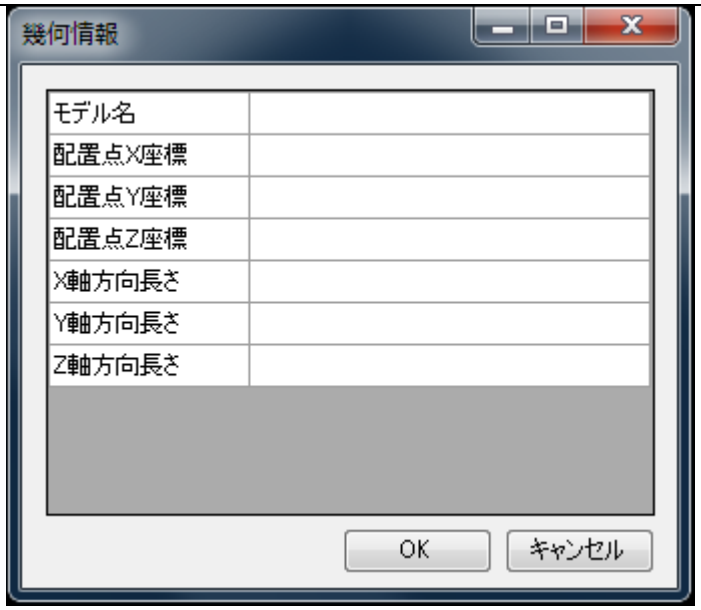
機能	説明
新規作成	作業中の全てのモデルの情報を破棄する．ただし，「新規作成」を選択すると，「情報を破棄して新規にファイルを作成しますか」というメッセージボックスを表示し「はい」を選択した場合のみ情報を破棄する．
開く	ファイルオープンダイアログを使用し，図面ファイルの情報を読込み，モデル空間上にモデルを作図する．
上書き保存	ファイル名を変えずに本システムが読込んでいる図面ファイルの情報を保存する．ただし，新規作成の場合は，保存するファイルが存在しないため，「名前を付けて保存」と同様の処理を行う．
名前を付けて保存	ファイルセーブダイアログを使用し，ファイル名を指定することで新規に図面ファイルを作成し，作成した図面ファイルに作業中の情報を保存する．
終了	システムを終了する．ただし，「終了」を選択すると，「システムを終了しますか」というメッセージボックスを表示し，「はい」を選択した場合のみシステムを終了する．

(2) 編集

「編集」では，作図したモデルの形状，色や位置などの情報を変更するための機能を提供する．「編集」で提供する機能一覧を以下に示す．

機能	説明
削除	モデル空間でマウス操作にて選択したモデルを削除する．
移動	モデル空間上において，マウス操作で選択したモデルに対して「平行移動」，「回転」，「尺度変更」を行う． ・平行移動 モデル空間上において，マウス操作で平行移動の対象となるモデルと平行移動後の配置座標を指定することで，モデルを平行移動する． ・回転 回転軸の設定ダイアログで回転軸を指定し，モデル空間上において，マウス

	操作で回転の対象となるモデルと回転角度を指定することで, モデルを回転する．回転軸の設定ダイアログを次に示す.  回転軸の設定ダイアログでは，モデルを回転する軸を「X 軸」,「Y 軸」,「Z 軸」の中から選択する．回転軸を選択して「OK」ボタンを押すと，ダイアログを閉じ，モデル空間上において，選択した回転軸を基準にマウス操作でモデルを回転できる．また，「キャンセル」ボタンを押した場合は，回転自体を取り消す. ・尺度変更 「尺度変更」では，モデル空間上において，マウス操作で尺度変更の対象となるモデルと尺度を指定することで，モデルを尺度変更する.
複写	モデル空間上において，マウス操作で選択したモデルの情報を引き継ぎ，「平行移動」,「回転」,「尺度変更」を行い，モデルを作図する.
テーブル要素	モデル空間上において，マウス操作で選択したモデルのレイヤ，色，線種，線幅の情報を変更する．テーブル要素設定ダイアログを次に示す.  テーブル要素設定ダイアログでは，テーブル要素をダイアログ上で設定し，「OK」ボタンを押すことで，モデルを編集する．また，「キャンセル」ボタンを押した場合，ダイアログを閉じ，テーブル要素の設定自体を取り消す.
幾何要素	モデル空間上において, マウス操作で選択したモデルの幾何情報を幾何情報ダイアログで編集する．例として，直方体を選択した場合に表示される幾何情報ダイアログを次に示す.

	 幾何情報ダイアログでは，各モデルの幾何情報を表示する．値を変更して「OK」ボタンを押すと，ダイアログを閉じ，変更した内容を基にモデルを再度作図する．また，「キャンセル」ボタンを押した場合は，幾何情報の編集自体を取り消す．

(3) 表示

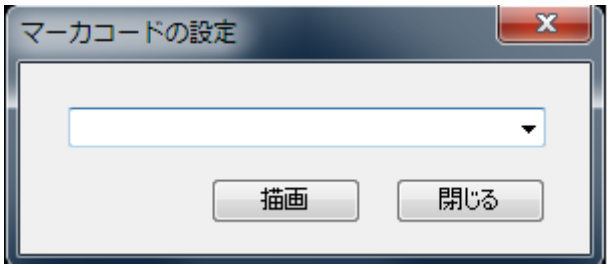
「表示」では，カメラの位置やモデルの表現方法を変更するための機能を提供する．「表示」で提供する機能一覧を以下に示す．

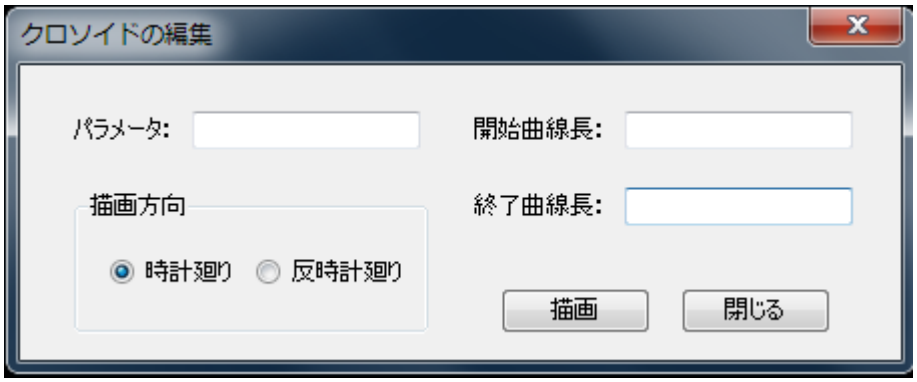
機能	説明
ズームイン	モデルを拡大表示するため，注視点に対してカメラを近づける．本システムでは，マウス操作（中ボタンを上に回転）を行うことでもズームインを可能にする．
ズームアウト	モデルを縮小表示するため，注視点からカメラを遠ざける．本システムでは，マウス操作（中ボタンを下に回転）を行うことでもズームアウトを可能にする．
射影切替	カメラの表示方法を投影方法（「平行投影」か「透視投影」）のどちらかに切り替える．
視点切替	モデルに対するカメラの位置（正面図，上面図，下面図，側面図，背面図，デフォルトの視点に戻す）を設定する．本システムでは，マウス操作（軸回転：右ボタン，平行移動：左ボタン）を行うことでもカメラの位置の切り替えを可能にする．
形状切替	選択したモデルの形状をサーフェスモデルやワイヤーフレームモデルに切り替える．

(4) 作図

「作図」では，モデル空間上に点，曲線，面要素を作図するための機能を提供する．「作

図」で提供する機能一覧を以下に示す.

機能	説明
点マーカ	モデル空間上において，マウス操作で配置点を指定し，マーカコードの設定ダイアログでマーカコードを指定することで点マーカを作図する. マーカコードの設定ダイアログを次に示す.  マーカコードの設定ダイアログでは，コンボボックスに設定されたマーカコードを選択し，「描画」ボタンを押すと，指定したマーカコードで点マーカを作図する．また，「閉じる」ボタンを押した場合は，点マーカの作図自体を取り消す.
線分	モデル空間上において，マウス操作で始点と終点を指定することで線分を作図する.
折線	モデル空間上において，マウス操作で複数の頂点を順に指定することで折線を作図する.
曲線	曲線では，「円」，「楕円」，「円弧」，「楕円弧」，「ベジェ曲線」，「NURBS 曲線」，「クロソイド」，「複合曲線」を実装する. ・円 モデル空間上において，マウス操作で中心点と半径を指定することで円を作図する. ・楕円 モデル空間上において，マウス操作で中心点，長半径の端，短半径の端と回転角を指定することで楕円を作図する. ・円弧 モデル空間上において，マウス操作で中心点，始点と終点を指定することで円弧を作図する. ・楕円弧 モデル空間上において，マウス操作で中心点，長半径の端，短半径の端，回転角，始点と終点を指定することで楕円弧を作図する. ・ベジェ曲線 モデル空間上において，マウス操作で複数の制御点を順に指定することでベジェ曲線を作図する．制御点は階数に応じた個数を指定する. ・NURBS 曲線 モデル空間上において，マウス操作で複数の制御点を順に指定することでNURBS 曲線を作図する．制御点は階数に応じた個数を指定する. ・クロソイド

	モデル空間上において，マウス操作で開始曲線長，終了曲線長を指定し，クロソイドの編集ダイアログでパラメータ，描画方向を指定することでクロソイドを作図する．クロソイドの編集ダイアログを次に示す．  クロソイドの編集ダイアログでは，クロソイドのパラメータであるパラメータ，描画方向，開始曲線長，終了曲線長の値を設定する．値を設定して「描画」ボタンを押すと，ダイアログを閉じ，設定した内容を基にクロソイドを作図する．「閉じる」ボタンを押すと，ダイアログを閉じ，編集作業を終了する． ・複合曲線 モデル空間上において，マウス操作で複数の幾何要素（折線，円弧，楕円弧，ベジェ曲線）を指定することで一つの複合曲線を定義する．
直方体	モデル空間上において，マウス操作で配置点，長方形の対角と高さを指定することで直方体を作図する．
ウェッジ	モデル空間上において，マウス操作で配置点，長方形の対角と高さを指定することでウェッジを作図する．
球	モデル空間上において，マウス操作で配置点と半径を指定することで球を作図する．
円錐	モデル空間上において，マウス操作で配置点，半径，高さと上面の位置を指定することで円錐を作図する．
円柱	モデル空間上において，マウス操作で配置点，半径と高さを指定することで円柱を作図する．
トーラス	モデル空間上において，マウス操作で配置点，大半径と小半径を指定することでトーラスを作図する．
曲面	「曲面」では，「ベジェ曲面」，「NURBS 曲面」を描画する．「曲面」では，「ベジェ曲面」，「NURBS 曲面」を実装する． ・ベジェ曲面 モデル空間上において，マウス操作で複数の制御点を順に指定することでベジェ曲面を作図する．制御点は階数に応じた個数を指定する． ・NURBS 曲面 モデル空間上において，マウス操作で複数の制御点を順に指定することでNURBS 曲面を作図する．制御点は階数に応じた個数を指定する．

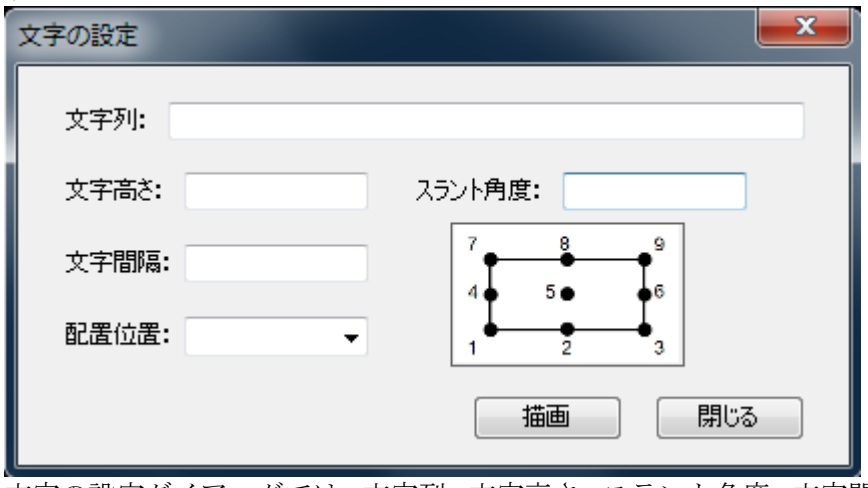
(5) 2D-3D

「2D-3D」では，曲線要素から面を生成するための機能を提供する．「2D-3D」で提供する機能一覧を以下に示す．

機能	説明
掃引面	モデル空間上において，マウス操作で掃引の対象となるモデル（線分，折線，円，円弧，楕円，楕円弧，ベジェ曲線，クロソイド，複合曲線）と掃引線となるモデル（線分，折線，ベジェ曲線）を指定することで面を生成する．
押出面	モデル空間上において，マウス操作で押出の対象となるモデル（線分，折線，円，円弧，楕円，楕円弧，ベジェ曲線，クロソイド，複合曲線）と押し出す先の点を指定することで面を生成する．
回転面	モデル空間上において，マウス操作で回転の対象となるモデル（線分，折線，円，円弧，楕円，楕円弧，ベジェ曲線，クロソイド，複合曲線）と回転面の半径を指定することで面を生成する．

(6) 注釈

「注釈」では，モデルに直線寸法，半径寸法，引出線などの注釈を挿入するための機能を提供する．「注釈」で提供する機能一覧を以下に示す．

機能	説明
文字	モデル空間上において，マウス操作で配置点を指定し，文字の設定ダイアログで文字列，文字高さ，文字間隔，配置位置とスラント角度を指定することでモデル空間上に文字を挿入する．文字の設定ダイアログを次に示す．  文字の設定ダイアログでは，文字列，文字高さ，スラント角度，文字間隔，配置位置を設定する．そして，「描画」ボタンを選択することで，設定した情報を基に文字を挿入する．また，「閉じる」ボタンを押した場合，文字の挿入自体を取り消す．
直線寸法	モデルに直線寸法を挿入する．直線寸法は，補助線の有無によって作成方法が異なる．補助線が有る場合は，モデル空間上において，マウス操作で

	直線寸法を挿入するモデル，直線寸法を挿入するために関連付ける 2 つの頂点と寸法線の位置を指定することでモデルに直線寸法を挿入する．補助線が無い場合は，モデル空間上において，マウス操作で直線寸法を挿入するモデルと直線寸法を挿入するために関連付ける 2 つの頂点を指定することでモデルに直線寸法を挿入する．
直径寸法	モデル空間上において，マウス操作で直径寸法を挿入するモデル（円）と直径寸法を挿入するために関連付ける頂点を指定することでモデルに直径寸法を挿入する．
半径寸法	モデル空間上において，マウス操作で半径寸法を挿入するモデル（円，円弧）と半径寸法を挿入するために関連付ける頂点を指定することでモデルに半径寸法を挿入する．
角度寸法	モデル空間上において，マウス操作で角度寸法を挿入するモデル（円，円弧）と角度寸法を挿入するために関連付ける 2 つの頂点を指定することでモデルに角度寸法を挿入する．
弧長寸法	モデル空間上において，マウス操作で弧長寸法を挿入するモデル（円，円弧）と弧長寸法を挿入するために関連付ける 2 つの頂点を指定することでモデルに弧長寸法を挿入する．
引出線	モデル空間上において，マウス操作で複数の頂点，文字列，文字高さ，文字間隔，配置位置とスラント角度を指定することでモデル空間上に引出線を挿入する．
バルーン	モデル空間上において，マウス操作で複数の頂点，文字列，文字高さ，文字間隔とスラント角度を指定することでモデル空間上にバルーンを挿入する．

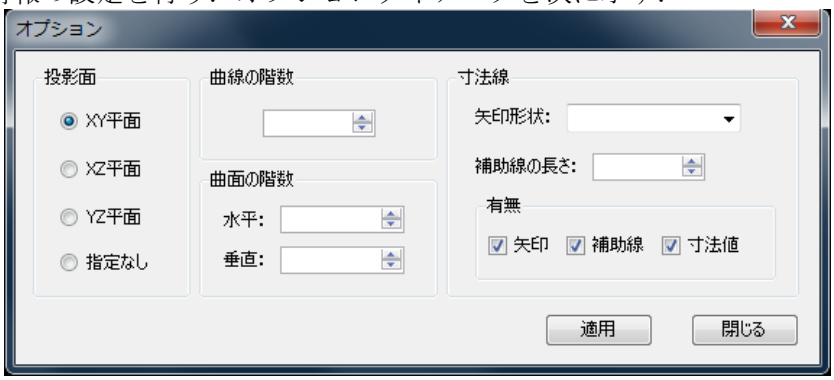
(7) 部品

「部品」では，モデルや部品間の関連付け，登録，挿入を行うための機能を提供する．「部品」で提供する機能一覧を以下に示す．

機能	説明
関連付け	基準となるモデルと関連付け対象のモデルを指定することで複合図形を作図する．
登録	モデル空間上において，マウス操作で登録する複合図形を指定し，ファイルセーブダイアログで保存場所とファイル名を指定することで複合図形の情報を保存する．複合図形の保存については，ファイルフォーマットの複合図形の形式で保存する．
挿入	ファイルオープンダイアログで挿入する複合図形のファイル名を指定し，モデル空間上において，マウス操作で配置点とすることでファイルの情報を読み込み，モデル空間上にモデルを作図する．

(8) 設定

「設定」では，レイヤ管理と各種の情報を設定するための機能を提供する．「設定」で提供する機能一覧を以下に示す．

機能	説明
レイヤ管理	レイヤ管理ダイアログ上でレイヤの表示切替，追加，削除などを行う．レイヤ管理ダイアログを次に示す．  レイヤ管理ダイアログでは，登録したレイヤの一覧を表示する．左側の「表示/非表示」では，レイヤ上に描画されているモデルを表示するか表示しないかを設定する．右側の「削除」は，レイヤを削除する．ただし，レイヤが残り１つの場合は，レイヤを削除することはできない．また，「追加」は，新しいレイヤを追加する．「追加」ボタンを押すと，表の一番下に「新規レイヤ」というレイヤを追加する．レイヤ名の変更については，レイヤ名のテキスト上で変更する．最後に，「適用」ボタンを押すと，ダイアログを閉じ，設定した情報が反映する．「閉じる」ボタンを押すと，設定した情報を破棄してダイアログを閉じる．
オプション	オプションダイアログ上で，モデルを作図する平面を示す投影面，ベジェ曲線（NURBS 曲線）の階数，ベジェ曲面（NURBS 曲面）の階数，寸法線の情報の設定を行う．オプションダイアログを次に示す．  投影面では，作図したモデルを投影する平面を「XY 平面」，「XZ 平面」，「YZ 平面」と「指定なし」の中から選択する．「指定なし」を選択した場合は，最も適した投影を算出する．曲線の階数と曲面の階数でベジェ曲線（NURBS 曲線）やベジェ曲面（NURBS 曲面）の階数を指定する．ただし，

	指定可能な値は全て「1」から「10」までとする．寸法線では，矢印の形状，補助線の長さ，矢印/補助線/寸法値の有無を指定する．補助線の長さについては，指定可能な値は「1」から「100」までとする．そして，適用ボタンを押すと，それ以降，各種のモデルを描画する際に，設定した内容を基に描画を行う．閉じるボタンを押した場合，指定した内容を取り消し，元の値に戻す．

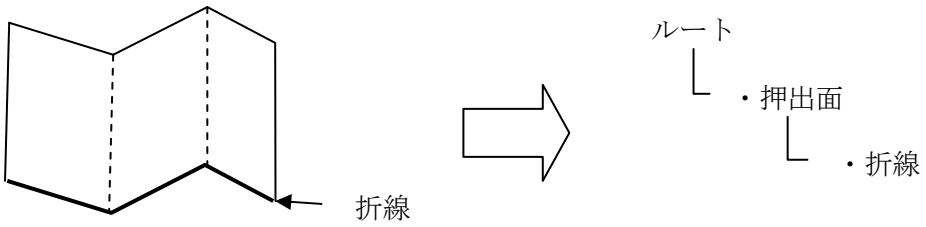
2.3.2 ツールバー

ツールバーで提供する機能一覧を以下に示す．

機能	説明
新規作成	メニューバーの「ファイル」→「新規作成」と同じ処理を行う．
開く	メニューバーの「ファイル」→「開く」と同じ処理を行う．
保存	メニューバーの「ファイル」→「保存」と同じ処理を行う．
レイヤ	メニューバーの「設定」→「レイヤ管理」と同じ処理を行う．また，レイヤのコンボボックスでは，作図するモデルを配置するレイヤを指定する．
色	コンボボックスを使用して作図するモデルの色を指定する．
線種	コンボボックスを使用して作図するモデルの線種を指定する．
線幅	コンボボックスを使用して作図するモデルの線幅を指定する．
フォント	コンボボックスを使用して作図するモデルのフォントを指定する．

2.4　ツリー領域

ツリー領域では，3 次元グラフィックスアプリケーションで作成したモデルの一覧を表示する．2D-3D 機能の押出面の作成操作によって作成されたモデルについては，以下に示すように，押出面を作成する際に使った幾何要素と作成されたモデルとの関係を階層構造で表示する．



押出面　　ツリー

2.5　モデル空間領域

モデル空間領域では，モデルの作成時におけるユーザ操作，作成したモデルの描画を行うための 3 次元のモデル空間を表示する．ユーザは，モデル空間上においてマウス操作を行うことで，作成するモデルのパラメータ値（線の場合，始点と終点の座標値）等を任意に設定することができる．また，モデル空間領域では，以下に示すマウス操作により，カメラを設定するための提供する．

機能	説明
右ドラッグ	カメラ視点の位置をドラッグ方向に平行移動する．
左ドラッグ	カメラ視点の位置をモデル空間の座標軸を基準に回転する．
ロール	カメラ視点の位置を前・後方向に移動する． ※ズームイン・ズームアウト

3　クラス構造

3.1　クラス構造の定義

本章では，3 次元グラフィックスアプリケーションで扱う情報を保持するためのクラス構造について定義する．クラス構造については，以下の 3 つに大別することができる．

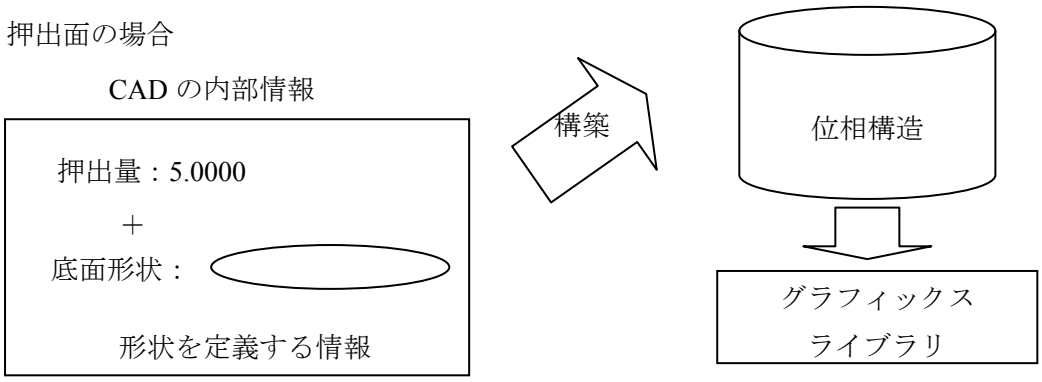
・モデル情報を保持するクラス
・注釈情報を保持するクラス
・その他のクラス

まず，モデル情報を保持するクラスとしては，3 次元グラフィックスアプリケーションで作成するモデルの形状情報を保持するための構造を定義する．次に，注記情報を保持するクラスでは，モデルに対して関連づける寸法要素や注釈情報を保持するための構造を定義する．最後に，その他のクラスとしては，レイヤや色，線種等の視覚情報に加え，3 次元グラフィックスアプリケーションを実行する上で必要な管理情報等を保持するための構造を定義する．

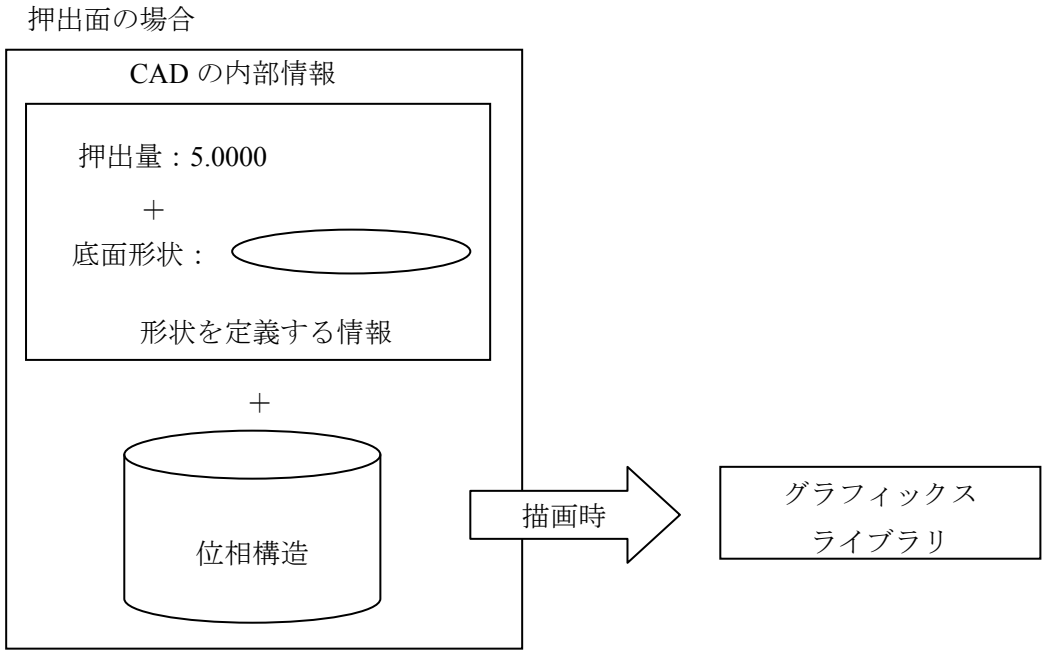
※本クラスの定義においては，Microsoft 社の VisualStudio.Net C#での開発を前提としている．

3.2　モデル情報を保持するクラス

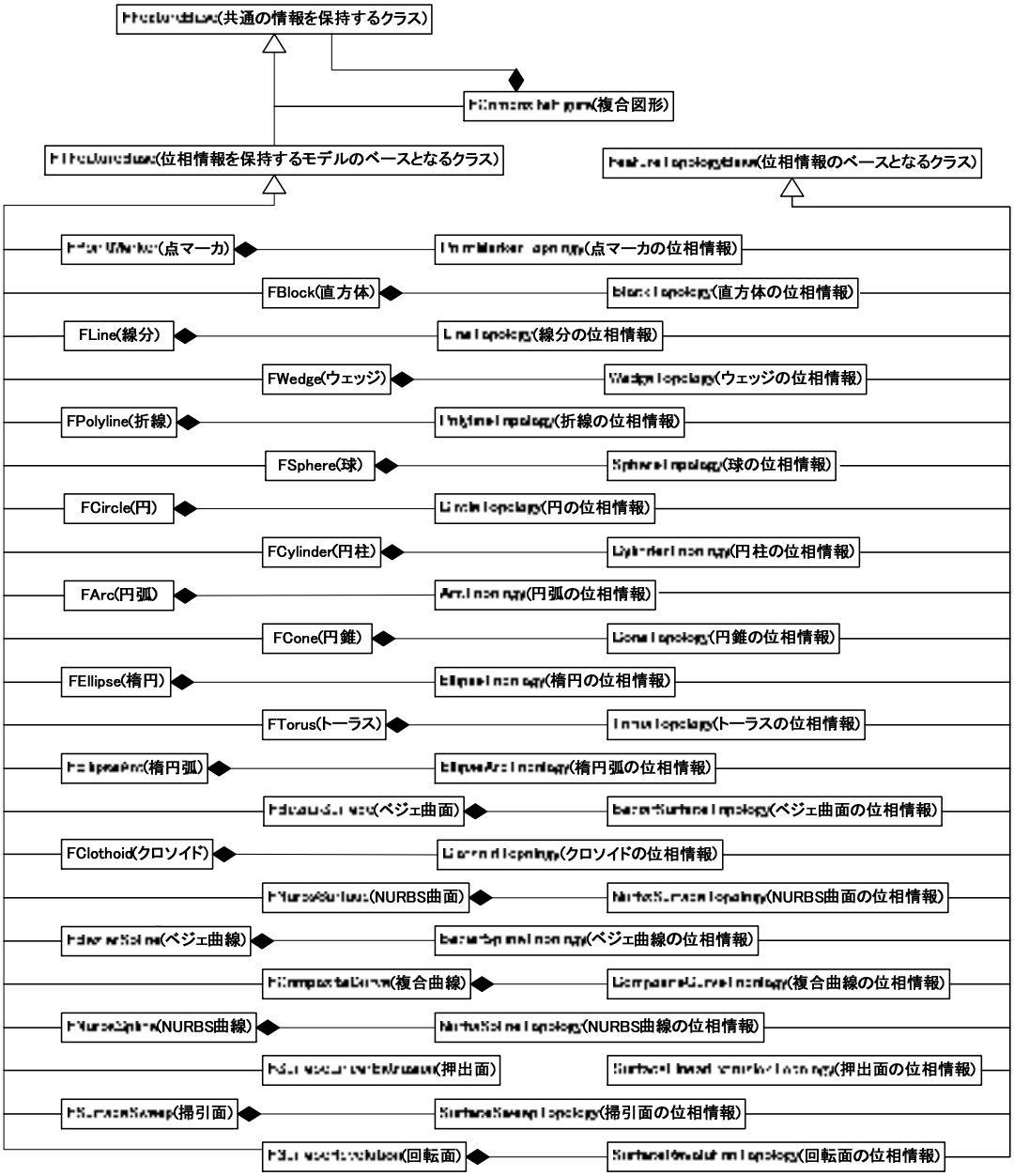
本研究では，モデル情報について，その形状を定義するための情報と，その位相構造を表現するための情報との 2 つの情報で管理する．現在，市場においては，パラメトリックモデリング機能を有した 3 次元 CAD が普及している．これらの CAD においては，モデルの形状を定義するための情報のみを内部で管理しており，また，それから位相構造を生成する機能を有している．そして，描画時においては，随時に生成した位相構造を使用することで，グラフィックスライブラリ間でのデータの受け渡しを実現する．押出面における例を以下に示す．



ただし，本研究においては，システム内部の仕組みを簡略化するために，モデル情報として，形状を定義する情報と位相構造を定義する情報の両方を内部で保持する．本研究にけるモデル情報の概要を以下に示す.



本研究で定義したクラス構造を以下に示す.



FFeatureBase クラスは，各モデルに共通する情報を定義する．FTFeatureBase クラスは，位相情報を保持するモデルのベースとなるクラスで，FFeatureBase クラスを継承する．そして，位相情報を保持するモデルを実装するクラスは，FTFeatureBase クラスを継承する．また，各モデルの位相情報を実装するクラスは，FeatureTopologyBase クラスを継承する．各モデルを実装するクラスは，各モデルの位相情報を実装するクラスをプロパティとして保持する．ただし，複合図形は，位相情報を保持しないため，FFeatureBase クラスのみを継承する．下記の各クラスについて説明する．

3.2.1 共通の情報を保持するクラス

FFeatureBase クラスは，各モデルの共通の情報を定義する．FFeatureBase クラスのメンバ変数を次に示す.

パラメータ	型	説明
m_iFeatureID	int	モデル ID
m_FeatureType	FEATURE_TYPE	モデルの種類
m_sFeatureName	String	モデル名
m_iLayerCode	int	レイヤコード
m_ColorObj	FeatureColor	色コード
m_LineTypeObj	FeatureLineType	線種コード
m_LineWidthObj	FeatureLineWidth	線幅コード
m_bContainTopology	bool	位相情報の有無

3.2.2 位相情報を保持するモデルのベースとなるクラス

FTFeatureBase クラスは，位相情報を保持するモデルのベースとなるクラスである．FTFeatureBase クラスのメンバ変数を次に示す.

パラメータ	型	説明
m_FeatureModelType	FEATURE_MODEL_TYPE	モデルの表示方法
m_dVectorX_X	double	x 軸ベクトルの x 方向の大きさ
m_dVectorX_Y	double	x 軸ベクトルの y 方向の大きさ
m_dVectorX_Z	double	x 軸ベクトルの z 方向の大きさ
m_dVectorY_X	double	y 軸ベクトルの x 方向の大きさ
m_dVectorY_Y	double	y 軸ベクトルの y 方向の大きさ
m_dVectorY_Z	double	y 軸ベクトルの z 方向の大きさ
m_dVectorZ_X	double	z 軸ベクトルの x 方向の大きさ
m_dVectorZ_Y	double	z 軸ベクトルの y 方向の大きさ
m_dVectorZ_Z	double	z 軸ベクトルの z 方向の大きさ

3.2.3 位相情報のベースとなるクラス

FeatureTopologyBase クラスは，位相情報のベースとなるクラスである．FeatureTopologyBase クラスは，メンバ変数を保持しない．その代わりに，位相の頂点, 稜線, 面の情報を関連付ける役割を持つ. 頂点, 稜線, 面の情報は, それぞれ, FeatureVertex クラス，FeatureEdge クラス，FeatureFace クラスが保持する．FeatureVertex クラス，FeatureEdge クラス，FeatureFace クラスについては次で説明する.

- 頂点（FeatureVertex）

FeatureVertex クラスは，位相の頂点の情報を保持する．FeatureVertex クラスのメンバ変数を次に示す．

パラメータ	型	説明
m_dVertexX	double	頂点の x 座標
m_dVertexY	double	頂点の y 座標
m_dVertexZ	double	頂点の z 座標

- 稜線（FeatureEdge）

FeatureEdge クラスは，位相の稜線の情報を保持する．FeatureEdge クラスのメンバ変数を次に示す．

パラメータ	型	説明
m_StartVertex	FeatureVertex	稜線の始点
m_EndVertex	FeatureVertex	稜線の終点

- 面（FeatureFace）

FeatureFace クラスは，位相の面の情報を保持する．FeatureFace クラスのメンバ変数を次に示す．

パラメータ	型	説明
m_hVertexManager	Hashtable	面の頂点に関する情報
m_hEdgeManager	Hashtable	面の稜線に関する情報

3.2.4 点マーカ

本項では，点マーカの情報を保持するクラスについて説明する．

- 点マーカ（FPointMaker）

FPointMarker クラスでは，点マーカの幾何情報を保持する．FPointMarker クラスのメンバ変数を次に示す．

パラメータ	型	説明
m_dStartX	double	配置点の x 座標
m_dStartY	double	配置点の y 座標
m_dStartZ	double	配置点の z 座標
m_iMarkerCode	int	マーカコード
m_dScale	double	尺度

m_PointMarkerTopology	PointMarkerTopology	点マーカの位相情報

- 点マーカの位相情報（PointMakerTopology）

PointMarkerTopology クラスでは，点マーカの位相情報を保持する．PointMarkerTopology クラスのメンバ変数を次に示す.

パラメータ	型	説明
m_hVertex	Hashtable	頂点に関する情報

3.2.5 線分

本項では，線分の情報を保持するクラスについて説明する.

- 線分（FLine）

FLine クラスでは，線分の幾何情報を保持する．FLine クラスのメンバ変数を次に示す.

パラメータ	型	説明
m_dStartX	double	始点の x 座標
m_dStartY	double	始点の y 座標
m_dStartZ	double	始点の z 座標
m_dEndX	double	終点の x 座標
m_dEndY	double	終点の y 座標
m_dEndZ	double	終点の z 座標
m_LineTopology	LineTopology	線分の位相情報

- 線分の位相情報（LineTopology）

LineTopology クラスでは，線分の位相情報を保持する．LineTopology クラスのメンバ変数を次に示す.

パラメータ	型	説明
m_hEdge	Hashtable	稜線に関する情報

3.2.6 折線

本項では，折線の情報を保持するクラスについて説明する.

- 折線（FPolyline）

FPolyline クラスでは，折線の幾何情報を保持する．FPolyline クラスのメンバ変数を次に示す.

パラメータ	型	説明
m_iNumber	int	頂点数
m_alX	ArrayList	x 座標の配列
m_alY	ArrayList	y 座標の配列
m_alZ	ArrayList	z 座標の配列
m_PolylineTopology	PolylineTopology	線分の位相情報

- 折線の位相情報（PolylineTopology）

PolylineTopology クラスでは，折線の位相情報を保持する．PolylineTopology クラスのメンバ変数を次に示す.

パラメータ	型	説明
m_hEdge	Hashtable	稜線に関する情報

3.2.7 円

本項では，円の情報を保持するクラスについて説明する.

- 円（FCircle）

FCircle クラスでは，円の幾何情報を保持する．FCircle クラスのメンバ変数を次に示す.

パラメータ	型	説明
m_dCenterX	double	中心点の x 座標
m_dCenterY	double	中心点の y 座標
m_dCenterZ	double	中心点の z 座標
m_dRadius	double	半径
m_CircleTopology	CircleTopology	円の位相情報

- 円の位相情報（CircleTopology）

CircleTopology クラスでは，円の位相情報を保持する．CircleTopology クラスのメンバ変数を次に示す.

パラメータ	型	説明
m_hEdge	Hashtable	稜線に関する情報

3.2.8 円弧

本項では，円弧の情報を保持するクラスについて説明する.

- 円弧（FArc）

FArc クラスでは，円弧の幾何情報を保持する．FArc クラスのメンバ変数を次に示す.

パラメータ	型	説明
m_dCenterX	double	中心点の x 座標
m_dCenterY	double	中心点の y 座標
m_dCenterZ	double	中心点の z 座標
m_dRadius	double	半径
m_iDirection	int	向きフラグ
m_EllipseTopology	EllipseTopology	楕円の位相情報
m_dStartAngle	double	始角
m_dEndAngle	double	終角
m_ArcTopology	ArcTopology	円弧の位相情報

- 円弧の位相情報（ArcTopology）

ArcTopology クラスでは，円弧の位相情報を保持する．ArcTopology クラスのメンバ変数を次に示す.

パラメータ	型	説明
m_hEdge	Hashtable	稜線に関する情報

3.2.9 楕円

本項では，楕円の情報を保持するクラスについて説明する.

- 楕円（FEllipse）

FEllipse クラスでは，楕円の幾何情報を保持する．FEllipse クラスのメンバ変数を次に示す.

パラメータ	型	説明
m_dCenterX	double	中心点の x 座標
m_dCenterY	double	中心点の y 座標
m_dCenterZ	double	中心点の z 座標
m_dRadiusX	double	長半径
m_dRadiusY	double	短半径
m_EllipseTopology	EllipseTopology	楕円の位相情報

- 楕円の位相情報（EllipseTopology）

EllipseTopology クラスでは，楕円の位相情報を保持する．EllipseTopology クラスのメンバ変数を次に示す.

パラメータ	型	説明
m_hEdge	Hashtable	稜線に関する情報

3.2.10　楕円弧

本項では，楕円弧の情報を保持するクラスについて説明する.

- 楕円弧（FEllipseArc）

FEllipseArc クラスでは，楕円弧の幾何情報を保持する．FEllipseArc クラスのメンバ変数を次に示す.

パラメータ	型	説明
m_dCenterX	double	中心の x 座標
m_dCenterY	double	中心の y 座標
m_dCenterZ	double	中心の z 座標
m_dRadiusX	double	長半径
m_dRadiusY	double	短半径
m_iDirection	int	描画方向
m_dStartAngle	double	始角
m_dEndAngle	double	終角
m_EllipseArcTopology	EllipseArcTopology	楕円弧の位相情報

- 楕円弧の位相情報（EllipseArcTopology）

EllipseArcTopology クラスでは，楕円弧の位相情報を保持する．EllipseArcTopology クラスのメンバ変数を次に示す.

パラメータ	型	説明
m_hEdge	Hashtable	稜線に関する情報

3.2.11　クロソイド

本項では，クロソイドの情報を保持するクラスについて説明する.

- クロソイド（FClothoid）

FClothoid クラスでは，クロソイドの幾何情報を保持する．FClothoid クラスのメンバ変数を次に示す.

パラメータ	型	説明
m_dBaseX	double	基点の x 座標
m_dBaseY	double	基点の y 座標
m_dBaseZ	double	基点の z 座標
m_iParameter	int	パラメータ
m_iDirection	int	描画方向
m_dStartLength	double	開始曲線長
m_dEndLength	double	終了曲線長
m_dAngle	double	回転角
m_ClothoidTopology	ClothoidTopology	クロソイドの位相情報

描画方向は，クロソイドを作図する方向を指定する．描画方向が 0 の場合は，反時計廻りにクロソイドを描画する．描画方向が 1 の場合は，時計廻りにクロソイドを作図する．

- クロソイドの位相情報（ClothoidTopology）

ClothoidTopology クラスでは，クロソイドの位相情報を保持する．ClothoidTopology クラスのメンバ変数を次に示す．

パラメータ	型	説明
m_hEdge	Hashtable	稜線に関する情報

3.2.12 ベジェ曲線

本項では，ベジェ曲線の情報を保持するクラスについて説明する．

- ベジェ曲線（FBezierSpline）

FBezierSpline クラスでは，ベジェ曲線の幾何情報を保持する．FBezierSpline クラスのメンバ変数を次に示す．

パラメータ	型	説明
m_iOpenClose	int	開閉区分
m_iOrder	int	階数
m_iNumber	int	制御点数
m_alX	ArrayList	x 座標の配列
m_alY	ArrayList	y 座標の配列
m_alZ	ArrayList	z 座標の配列
m_BezierSplineTopology	BezierSplineTopology	ベジェ曲線の位相情報

- ベジェ曲線の位相情報（BezierSplineTopology）

BezierSplineTopology クラスでは, ベジェ曲線の位相情報を保持する. BezierSplineTopology クラスのメンバ変数を次に示す.

パラメータ	型	説明
m_hEdge	Hashtable	稜線に関する情報

3.2.13 NURBS曲線

本項では，NURBS 曲線の情報を保持するクラスについて説明する.

- NURBS 曲線（FNurbsSpline）

FNurbsSpline クラスでは，NURBS 曲線の幾何情報を保持する．FNurbsSpline クラスのメンバ変数を次に示す.

パラメータ	型	説明
m_iOpenClose	int	開閉区分
m_iOrder	int	階数
m_iNumber	int	制御点数
m_alKnot	ArrayList	knot ベクトルの配列
m_alX	ArrayList	x 座標の配列
m_alY	ArrayList	y 座標の配列
m_alZ	ArrayList	z 座標の配列
m_alWeight	ArrayList	重みの配列
m_NurbsSplineTopology	NurbsSplineTopology	NURBS 曲線の位相情報

開閉区分とは，NURBS 曲線の開閉を指定する値である．開閉区分が 0 の場合，閉じた NURBS 曲線を作図する．開閉区分が 1 の場合，開いた NURBS 曲線を作図する．制御点数とは，作成する NURBS 曲線全体で必要となる頂点の数であり，階数とは，1 次の NURBS 曲線を作図する際に必要となる頂点の数である．knot ベクトルの配列とは，knot ベクトルを保持する配列であり，重みの配列とは，各頂点におけるウェイトの値を保持する配列である．N 次の階数で構成される NURBS 曲線を M 個作図する場合，制御点数は，(N-1)*M + 1 個になる．また，指定する knot ベクトルの数は，N+M 個になる．knot ベクトルの値を指定する際には，前項より大きいか，同じ値を指定する．そのため，i 番目の knot ベクトルを指定する際には，i 番目の knot ベクトルの値が knot[i]≧knot[i-1]となるように指定する.

- NURBS 曲線の位相情報（NurbsSplineTopology）

NurbsSplineTopology クラスでは, NURBS 曲線の位相情報を保持する. NurbsSplineTopology クラスのメンバ変数を次に示す.

パラメータ	型	説明
m_hEdge	Hashtable	稜線に関する情報

3.2.14 直方体

本項では，直方体の情報を保持するクラスについて説明する．

- 直方体（FBlock）

FBlock クラスでは，直方体の幾何情報を保持する．FBlock クラスのメンバ変数を次に示す．

パラメータ	型	説明
m_dCenterX	double	配置点の x 座標
m_dCenterY	double	配置点の y 座標
m_dCenterZ	double	配置点の z 座標
m_dLengthX	double	x 軸方向への大きさ
m_dLengthY	double	y 軸方向への大きさ
m_dLengthZ	double	z 軸方向への大きさ
m_BlockTopology	BlockTopology	直方体の位相情報

- 直方体の位相情報（BlockTopology）

BlockTopology クラスでは，直方体の位相情報を保持する．BlockTopology クラスのメンバ変数を次に示す．

パラメータ	型	説明
m_hFace	Hashtable	面に関する情報

3.2.15 ウェッジ

本項では，ウェッジの情報を保持するクラスについて説明する．

- ウェッジ（FWedge）

FWedge クラスでは，ウェッジの幾何情報を保持する．FWedge クラスのメンバ変数を次に示す．

パラメータ	型	説明
m_dCenterX	double	配置点の x 座標
m_dCenterY	double	配置点の y 座標

m_dCenterZ	double	配置点の z 座標
m_dLengthX	double	x 軸方向への大きさ
m_dLengthY	double	y 軸方向への大きさ
m_dLengthZ	double	z 軸方向への大きさ
m_dLtX	double	x 軸方向の短い辺の大きさ
m_WedgeTopology	WedgeTopology	ウェッジの位相情報

- ウェッジの位相情報（WedgeTopology）

WedgeTopology クラスでは，ウェッジの位相情報を保持する．WedgeTopology クラスのメンバ変数を次に示す.

パラメータ	型	説明
m_hFace	Hashtable	面に関する情報

3.2.16 球

本項では，球の情報を保持するクラスについて説明する.

- 球（FSphere）

FSphere クラスでは, 球の幾何情報を保持する. FSphere クラスのメンバ変数を次に示す.

パラメータ	型	説明
m_dCenterX	double	配置点の x 座標
m_dCenterY	double	配置点の y 座標
m_dCenterZ	double	配置点の z 座標
m_dRadius	double	半径
m_SphereTopology	SphereTopology	球の位相情報

- 球の位相情報（SphereTopology）

SphereTopology クラスでは，球の位相情報を保持する．SphereTopology クラスのメンバ変数を次に示す.

パラメータ	型	説明
m_hFace	Hashtable	面に関する情報

3.2.17　円柱

本項では，円柱の情報を保持するクラスについて説明する．

- 円柱（FCylinder）

FCylinder クラスでは，円柱の幾何情報を保持する．FCylinder クラスのメンバ変数を次に示す．

パラメータ	型	説明
m_dCenterX	double	配置点の x 座標
m_dCenterY	double	配置点の y 座標
m_dCenterZ	double	配置点の z 座標
m_dHeight	double	高さ
m_dRadius	double	半径
m_CylinderTopology	CylinderTopology	円柱の位相情報

- 円柱の位相情報（CylinderTopology）

CylinderTopology クラスでは，円柱の位相情報を保持する．CylinderTopology クラスのメンバ変数を次に示す．

パラメータ	型	説明
m_hFace	Hashtable	面に関する情報

3.2.18　円錐

本項では，円錐の情報を保持するクラスについて説明する．

- 円錐（FCone）

FCone クラスでは，円錐の幾何情報を保持する．FCone クラスのメンバ変数を次に示す．

パラメータ	型	説明
m_dCenterX	double	配置点の x 座標
m_dCenterY	double	配置点の y 座標
m_dCenterZ	double	配置点の z 座標
m_dRadius	double	半径
m_dHeight	double	高さ
m_dSemiAngle	double	母線と回転軸の間の角度
m_ConeTopology	ConeTopology	円錐の位相情報

- 円錐の位相情報（ConeTopology）

ConeTopology クラスでは，円錐の位相情報を保持する．ConeTopology クラスのメンバ変数を次に示す．

パラメータ	型	説明
m_hFace	Hashtable	面に関する情報

3.2.19 トーラス

本項では，トーラスの情報を保持するクラスについて説明する．

- トーラス（FTorus）

FTorus クラスでは，トーラスの幾何情報を保持する．FTorus クラスのメンバ変数を次に示す．

パラメータ	型	説明
m_dCenterX	double	配置点の x 座標
m_dCenterY	double	配置点の y 座標
m_dCenterZ	double	配置点の z 座標
m_dMajorRadius	double	大半径
m_dMinorRadius	double	小半径
m_TorusTopology	TorusTopology	トーラスの位相情報

- トーラスの位相情報（TorusTopology）

TorusTopology クラスでは，トーラスの位相情報を保持する．TorusTopology クラスのメンバ変数を次に示す．

パラメータ	型	説明
m_hFace	Hashtable	面に関する情報

3.2.20 ベジェ曲面

本項では，ベジェ曲面の情報を保持するクラスについて説明する．

- ベジェ曲面（FBezierSurface）

FBezierSurface クラスでは，ベジェ曲面の幾何情報を保持する．FBezierSurface クラスのメンバ変数を次に示す．

パラメータ	型	説明
m_iNumber	int	制御点数
m_iOrder1	int	平面の階数
m_iOrder2	int	奥行の階数
m_alX	ArrayList	x 座標の配列
m_alY	ArrayList	y 座標の配列
m_alZ	ArrayList	z 座標の配列
m_BezierSurfaceTopology	BezierSurfaceTopology	ベジェ曲線の位相情報

- ベジェ曲面の位相情報（BezierSurfaceTopology）

BezierSurfaceTopology クラスでは，ベジェ曲面の位相情報を保持する．BezierSurfaceTopology クラスのメンバ変数を次に示す．

パラメータ	型	説明
m_hFace	Hashtable	面に関する情報

3.2.21 NURBS曲面

本項では，NURBS 曲面の情報を保持するクラスについて説明する．

- NURBS 曲面（FNurbsSurface）

FNurbsSurface クラスでは，NURBS 曲面の幾何情報を保持する．FNurbsSurface クラスのメンバ変数を次に示す．

パラメータ	型	説明
m_iNumber	int	頂点数
m_iOrder1	int	平面 制御点数
m_iOrder2	int	奥行 制御点数
m_alKnot1	ArrayList	平面 knot ベクトル
m_alKnot2	ArrayList	奥行 knot ベクトル
m_alX	ArrayList	x 座標の配列
m_alY	ArrayList	y 座標の配列
m_alZ	ArrayList	z 座標の配列
m_alWeight	ArrayList	ウェイトの配列
m_NurbsSurfaceTopology	NurbsSurfaceTopology	NURBS 曲線の位相情報

- NURBS 曲面の位相情報（NurbsSurfaceTopology）

NurbsSurfaceTopology クラスでは，NURBS 曲面の位相情報を保持する．NurbsSurfaceTopology クラスのメンバ変数を次に示す．

パラメータ	型	説明
m_hFace	Hashtable	面に関する情報

3.2.22　複合曲線

本項では，複合曲線の情報を保持するクラスについて説明する．

- 複合曲線（FCompositeCurve）

FCompositeCurve クラスでは，複合曲線の幾何情報を保持する．FCompositeCurve クラスのメンバ変数を次に示す．

パラメータ	型	説明
m_alFeature	ArrayList	モデルの配列
m_alX	ArrayList	x 座標の配列
m_alY	ArrayList	y 座標の配列
m_alZ	ArrayList	z 座標の配列
m_CompositeCurveTopology	CompositeCurveTopology	複合曲線の位相情報

- 複合曲線の位相情報（CompositeCurveTopology）

CompositeCurveTopology クラスでは，複合曲線の位相情報を保持する．CompositeCurveTopology クラスのメンバ変数を次に示す．

パラメータ	型	説明
m_hFace	Hashtable	面に関する情報

3.2.23　押出面

本項では，押出面の情報を保持するクラスについて説明する．

- 押出面（FSurfaceLinearExtrusion）

FSurfaceLinearExtrusion クラスでは, 押出面の幾何情報を保持する. FSurfaceLinearExtrusion クラスのメンバ変数を次に示す．

パラメータ	型	説明
m_fFeatureOfSweep	FFeatureBase	押出の対象モデル
m_dSweepVectorX	double	押出方向ベクトルの x 方向

m_dSweepVectorY	double	押出方向ベクトルの y 方向
m_dSweepVectorZ	double	押出方向ベクトルの z 方向
m_SurfaceLinearExtrusionTopology	SurfaceLinearExtrusionTopology	押出面の位相情報

- 押出面の位相情報（SurfaceLinearExtrusionTopology）

SurfaceLinearExtrusionTopology クラスでは，押出面の位相情報を保持する．SurfaceLinearExtrusionTopology クラスのメンバ変数を次に示す．

パラメータ	型	説明
m_hFace	Hashtable	面に関する情報

3.2.24 掃引面

本項では，掃引面の情報を保持するクラスについて説明する．

- 掃引面（FSurfaceSweep）

FSurfaceSweep クラスでは，掃引面の幾何情報を保持する．FSurfaceSweep クラスのメンバ変数を次に示す．

パラメータ	型	説明
m_fFeatureOfSweep	FFeatureBase	掃引の対象モデル
m_fSweepLine	FFeatureBase	掃引線のモデル
m_SurfaceSweepTopology	SurfaceSweepTopology	掃引面の位相情報

- 掃引面の位相情報（SurfaceSweepTopology）

SurfaceSweepTopology クラスでは，掃引面の位相情報を保持する．SurfaceSweepTopology クラスのメンバ変数を次に示す．

パラメータ	型	説明
m_hFace	Hashtable	面に関する情報

3.2.25 回転面

本項では，回転面の情報を保持するクラスについて説明する．

- 回転面（FSurfaceRevolution）

FSurfaceRevolution クラスでは，回転面の幾何情報を保持する．FSurfaceRevolution クラスのメンバ変数を次に示す.

パラメータ	型	説明
m_fFeature	FFeatureBase	回転の対象モデル
m_dAxisX	double	回転軸の中心 x 座標
m_dAxisY	double	回転軸の中心 y 座標
m_dAxisZ	double	回転軸の中心 z 座標
m_SurfaceRevolutionTopology	SurfaceRevolutionTopology	回転面の位相情報

- 回転面の位相情報（SurfaceRevolutionTopology）

SurfaceRevolutionTopology クラスでは，回転面の位相情報を保持する．SurfaceRevolutionTopology クラスのメンバ変数を次に示す.

パラメータ	型	説明
m_hFace	Hashtable	面に関する情報

3.2.26 複合図形

本項では，複合図形の情報を保持するクラスについて説明する．FCompositeFigure クラスは，複合図形の情報を保持する．FCompositeFigure クラスのメンバ変数を次に示す.

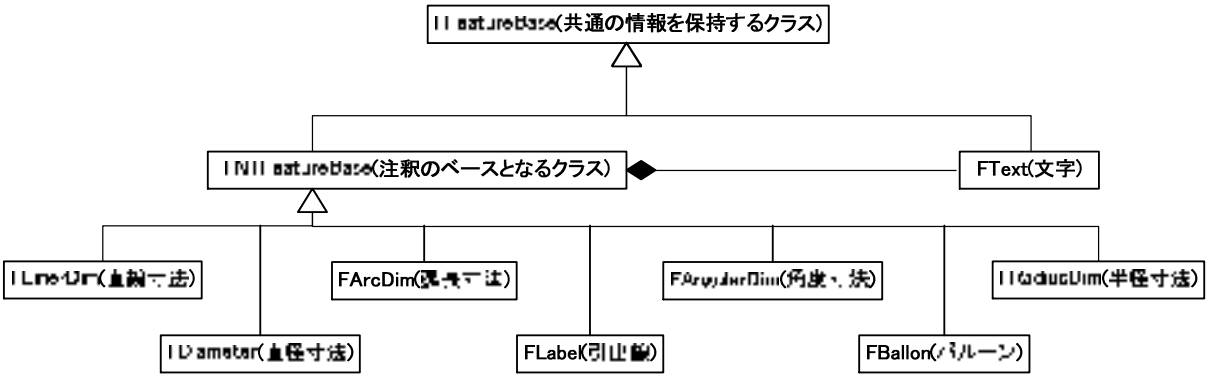
パラメータ	型	説明
m_dCenterX	double	配置点の x 座標
m_dCenterY	double	配置点の y 座標
m_dCenterZ	double	配置点の z 座標
m_STFeature	FFeatureBase	基準モデル
m_PPFeature	FFeatureBase	関連モデル
m_dVectorX	double	単位ベクトルの x 方向の大きさ
m_dVectorY	double	単位ベクトルの y 方向の大きさ
m_dVectorZ	double	単位ベクトルの z 方向の大きさ
m_dDistance	double	基点間の距離

3.3 注釈情報を保持するクラス

市販される 3 次元 CAD においては，注釈情報の詳細な形状の定義を管理するものと，そうでないものが存在する．前者については，カメラの表示方向に影響を受けず，常に同じ形状で注釈が表示される．後者においては，カメラの表示方向に応じて，注釈が常に正面

方向を向くように動的に形状が変更される．このうち，本研究では，前者の方法を採用する．したがって，注記情報については，3 次元空間上における一意な形状を保持するためのクラス構造を定義する．また，形状の定義においては，SXF 仕様を参考としており，補助線形状や矢印形状等の詳細な形状の定義を可能とする．

本研究で定義したクラス構造を以下に示す．



FNTFeatureBase クラスは，注釈のベースとなるクラスで，FTFeatureBase クラスと同様で FFeatureBase クラスを継承する．そして，注釈毎にクラスを定義し，注釈を実装するクラスは，FTFeatureBase クラスを継承する．ただし，文字は，全ての寸法で使用するため，FNTFeatureBase クラスのプロパティとして保持する．

3.3.1 文字

本項では，文字の情報を保持するクラスについて説明する．FText クラスでは，文字の幾何情報を保持する．FText クラスのメンバ変数を次に示す．

パラメータ	型	説明
m_sFont	String	フォント名
m_sString	String	文字列
m_dTextX	double	配置点の x 座標
m_dTextY	double	配置点の y 座標
m_dTextZ	double	配置点の z 座標
m_dHeight	double	文字列範囲の高さ
m_dWidth	double	文字列範囲の幅
m_dSpc	double	文字間隔
m_dSlant	double	スラント角度

m_iDirect	int	文字方向
m_iB_pnt	int	文字列の配置位置

3.3.2 直線寸法

本項では, 直線寸法の情報を保持するクラスについて説明する. FLinearDim クラスでは, 直線寸法の幾何情報を保持する. FLinearDim クラスのメンバ変数を次に示す.

パラメータ	型	説明
m_iConnectFeatureID	int	関連付けたモデル名
m_iVertexID1	int	関連付いた頂点番号（始点）
m_iVertexID2	int	関連付いた頂点番号（終点）
m_dSunX1	double	寸法線始 X 座標
m_dSunY1	double	寸法線始 Y 座標
m_dSunZ1	double	寸法線始 Z 座標
m_dSunX2	double	寸法線終 X 座標
m_dSunY2	double	寸法線終 Y 座標
m_dSunZ2	double	寸法線終 Z 座標
m_iFlg2	int	補助線 1 の有無
m_dHo1X0	double	補助線 1 の基点の x 座標
m_dHo1Y0	double	補助線 1 の基点の y 座標
m_dHo1Z0	double	補助線 1 の基点の z 座標
m_dHo1X1	double	補助線 1 の始点の x 座標
m_dHo1Y1	double	補助線 1 の始点の y 座標
m_dHo1Z1	double	補助線 1 の始点の z 座標
m_dHo1X2	double	補助線 1 の終点の x 座標
m_dHo1Y2	double	補助線 1 の終点の y 座標
m_dHo1Z2	double	補助線 1 の終点の z 座標
m_iFlg3	int	補助線 2 の有無
m_dHo2X0	double	補助線 2 の基点の x 座標
m_dHo2Y0	double	補助線 2 の基点の y 座標
m_dHo2Z0	double	補助線 2 の基点の z 座標
m_dHo2X1	double	補助線 2 の始点の x 座標
m_dHo2Y1	double	補助線 2 の始点の y 座標
m_dHo2Z1	double	補助線 2 の始点の z 座標
m_dHo2X2	double	補助線 2 の終点の x 座標
m_dHo2Y2	double	補助線 2 の終点の y 座標
m_dHo2Z2	double	補助線 2 の終点の z 座標
m_iArr1Code1	int	始点の矢印コード
m_iArr1Code2	int	始点の矢印の向き
m_dArr1X	double	始点の矢印の x 座標
m_dArr1Y	double	始点の矢印の y 座標
m_dArr1Z	double	始点の矢印の z 座標
m_dArr1R	double	始点の矢印の倍率

m_iArr2Code1	int	終点の矢印コード
m_iArr2Code2	int	終点の矢印の向き
m_dArr2X	double	終点の矢印の x 座標
m_dArr2Y	double	終点の矢印の y 座標
m_dArr2Z	double	終点の矢印の z 座標
m_dArr2R	double	終点の矢印の倍率
m_iFlg4	int	寸法値の有無

3.3.3 直径寸法

本項では，直径寸法の情報を保持するクラスについて説明する．FDiameterDim クラスでは，直径寸法の幾何情報を保持する．FDiameterDim クラスのメンバ変数を次に示す.

パラメータ	型	説明
m_iFeatureID	int	関連付けたモデル名
m_iVertexID1	int	関連付けた頂点番号
m_iVertexID2	int	関連付けた頂点番号
m_dSunX1	double	寸法線始 X 座標
m_dSunY1	double	寸法線始 Y 座標
m_dSunZ1	double	寸法線始 Z 座標
m_dSunX2	double	寸法線終 X 座標
m_dSunY2	double	寸法線終 Y 座標
m_dSunZ2	double	寸法線終 Z 座標
m_dAngle	double	寸法線の方向
m_iArr1Code1	int	始点の矢印コード
m_iArr1Code2	int	始点の矢印の向き
m_dArr1X	double	始点の矢印の x 座標
m_dArr1Y	double	始点の矢印の y 座標
m_dArr1Z	double	始点の矢印の z 座標
m_dArr1R	double	始点の矢印の倍率
m_iArr2Code1	int	終点の矢印コード
m_iArr2Code2	int	終点の矢印の向き
m_dArr2X	double	終点の矢印の x 座標
m_dArr2Y	double	終点の矢印の y 座標
m_dArr2Z	double	終点の矢印の z 座標
m_dArr2R	double	終点の矢印の倍率
m_iFlg	int	寸法値の有無

3.3.4 半径寸法

本項では, 半径寸法の情報を保持するクラスについて説明する. FRadiusDim クラスでは, 半径寸法の幾何情報を保持する．FRadiusDim クラスのメンバ変数を次に示す.

パラメータ	型	説明
m_iFeatureID	int	関連付けたモデル名
m_iVertexID	int	関連付けた頂点番号
m_dSunX1	double	寸法線始 X 座標
m_dSunY1	double	寸法線始 Y 座標
m_dSunZ1	double	寸法線始 Z 座標
m_dSunX2	double	寸法線終 X 座標
m_dSunY2	double	寸法線終 Y 座標
m_dSunZ2	double	寸法線終 Z 座標
m_iArrCode1	int	矢印コード
m_iArrCode2	int	矢印の向き
m_dArr1X	double	矢印の x 座標
m_dArr1Y	double	矢印の y 座標
m_dArr1Z	double	矢印の z 座標
m_dArr1R	double	矢印の倍率
m_iFlg	int	寸法値の有無

3.3.5 角度寸法

本項では，角度寸法の情報を保持するクラスについて説明する．FAngularDim クラスでは，角度寸法の幾何情報を保持する．FAngularDim クラスのメンバ変数を次に示す．

パラメータ	型	説明
m_iConnectFeatureID	int	関連付けたモデル名
m_iVertexID1	int	関連付けた頂点番号
m_iVertexID2	int	関連付けた頂点番号
m_dSunX	double	寸法線始 X 座標
m_dSunY	double	寸法線始 Y 座標
m_dSunZ	double	寸法線始 Z 座標
m_dSunRadius	double	寸法線の半径
m_dSunAngle0	double	寸法線の始角
m_dSunAngle1	double	寸法線の終角
m_iFlg2	int	補助線 1 の有無
m_dHo1X0	double	補助線 1 の基点の x 座標
m_dHo1Y0	double	補助線 1 の基点の y 座標
m_dHo1Z0	double	補助線 1 の基点の z 座標
m_dHo1X1	double	補助線 1 の始点の x 座標
m_dHo1Y1	double	補助線 1 の始点の y 座標
m_dHo1Z1	double	補助線 1 の始点の z 座標
m_dHo1X2	double	補助線 1 の終点の x 座標
m_dHo1Y2	double	補助線 1 の終点の y 座標
m_dHo1Z2	double	補助線 1 の終点の z 座標
m_iFlg3	int	補助線 2 の有無
m_dHo2X0	double	補助線 2 の基点の x 座標

m_dHo2Y0	double	補助線 2 の基点の y 座標
m_dHo2Z0	double	補助線 2 の基点の z 座標
m_dHo2X1	double	補助線 2 の始点の x 座標
m_dHo2Y1	double	補助線 2 の始点の y 座標
m_dHo2Z1	double	補助線 2 の始点の z 座標
m_dHo2X2	double	補助線 2 の終点の x 座標
m_dHo2Y2	double	補助線 2 の終点の y 座標
m_dHo2Z2	double	補助線 2 の終点の z 座標
m_iArr1Code1	int	始点の矢印コード
m_iArr1Code2	int	始点の矢印の向き
m_dArr1X	double	始点の矢印の x 座標
m_dArr1Y	double	始点の矢印の y 座標
m_dArr1Z	double	始点の矢印の z 座標
m_dArr1R	double	始点の矢印の倍率
m_iArr2Code1	int	終点の矢印コード
m_iArr2Code2	int	終点の矢印の向き
m_dArr2X	double	終点の矢印の x 座標
m_dArr2Y	double	終点の矢印の y 座標
m_dArr2Z	double	終点の矢印の z 座標
m_dArr2R	double	終点の矢印の倍率
m_iFlg4	int	寸法値の有無

3.3.6 弧長寸法

本項では，弧長寸法の情報を保持するクラスについて説明する．FArcDim クラスでは，弧長寸法の幾何情報を保持する．FArcDim クラスのメンバ変数を次に示す.

パラメータ	型	説明
m_iConnectFeatureID	int	関連付けたモデル名
m_iVertexID1	int	関連付けた頂点番号
m_iVertexID2	int	関連付けた頂点番号
m_dSunX	double	寸法線始 X 座標
m_dSunY	double	寸法線始 Y 座標
m_dSunZ	double	寸法線始 Z 座標
m_dSunRadius	double	寸法線の半径
m_dSunAngle0	double	寸法線の始角
m_dSunAngle1	double	寸法線の終角
m_iFlg2	int	補助線 1 の有無
m_dHo1X0	double	補助線 1 の基点の x 座標
m_dHo1Y0	double	補助線 1 の基点の y 座標
m_dHo1Z0	double	補助線 1 の基点の z 座標
m_dHo1X1	double	補助線 1 の始点の x 座標
m_dHo1Y1	double	補助線 1 の始点の y 座標
m_dHo1Z1	double	補助線 1 の始点の z 座標

m_dHo1X2	double	補助線 1 の終点の x 座標
m_dHo1Y2	double	補助線 1 の終点の y 座標
m_dHo1Z2	double	補助線 1 の終点の z 座標
m_iFlg3	int	補助線 2 の有無
m_dHo2X0	double	補助線 2 の基点の x 座標
m_dHo2Y0	double	補助線 2 の基点の y 座標
m_dHo2Z0	double	補助線 2 の基点の z 座標
m_dHo2X1	double	補助線 2 の始点の x 座標
m_dHo2Y1	double	補助線 2 の始点の y 座標
m_dHo2Z1	double	補助線 2 の始点の z 座標
m_dHo2X2	double	補助線 2 の終点の x 座標
m_dHo2Y2	double	補助線 2 の終点の y 座標
m_dHo2Z2	double	補助線 2 の終点の z 座標
m_iArr1Code1	int	始点の矢印コード
m_iArr1Code2	int	始点の矢印の向き
m_dArr1X	double	始点の矢印の x 座標
m_dArr1Y	double	始点の矢印の y 座標
m_dArr1Z	double	始点の矢印の z 座標
m_dArr1R	double	始点の矢印の倍率
m_iArr2Code1	int	終点の矢印コード
m_iArr2Code2	int	終点の矢印の向き
m_dArr2X	double	終点の矢印の x 座標
m_dArr2Y	double	終点の矢印の y 座標
m_dArr2Z	double	終点の矢印の z 座標
m_dArr2R	double	終点の矢印の倍率
m_iFlg4	int	寸法値の有無

3.3.7 引出線

本項では，引出線の情報を保持するクラスについて説明する．FLabel クラスでは，引出線の幾何情報を保持する．FLabel クラスのメンバ変数を次に示す.

パラメータ	型	説明
m_iNumber	int	頂点数
m_alX	ArrayList	x 座標の配列
m_alY	ArrayList	y 座標の配列
m_alZ	ArrayList	z 座標の配列
m_iArrCode	int	矢印コード
m_dArrR	double	矢印倍率
m_iFlg	int	寸法値の有無

3.3.8 バルーン

本項では，バルーンの情報を保持するクラスについて説明する．FBalloon クラスでは，

バルーンの幾何情報を保持する．FBalloon クラスのメンバ変数を次に示す．

パラメータ	型	説明
m_iNumber	int	頂点数
m_alX	ArrayList	x 座標の配列
m_alY	ArrayList	y 座標の配列
m_alZ	ArrayList	z 座標の配列
m_dCenterX	double	中心の x 座標
m_dCenterY	double	中心の y 座標
m_dCenterZ	double	中心の z 座標
m_dRadius	double	半径
m_iArr1Code1	int	矢印コード
m_dArrR	double	矢印倍率
m_iFlg	int	寸法値の有無

3.4　その他のクラス

その他のクラスとして，モデルの視覚情報を管理するための構造と，システム全般の情報を管理するための構造を定義する．特に，視覚情報については，SXF 仕様を参考に，その構造を定義した．ISO10303 の定義においては，1 つのモデルに対して，複数の見え方を定義することができる．一方，SXF においては，1 つのモデルに対して，1 つの見え方のみを定義することができる．市場に提供される 3 次元 CAD においては，一般的に SXF と同様となっている．そこで，本研究においても，モデルに対して，1 つの見え方のみを定義できるものとした．

3.4.1　レイヤ

本項では，レイヤの情報を保持するクラスについて説明する．FLayer クラスでは，レイヤの情報を保持する．FLayer クラスのメンバ変数を次に示す．

メンバ変数	型	説明	備考
m_sLayerName	String	レイヤ名	
m_bFlag	bool	表示/非表示フラグ	0:非表示　1:表示

3.4.2　色

本項では，色の情報を保持するクラスについて説明する．FeatureColor クラスでは，色の情報を保持する．FeatureColor クラスのメンバ変数を次に示す．

メンバ変数	型	説明

m_Color	FEATURE_COLOR	色コード

3.4.3 線種

本項では, 線種の情報を保持するクラスについて説明する. FeatureLineType クラスでは, 線種の情報を保持する．FeatureLineType クラスのメンバ変数を次に示す.

メンバ変数	型	説明
m_LineType	FEATURE_LINETYPE	線種コード

3.4.4 線幅

本項では, 線幅の情報を保持するクラスについて説明する. FeatureLineWidth クラスでは, 線幅の情報を保持する．FeatureLineWidth クラスのメンバ変数を次に示す.

メンバ変数	型	説明
m_LineWidth	FEATURE_LINEWIDTH	線幅コード

3.4.5 フォント

本項では，フォントの情報を保持するクラスについて説明する．FFont クラスでは，フォントの情報を保持する．FFont クラスのメンバ変数を次に示す.

メンバ変数	型	説明
m_sFont	String	フォント名

3.4.6 各種管理クラス

本項では，各種の管理クラスとして，wgl 設定，データ管理，ビュー管理，ファイル入出力管理のクラスについて説明する.

- wglFormat クラス，wglRender クラス

wglFormat クラスと wglRender クラスは，wgl によるモデル空間の設定やモデルの描画を設定するクラスである．wglFormat クラスは，Windows アプリケーションにおけるモデル空間の設定情報を保持する．wglFormat クラスのメンバ変数を次に示す.

メンバ変数	型	説明
m_HWnd	HWND	ハンドル
m_HDC	HDC	デバイスコンテキスト

m_Hglrc	HGLRC	レンダリングコンテキスト

wglRender クラスでは，Windows アプリケーションにおけるモデルの描画を設定する．wglRender クラスのメンバ変数を次に示す．

メンバ変数	型	説明
m_wglFormatSet	wglFormat	wgl による描画空間設定情報

- DataManager クラス

DataManager クラスは，本システムの様々な情報を管理するクラスである．DataManager クラスのメンバ変数を次に示す．

メンバ変数	型	説明
m_iFeatureID	int	モデル作成 ID
m_iSelectedFeatureID	int	現在フォーム上で選択されているモデルの ID
m_iSelectedLayerCode	ArrayList	現在フォーム上で選択されているレイヤコード
m_SelectedFeatureColor	FEATURE_COLOR	現在フォーム上で選択されている色情報
m_SelectedFeatureLineType	FEATURE_LINETYPE	現在フォーム上で選択されている線種情報
m_SelectedFeatureLineWidth	FEATURE_LINEWIDTH	現在フォーム上で選択されている線幅情報
m_strSelectedFont	String	現在フォーム上で選択されているフォントコード
m_hFeatureManager	Hashtable	モデル管理オブジェクト
m_alLayerManager	ArrayList	レイヤ管理オブジェクト
m_hFontManager	Hashtable	フォントコード管理オブジェクト
m_VManager	ViewManager	ビュー管理オブジェクト

- ViewManager クラス

ViewManager クラスは，本システムの視点やカメラの情報を管理するクラスである．ViewManager クラスのメンバ変数を次に示す．

メンバ変数	型	説明
m_ProjectionType	PROJECTION_TYPE	射影タイプ
m_dDistance	double	注視点からの距離
m_dElevation	double	注視点からの高さを表す角度

m_dAzimuth	double	注視点からの方向を表す角度
m_dX	double	視点の位置を表す X 座標
m_dY	double	視点の位置を表す Y 座標
m_dLeft	double	描画空間左端
m_dRight	double	描画空間右端
m_dBottom	double	描画空間下部
m_dTop	double	描画空間上部
m_dZNear	double	描画空間手前
m_dZFar	double	描画空間奥
m_dWidth	double	描画空間幅
m_dHeight	double	描画空間高さ

- FileReader クラス，FileWriter クラス

FileReader クラスと FileWriter クラスは，本システムで作成したモデルの情報の読込みや保存を管理するクラスである．FileReader クラスでは，保存した図面ファイルからモデルの情報を読込みためのクラスである．FileReader クラスのメンバ変数を次に示す．

メンバ変数	型	説明
m_iFeatureID	int	モデル ID
m_srReader	StreamReader	入力用ストリームリーダ
m_dCenterX	double	配置点 X 座標
m_dCenterY	double	配置点 Y 座標
m_dCenterZ	double	配置点 Z 座標

FileWriter クラスは，モデルの情報を図面ファイルに保存するためのクラスである．FileWriter クラスのメンバ変数を次に示す．

メンバ変数	型	説明
m_swWriter	StreamWriter	出力用のストリームライター
m_iCounter	int	モデル数

4　実装仕様

4.1　実装仕様の定義

本章では，3 次元アプリケーションの実装仕様について定義する．平成 20 年度の研究においては，「報告書　第 5 章　OpenGL による 3 次元モデルの表示方法」「報告書　第 6 章 OpenGL による CAD 機能の実装方法」において，OpenGL を用いた以下の機能の実装方法について取りまとめた．

機能		説明
表示機能	注釈	各寸法線（直線，角度，引出など）の描画方法
	幾何属性の操作	モデルの色，線種，線幅の変更方法
	文字	文字要素の描画方法
	カメラ操作	カメラの操作方法
	投影法操作	透視投影と平行投影の実装方法
モデル表示	点要素	点マーカの描画方法
	曲線要素	線分，折線，円，円弧，楕円，楕円弧などの描画方法
	面要素	直方体，球，円柱，トーラスなどの描画方法

そこで，本研究では，上記の機能に加え，以下の機能を新たに調査することで，3 次元グラフィックスアプリケーションの実装仕様として取りまとめた．

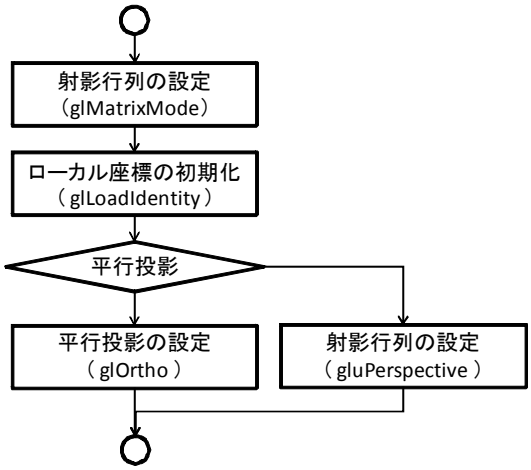
機能		説明
面作成（2D-3D）	押し出しによる面の作成	曲線要素の押し出しによる面要素の作成方法
	回転による面の作成	曲線要素の回転による面要素の作成方法
	スイープによる面の作成	曲線要素のスイープによる面要素の作成方法
モデル編集	平行移動（複写）	作成したモデルの平行移動の方法
	回転移動（複写）	作成したモデルの回転移動の方法
	スケール（複写）	作成したモデルの拡大・縮小の方法
CAD 機能	ファイルの入出力	ファイル仕様の策定
	レイヤ表現	レイヤの切り替え方法を調査し，実装する．
	マウス操作関連	マウス操作によるモデルの描画・選択・編集方法

4.2　表示

本章では，ウィンドウへの投影方法（透視投影，平行投影）の実装方法，モデル空間上でのカメラの位置（ズームイン，ズームアウト，3D ビュー）の設定方法，モデルの表示方法（サーフェスモデル，ワイヤーフレームモデル）について説明する．

4.2.1　射影切替

射影切替では, 投影法を切り替えることで行う. OpenGL による投影法の設定では, まず, 射影行列を設定する．次に，ローカル座標を初期化する．最後に，投影法（平行投影，透視投影）を設定する．OpenGL での投影法の設定手順を以下に示す.



射影行列を設定するには glMatrixMode 関数を使用する．glMatrixMode 関数の定義を以下に示す.

```
void glMatrixMode(GLenum mode)
```

glMatrixMode 関数は 1 つの引数をとり，戻り値はない．glMatrixMode 関数の引数および本システムにおける初期値を次に示す.

引数名	型	引数の意味	初期値
mode	GLenum	行列演算の種類	GL_PROJECTION

ローカル座標を初期化するには glLoadIdentity 関数を使用する. glLoadIdentity 関数は引数も戻り値も存在しない.

平行投影を設定するには glOrtho 関数を使用する．glOrtho 関数の定義を以下に示す．

```
void glOrtho(GLdouble left,  GLdouble right,  GLdouble bottom,  GLdouble top,
             GLdouble near,  GLdouble far)
```

glOrtho 関数は 6 つの引数をとり，戻り値はない．glOrtho 関数の引数および本システムにおける初期値を次に示す．

引数名	型	引数の意味	初期値
left	GLdouble	投影面の左方向の値	-30
right	GLdouble	投影面の右方向の値	-30
bottom	GLdouble	投影面の下方向の値	-30*0.764
top	GLdouble	投影面の上方向の値	30*0.764
near	GLdouble	投影面の手前の位置を示すす値	0.1
far	GLdouble	投影面の奥行きを示す値	10000

また，透視投影を設定するには gluPerspective 関数を使用する．gluPerspective 関数の定義を次に示す．

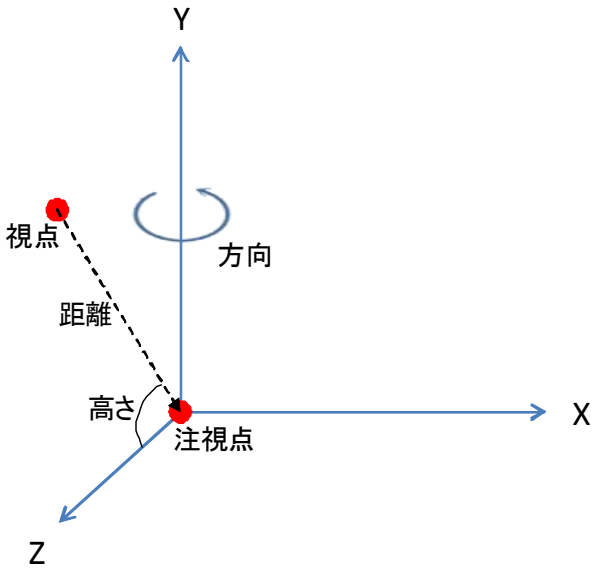
```
void gluPerspective(GLdouble fovy,  GLdouble aspect,  GLdouble znear,  GLdouble zfar)
```

gluPerspective 関数は 4 つの引数をとり，戻り値はない．gluPerspective 関数の引数を次に示す．

引数名	型	引数の意味	初期値
fovy	GLdouble	ビューボリュームの上下の開き角	40
aspect	GLdouble	投影面の幅と高さの比率	幅と高さの比
znear	GLdouble	視点から投影面までの距離	0.1
zfar	GLdouble	視点からビューボリュームの底面までの距離	10000

4.2.2　ズームイン・ズームアウト

OpenGL では，注視点の位置，注視点に対する距離，高さと方向によって描画空間の見え方を変化させる．本システムにおける視点の概念を次に示す．



ズームイン・ズームアウトは，注視点と視点の距離を変更することで行う．注視点と視点の距離を変更するには, glTranslated 関数を使用する. glTranslated 関数の定義を次に示す.

```
void glTranslated(GLdouble x, GLdouble y, GLdouble z)
```

glTranslated 関数は 3 つの引数をとり, 戻り値はない. glTranslated 関数の引数を次に示す.

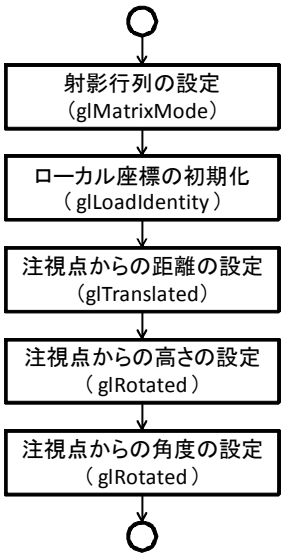
引数名	型	引数の意味	初期値
x	GLdouble	x 方向の移動量	0
y	GLdouble	y 方向の移動量	0
z	GLdouble	z 方向の移動量	100

ズームイン・アウトでは，glTranslated 関数の引数 z を変更することで行う．
また，ズームイン・アウトは，マウスホイールからとメニューバーからの 2 つの操作を実装する．マウスホイールからの操作の場合，PictureBox 上でのマウスホイールイベントを検知し，ズームインは上向きの回転で行い，ズームアウトは下向きの回転で行う．その際の z については，"1/32"ずつ加算および引算するように設定する．一方，メニューバーからの操作の場合の z については，"10"ずつ加算および引算するように設定する．

4.2.3 3Dビュー

3D ビューにおける視点の位置を変更は，注視点と視点の距離，角度と方向を変更することで行う．注視点と視点の距離を変更するには，ズームイン・アウトと同様に glTranlated

関数を使用する．注視点と視点の角度と方向を変更には，glRotated 関数を使用する．3D ビューにおける視点の位置の変更手順を次に示す．



まず，glMatrixMode 関数を使用して，射影行列を設定する．視点の位置を変更する場合，glMatrixMode 関数の引数には，「GL_MODELVIEW」を指定する．次に，glLoadIdentity 関数を使用して，ローカル座標を初期化する．最後に，glTranslated 関数と glRotated 関数を使用して，視点の位置を設定する．

glRotated 関数の定義を次に示す．

```
void glRotated(GLdouble angle, GLdouble x, GLdouble y, GLdouble z)
```

glRotated 関数は 4 つの引数をとり，戻り値はない．glRotated 関数の引数を次に示す．

引数名	型	引数の意味	初期値(高さ)	初期値（方向）
angle	GLdouble	回転角度	0.0	0.0
x	GLdouble	回転軸の x 方向のベクトル	1.0	0.0
y	GLdouble	回転軸の y 方向のベクトル	0.0	1.0
z	GLdouble	回転軸の z 方向のベクトル	0.0	0.0

(1) マウス操作によるカメラ位置の移動

マウス操作によるカメラ位置の移動では，平行移動については，マウスの左ボタンを押しながらマウスカーソルを移動することで行う．回転移動については，マウスの右ボタンを押しながらマウスカーソルを移動することで行う．

カメラの平行移動では，PictureBox 上でのマウスの押下を検知し，マウスの移動時にマウスの移動量を glTranslated 関数の x と y に設定する．

カメラの回転移動では，平行移動と同様に取得したマウスの移動量から角度と方向を算出し，glRotated 関数の angle に値を設定する．

(2) メニューバーからの視点の位置の設定

本システムでは，マウス操作以外にメニューバーからも視点の位置を変更できるようにする．本システムでは，正面図，上面図，下面図，側面図，背面図とデフォルトの戻すを実装する．各視点の実装方法について次に示す.

正面図

正面図では，z 軸の正方向から xy 平面の方向に視点を設定する．具体的には，glTranslate 関数の x と y の値と glRotated 関数の angle の値を全て「0」とする．

上面図

上面図では，y 軸の正方向から xz 平面の方向に視点を設定する．具体的には，glTranslate 関数の x と y の値を「0」，1 つ目の glRotated 関数の angle の値を「90」，2 つ目の glRotated 関数の angle の値を「0」とする．

下面図

下面図では，y 軸の負方向から xz 平面の方向に視点を設定する．具体的には，glTranslate 関数の x と y の値を「0」，1 つ目の glRotated 関数の angle の値を「-90」，2 つ目の glRotated 関数の angle の値を「0」とする．

側面図

側面図では，x 軸の正（負）方向から yz 平面の方向に視点を設定する．具体的には，glTranslate 関数の x と y の値を「0」，1 つ目の glRotated 関数の angle の値を「0」，2 つ目の glRotated 関数の angle の値を「-90」（負方向），「90」（正方向）とする．

背面図

背面図では，z 軸の負方向から xy 平面の方向に視点を設定する．具体的には，glTranslate 関数の x と y の値を「0」，1 つ目の glRotated 関数の angle の値を「0」，2 つ目の glRotated 関数の angle の値を「180」とする．

デフォルトに戻す

デフォルトに戻すでは，視点の位置を初期状態にする．具体的には，glTranslate 関数の x と y の値を「0」，z の値を「100」，glRotated 関数の angle の値を共に「0」とする．

4.2.4　モデル形状の切替

モデル形状の切替では，描画したモデルをサーフェスモデルとワイヤーフレームモデルに切り替えする．モデル形状の切替では，glPolygonMode 関数を使用し，各モデルの描画前に設定する．glPolygonMode 関数の定義を次に示す．

```
void glPolygonMode(GLenum face, GLenum mode)
```

glPolygonMode 関数は 2 つの引数をとり，戻り値はない．glPolygonMode 関数の引数を次に示す．

引数名	型	引数の意味	初期値(角度)
face	GLenum	多角形の向き	GL_FRONT_AND_BACK
mode	GLenum	多角形のラスタライズ方法	GL_FILL

サーフェスモデルとワイヤーフレームモデルを切り替えるには，glPolygonMode 関数の mode の値を変更する．具体的には，mode の値が「GL_FILL」の場合，サーフェスモデルで描画され，「GL_LINE」の場合，ワイヤーフレームモデルで描画される．

4.3　テーブル要素の設定

本章では，モデルを配置するレイヤ，モデルの色，線種，線幅，そして，使用する文字フォントの設定方法について説明する．

4.3.1　レイヤ

本節では，レイヤの設定方法について説明する．レイヤは，モデルを配置する層であり，モデルは必ずレイヤに配置する必要がある．本システムでは，各モデルがレイヤコードを保持することで，モデルとレイヤを関連付ける．レイヤコードは 1 から始まるシーケンシャル番号とする．また，レイヤコードは，レイヤ名を保持する．レイヤコードとレイヤ名の関係を次に示す．

レイヤコード	レイヤ名
1	1 番目に指定したレイヤ名
2	2 番目に指定したレイヤ名
…	…

256	256 番目に指定したレイヤ名

4.3.2 色の設定

本節では，色の設定方法について説明する．本システムでは，色コードをモデルのパラメータとして保持することで，色とモデルを関連付ける．色は，列挙型で定数名と色コードを関連付ける．また，色コードは，1 から始まるシーケンシャル番号とする．本システムにおける色コードと色の関係を次に示す.

定数名	色コード	色名
BLACK	1	黒
RED	2	赤
GREEN	3	緑
BLUE	4	青
YELLOW	5	黄
MAGENTA	6	マジェンタ
CYAN	7	シアン
WHITE	8	白
DEEPPINK	9	牡丹
BROWN	10	茶
ORANGE	11	橙
LIGHTGREEN	12	薄緑
LIGHTBLUE	13	薄青
LAVENDER	14	青紫
LIGHTGRAY	15	明灰
DARKGRAY	16	暗灰

OpenGL では，glColor3d や glColor4d などの glColor*関数を使用することで図形の色を設定することができる．本システムでは，glColor3d 関数を使用して図形の色を設定する．glColor3d 関数の定義を次に示す.

```
void glColor3d(GLdouble r, GLdouble g, GLdouble b)
```

glColor3d 関数は，3 個の引数をとり，戻り値はない．glColor3d 関数の引数を次に示す.

引数名	引数の意味
r	赤色の光の強さ

g	緑色の光の強さ
b	青色の光の強さ

r，g と b の値の範囲は 0 から 1 に設定する．例えば，黄色を設定する場合，r に 1.0，g に 1.0，b に 0.0 を指定する．

4.3.3 線種

本節では，線種の設定方法について説明する．本システムでは，線種コードをモデルのパラメータとして保持することで，線種とモデルを関連付ける．線種コードは，1 から始まるシーケンシャル番号とする．本システムにおける線種コードと線種の関係を次に示す．

定数名	線種コード	線種名
CONTINUOUS	1	実線
DASHED	2	破線
DASHEDSPACED	3	跳び破線

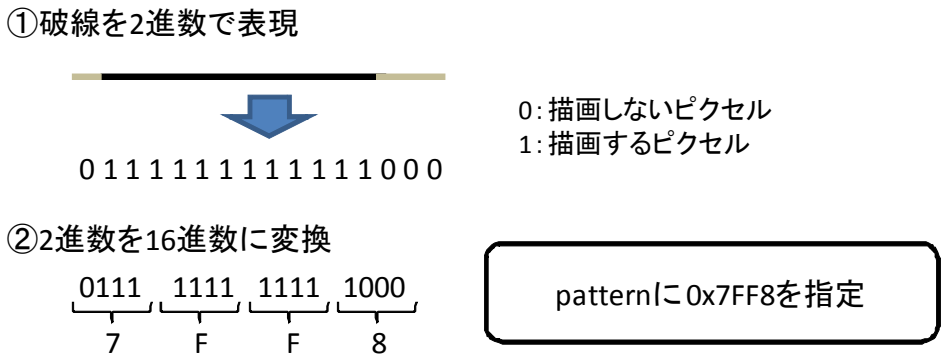
OpenGL では，glLineStipple 関数を使用することで図形の線種を設定する．glLineStipple 関数の定義を次に示す．

```
void glLineStipple(GLint factor, GLushort pattern)
```

glLineStipple 関数は，2 個の引数をとり，戻り値はない．glLineStipple 関数の引数を次に示す．

引数名	引数の意味
factor	破線の拡大率
pattern	破線のパターン

破線のパターンは，16 進数で指定する．破線のパターンの指定例を次に示す．



glLineStipple 関数の引数には，2 進数に変換した際に，描画するピクセルが 1，描画しないピクセルが 0 となるような 16 進数の値を指定する．

4.3.4 線幅

本節では，線幅の設定方法について説明する．本システムでは，線幅コードをモデルのパラメータとして保持することで，線幅とモデルを関連付ける．線幅コードは，1 から始まるシーケンシャル番号とする．本システムにおける線幅コードと線幅の関係を次に示す．

定数名	線幅コード	線幅
W013	1	0.13
W018	2	0.18
W025	3	0.25
W035	4	0.35
W050	5	0.5
W070	6	0.7
W100	7	1.0
W140	8	1.4
W200	9	2.0

OpenGL では，glLineWidth 関数を使用することで図形の線幅を設定する．glLineWidth 関数の定義を次に示す．

```
void glLineWidth(GLfloat width)
```

glLineWidth 関数は，1 個の引数をとり，戻り値はない．glLineWidth 関数の引数を次に示す．

引数名	引数の意味

width	線の幅

線の幅はピクセル単位で指定する．線の幅を 1 ピクセルに設定する場合は，width に 1.0 を指定する．

4.3.5　文字フォント

本節では文字フォントの設定方法について説明する．本システムでは，文字フォントコードをモデルのパラメータとして保持することで，文字フォントとモデルを関連付ける．文字フォントコードは 1 から始まるシーケンシャル番号とする．本システムにおける文字フォントコードと文字フォントの関係を次に示す．

文字フォントコード	文字フォント名
1	1 番目に指定した文字フォント名
2	2 番目に指定した文字フォント名
…	…
1024	1024 番目に指定した文字フォント名

文字フォントについては，PC によって使用できるフォントが異なる．そのため，本システムでは，使用できる文字フォントを取得する．使用できる文字フォントの取得方法を次に示す．

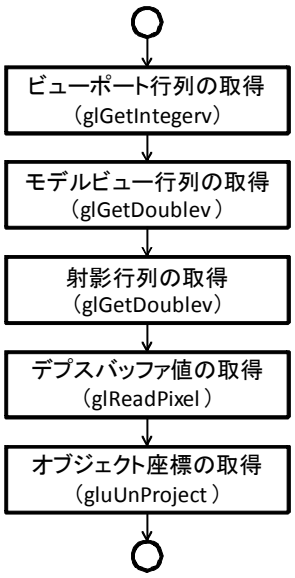
```
// フォントリストの作成
System::Drawing::Text::InstalledFontCollection^ ifcFont =
    gcnew   System::Drawing::Text::InstalledFontCollection();
cli::array<FontFamily^> ^cFontList = ifcFont->Families;
for each(FontFamily^ f in cFontList){
    tscbFont->Items->Add(f->Name);
}
```

4.4　ウィンドウ座標からオブジェクト座標への変換

4.4.1　ウィンドウ座標からオブジェクト座標への変換

マウス操作でモデル空間上にモデルを描画するには，PictureBox（ウィンドウ座標）上の 2 次元座標をモデル空間（オブジェクト座標）上の 3 次元座標に変換する必要がある．本システムでは，glGetIntegerv 関数，glGetDoublev 関数，glReadPixel 関数と gluUnProject 関数を

使用してウィンドウ座標をオブジェクト座標に変換する．ウィンドウ座標からオブジェクト座標への変換の手順を次に示す．



まず，glGetInteferv 関数を使用して，現在のビューポート行列を，glGetdoublev 関数を使用して，現在のモデルビュー行列（視点の位置）を，そして，glGetdoublev 関数を使用して，現在の射影行列を取得する．次に，glReadPixel 関数を使用して，現在のデプスバッファの値を取得する．最後に，これらの情報とウィンドウ座標の x，y の値から gluUnProject 関数を使用して，オブジェクト座標を算出する．

glGetIntegerv 関数の定義を次に示す．

```
void glGetIntegerv(GLenum pname, GLint* params)
```

glGetIntegerv 関数は 2 つの引数をとり，戻り値はない．glGetIntegerv 関数の引数を次に示す．

引数名	型	引数の意味	初期値
pname	GLenum	取得するパラメータの種類	GL_VIEWPORT
params	GLint*	取得するパラメータのポインタ	

ビューポート行列を取得するには，glGetIntegerv 関数の 1 つ目の引数に「GL_VIEWPORT」を指定する．また，取得するパラメータは 4*1 の行列であるため，2 つ目の引数には，要素数 4 の配列のポイントを指定する．

glGetDoublev 関数の定義を次に示す．

```
void glGetDoublev(GLenum pname, GLdouble* viewport)
```

glGetDoublev 関数は 2 つの引数をとり，戻り値はない．glGetDoublev 関数の引数を次に示す．

引数名	型	引数の意味	初期値
pname	GLenum	取得するパラメータの種類	GL_MODELVIEW_MATRIX
params	GLdouble*	取得するパラメータのポインタ	

モデルビュー行列を取得するには，glGetDoublev 関数の 1 つ目の引数に「GL_MODELVIEW」を指定し，射影行列を取得するには，「GL_PROJECTION」を指定する．また，取得するパラメータは 4*4 の行列であるため，2 つ目の引数には，要素数 16 の配列のポイントを指定する．

glReadPixels 関数の定義を次に示す．

```
void glReadPixels(GLint x, GLint y, GLsizei width, GLsizei height,
                  GLenum format, GLenum type, GLvoid* pixels)
```

glReadPixels 関数は 6 つの引数をとり，戻り値はない．glReadPixels 関数の引数を次に示す．

引数名	型	引数の意味	初期値
x	GLint	ウィンドウ座標の x 値	
y	GLint	ウィンドウ座標の y 値	
width	GLsizei	ピクセル方形の領域の幅	1
height	GLsizei	ピクセル方形の領域の高さ	1
format	GLenum	データフォーマット	GL_DEPTH_COMPONENT
type	GLenum	データ型	GL_FLOAT
pixels	GLvoid*	取得するデータのポインタ	

デプスバッファを取得するには，glReadPixels 関数の 5 つ目の引数に「GL_DEPTH_COMPONENT」を指定する．また，取得するパラメータの型は float 型であるため，6 つ目の引数には，「GL_FLOAT」を指定する．

gluUnProject 関数の定義を次に示す．

```
int gluUnProject(GLdouble x, GLdouble y, GLdouble z, GLdouble modelview[16]
                 GLdouble project[16], GLdouble viewport[4],
                 GLdouble* objX, GLdouble* objY, GLdouble* objZ)
```

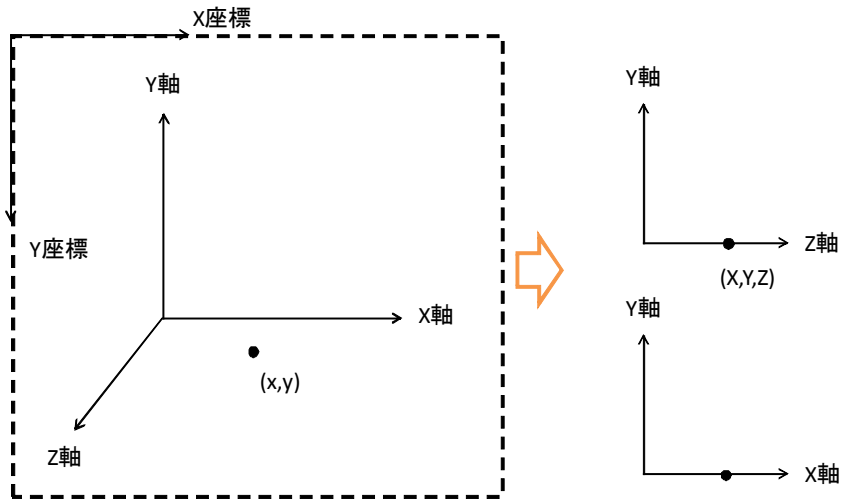
gluUnProject 関数は 9 つの引数をとり，戻り値は int 型である．gluUnProject 関数の引数を次に示す．

引数名	型	引数の意味
x	GLdouble	ウィンドウ座標の x 値
y	GLdouble	ウィンドウ座標の y 値
z	GLdouble	ウィンドウ座標の z 値
modelview	GLdouble	モデルビュー行列
project	GLdouble	射影行列
viewport	GLdouble	ビューポート行列
objX	GLdouble*	オブジェクト座標の x 値
objY	GLdouble*	オブジェクト座標の y 値
objZ	GLdouble*	オブジェクト座標の z 値

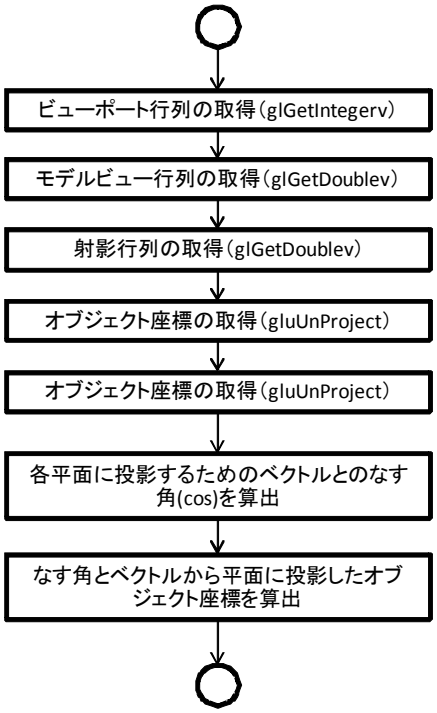
オブジェクト座標は，これまでに取得した各行列，デプスバッファとウィンドウ座標から gluUnProject を使用して算出する．

4.4.2　本システムにおけるオブジェクト座標への変換

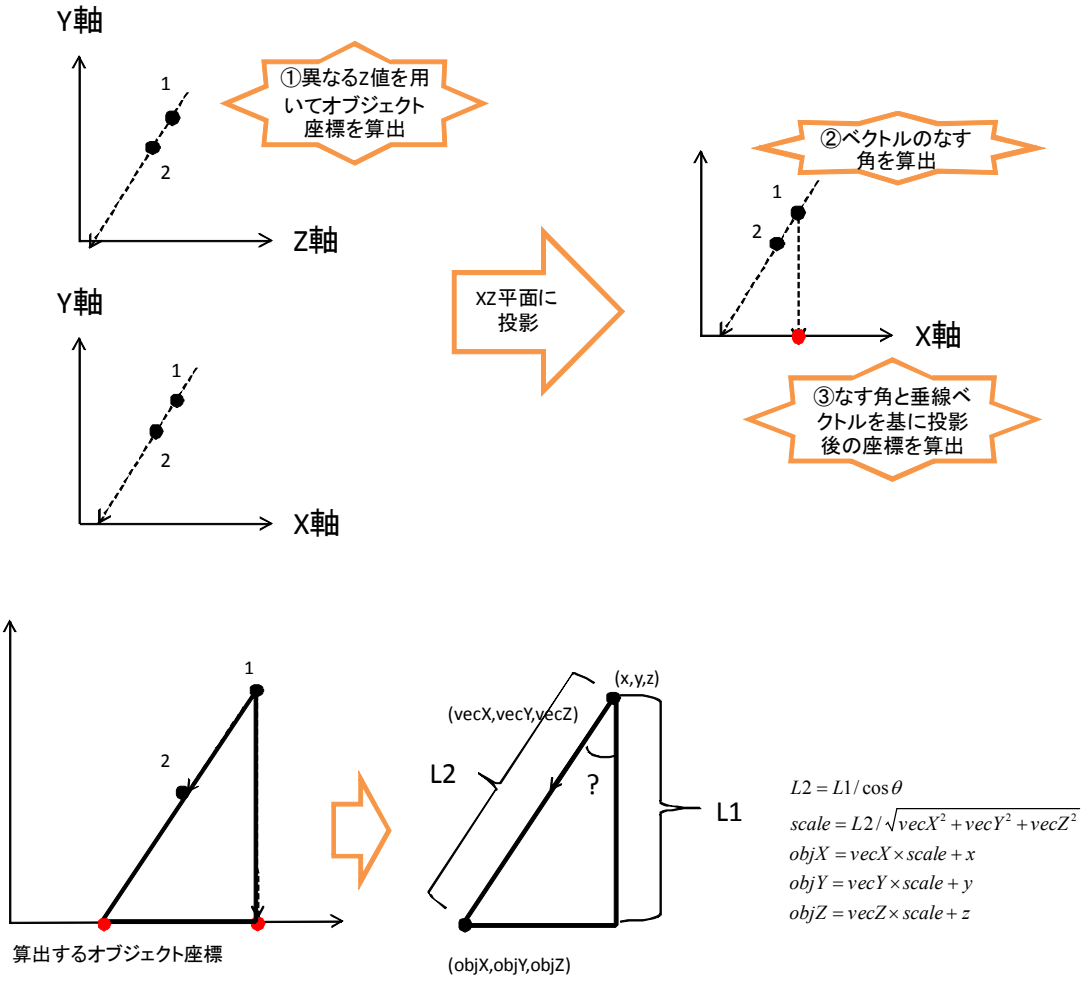
本システムでは，各モデルを指定もしくは任意の平面上に描画できるようにする．そのため，平面上に投影したオブジェクト座標を算出する．本システムで算出するオブジェクト座標の例を次に示す．



本システムにおけるウィンドウ座標からオブジェクト座標の変換手順を次に示す.



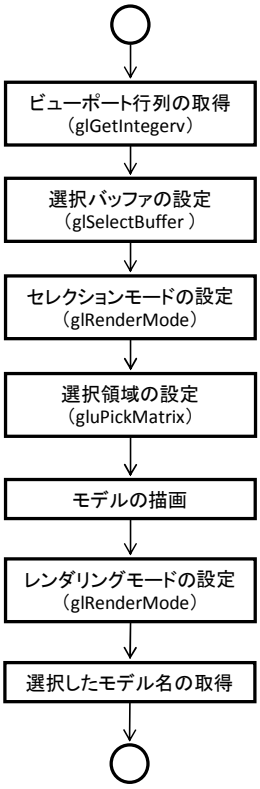
まず，glGetInteferv 関数を使用して，現在のビューポート行列を，glGetdoublev 関数を使用して，現在のモデルビュー行列（視点の位置）を，そして，glGetdoublev 関数を使用して，現在の射影行列を取得する．次に，異なるデブスバッファ値を用いてウィンドウ座標の x，y の値から gluUnProject 関数を使用して，オブジェクト座標を 2 つ算出する．最後に，算出した 2 つのオブジェクト座標を用いて，ある平面に投影したオブジェクト座標を算出する．具体的には，まず，OpenGL で算出したオブジェクト座標に対してデプスバッファの値を変更し，オブジェクト座標のベクトルを算出する．次に，オブジェクト座標とオブジェクト座標から平面に垂直線を下した際の座標とのベクトルを算出する．最後に，ベクトル間のなす角を算出し，平面に投影したオブジェクト座標を算出する．（下記の図参照）



本システムでは，投影する平面について，XY 平面，XZ 平面，YZ 平面と任意を選択可能にし，任意の場合は，各々の平面に投影した際のベクトルのなす角(cos)が最大となる平面に投影する．

4.5　マウス操作によるモデルの選択

マウス操作でモデル空間上のモデルを選択するには，PictureBox（ウィンドウ座標）上の 2 次元座標からモデル空間（オブジェクト座標）上のモデルの情報を把握する必要がある．OpenGL では，各モデルがどの深さに描画されているを記録している．本システムでは，この情報を利用してモデル空間上のモデルを選択する．モデルの選択では，モデルの描画処理に glGetIntegerv 関数，glSelectBuffer 関数，glRenderMode 関数，gluPickMatrix 関数を追加する必要がある．本システムのモデルの選択の手順を次に示す．



まず，glGetInteferv 関数を使用して，現在のビューポート行列を取得する．次に，glSelectBuffer 関数を使用して，モデルの選択に使用するバッファを設定する．そして，glRenderMode 関数を使用して，選択モードに移行し，gluPickMatrix 関数を使用して，選択領域を選択する．最後に，モデルの描画後，再び，glRenderMode 関数を使用してレンダリングモードに移行し，選択したモデル数を取得する．そして，選択したモデル数を基に選択したモデル名を取得する．ただし，モデルの描画時に，glLoadName 関数を使用して，モデルを識別するための名前を設定する必要がある．

glRenderMode 関数の定義を次に示す．

```
int glRenderMode(GLenum mode)
```

glRenderMode 関数は 1 つの引数をとり，戻り値は int 型である．glRenderMode 関数の引数を次に示す．

引数名	型	引数の意味	初期値
mode	GLenum	レンダリングのタイプ	GL_SELECt

glSelectBuffer 関数の定義を次に示す．

```
int glSelectBuffer(GLsizei size, GLuint* buffer)
```

glSelectBuffer 関数は 2 つの引数をとり，戻り値は int 型である．glSelectBuffer 関数の引数を次に示す.

引数名	型	引数の意味
size	GLsizei	要素数
buffer	GLuint	選択したオブジェクトの名前などを格納する配列

gluPickMatrix 関数の定義を次に示す.

```
int gluPickMatrix(GLdouble x, GLdouble y, GLdouble dx, GLdouble dy, GLint *viewport)
```

gluPickMatrix 関数は 2 つの引数をとり，戻り値は int 型である．gluPickMatrix 関数の引数を次に示す.

引数名	型	引数の意味
x	GLdouble	ウィンドウ座標（X 座標）
y	GLdouble	ウィンドウ座標（Y 座標）
dx	GLdouble	選択する領域の大きさ
dy	GLdouble	選択する領域の大きさ
viewport	GLint*	ビューポート

glLoadName 関数の定義を次に示す.

```
void glLoadName(GLuint name)
```

glLoadName 関数は 1 つの引数をとり，戻り値ない. glLoadName 関数の引数を次に示す.

引数名	型	引数の意味
name	GLuint	モデル名

選択したモデルのモデル名は，glSelectBuffer 関数で設定したバッファと glRenderMode 関数の戻り値によって取得したモデル数を基に取得する．選択したモデル名の取得方法のサンプルを次に示す.

```
int wglRender::SelectHits(GLuint hits,GLuint *buf)
{
	unsigned int i, j;

	GLuint hit_name = -1;
	float depth_min = 10.0f;
	float depth_1 = 1.0f;
	float depth_2 = 1.0f;
	GLuint depth_name;
	GLuint *ptr;

	if(hits <= 0)
		return -1;

	ptr = (GLuint*)buf;
	for(i = 0;i < hits;i++)
	{
		depth_name = *ptr;
		ptr++;
		depth_1 = (float)*ptr/0x7fffffff;
		ptr++;
		depth_2 = (float)*ptr/0x7fffffff;
		ptr++;

		if(depth_min > depth_1)
		{
			depth_min = depth_1;
			for(j = 0;j < depth_name;j++)
			{
				hit_name = *ptr;
				ptr++;
			}
		}
		else
		{
			for(j = 0;j < depth_name;j++)
			{
				ptr++;
			}
		}
	}
	return hit_name;
}
```

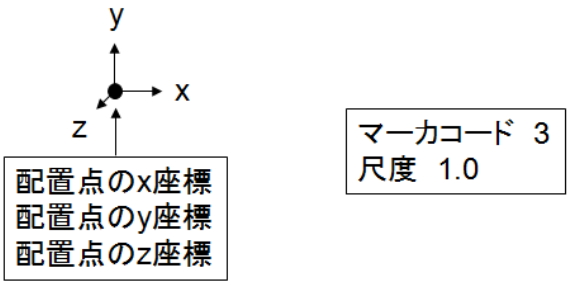
4.6　モデルの描画

4.6.1　点マーカ

本節では，点マーカの描画方法について説明する．

幾何情報

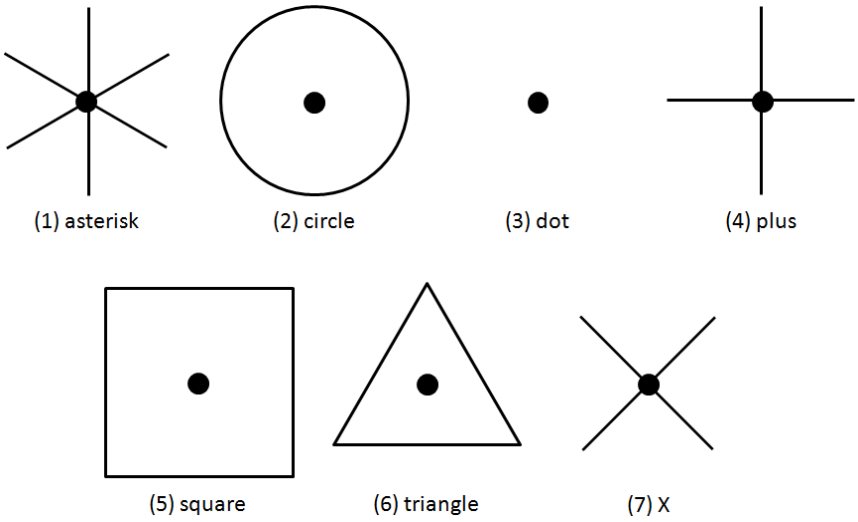
点マーカの幾何情報を次に示す.



本システムでは，マーカコードに応じて asterisk，circle，dot，plus，square，triangle と X の 7 種類の点マーカを実装する．マーカコードと点マーカの種類の対応を次に示す.

マーカコード	点マーカの種類
1	asterisk
2	circle
3	dot
4	plus
5	square
6	triangle
7	X

各種点マーカの概要を次に示す.



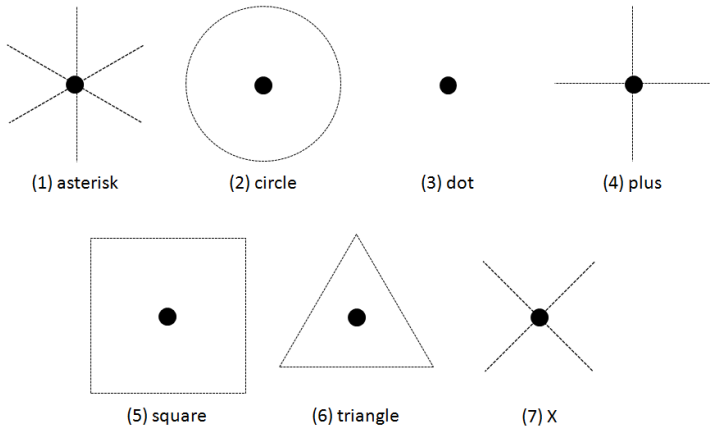
上図の黒点は，各種点マーカの配置座標であり，配置座標を基に図形の頂点を算出し，図形を描画する.

位相情報

本システムでは，点マーカの位相情報として，頂点に関する情報を保持する．点マーカの位相を次に示す．

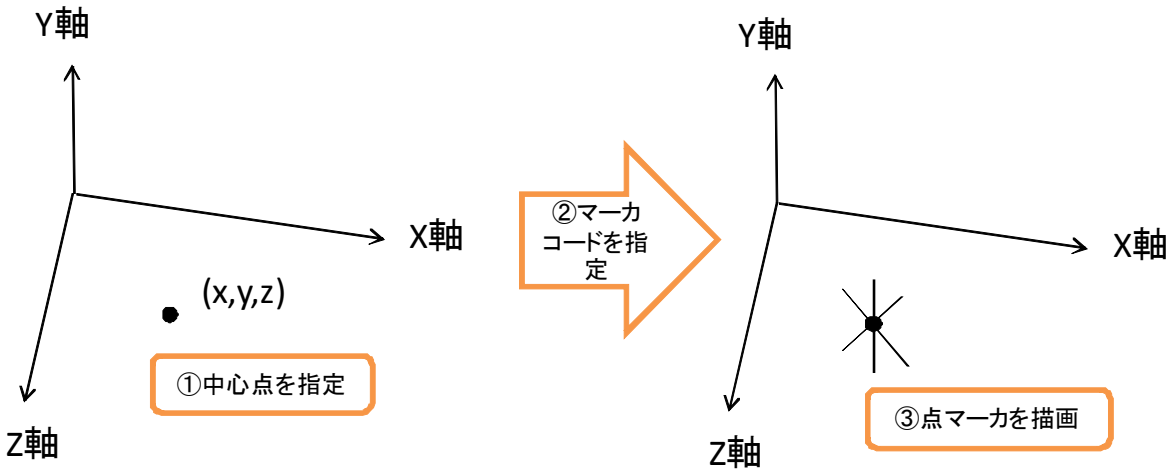
パラメータ	型	説明
m_hVertex	Hashtable	頂点に関する情報

本システムでは，配置座標を点マーカの頂点とする．点マーカの位相情報を次に示す．



描画方法

本システムでは，モデル空間上においてマウス操作で点マーカを描画する．点マーカの描画手順を次に示す．



本システムでは，まず，配置点（X ,Y ,Z）をマウス操作で指定する．次に，ダイアログ上でマーカコードを指定する．最後に，マーカコードに応じた点マーカを描画する．

点マーカを描画するためには，OpenGL の glVertex3d 関数を使用する．glVertex3d 関数の

定義を次に示す.

```
void glVertex3d(GLdouble x, GLdouble y, GLdouble z)
```

glVertex3d 関数は，3 つの引数をとり，戻り値はない．glVertex3d 関数の引数を次に示す.

引数名	引数の意味
x	配置点の x 座標
y	配置点の y 座標
z	配置点の z 座標

本システムでは，glVertex3d 関数を使用して，7 種類の点マーカを描画する．配置点の X 座標，Y 座標，Z 座標，尺度とマーカコードの大きさを設定して点マーカを描画する．各点マーカの描画方法を次に示す.

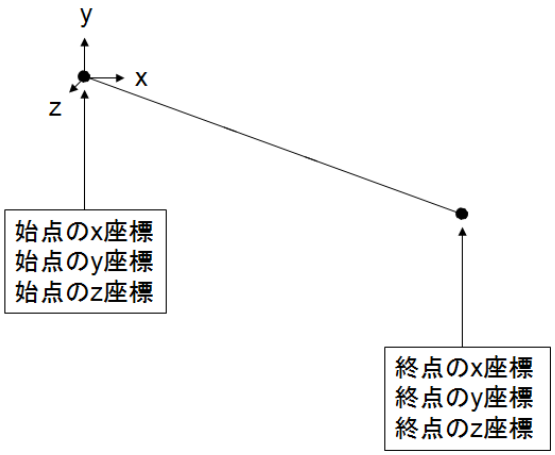
マーカコード	マーカ名	実装方法
1	asterisk	3 本の線
2	circle	折線に近似した円
3	dot	1 つの点
4	plus	2 本の線
5	square	1 本の線
6	triangle	1 本の線
7	X	2 本の線

4.6.2 線分

本節では，線分の描画方法について説明する.

幾何情報

線分の幾何情報を次に示す.

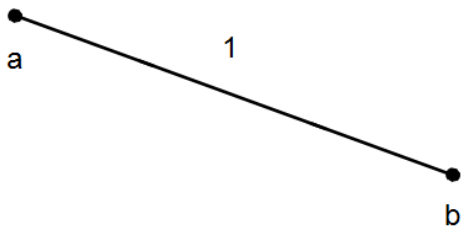


位相情報

本システムでは，線分の位相情報として，頂点と稜線に関する情報を保持する．線分の位相情報を次に示す．

パラメータ	型	説明
m_hEdge	Hashtable	稜線に関する情報

本システムでは，始点と終点の座標を線分の頂点とし，始点と終点からなる線分を稜線とする．線分の位相情報を次に示す．

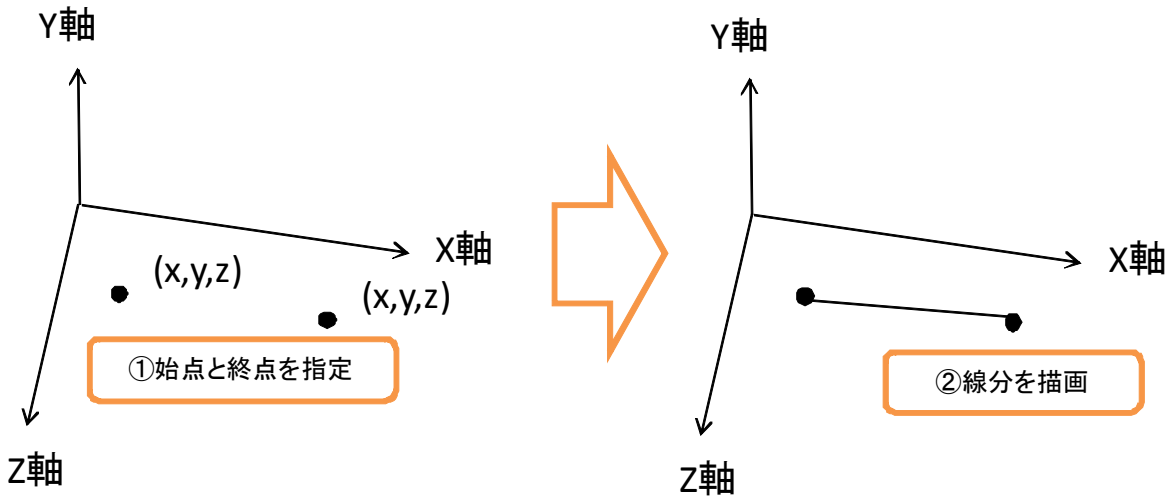


線分の始点座標と終点座標を位相情報の頂点に定義し，2 つの頂点から稜線を定義する．線分の稜線と頂点の対応を次に示す．

稜線番号	頂点番号	
	始点	終点
1	a	b

描画方法

本システムでは，モデル空間上においてマウス操作で線分を描画する．線分の描画手順を次に示す．



本システムでは，まず，始点（X ,Y ,Z）をマウス操作で指定する．次に，終点（X ,Y ,Z）をマウス操作で指定する．最後に，始点と終点を結ぶ線分を描画する．ただし，始点と終点の座標は，同一平面上に投影することとする．また，線分の描画の際，終点を指定するまでの間，線分のラバーバンドを描画する．線分のラバーバンドの描画では，マウスの移動イベントを検出し，移動する度にマウスの 3 次元座標を算出し，線分を描画する．

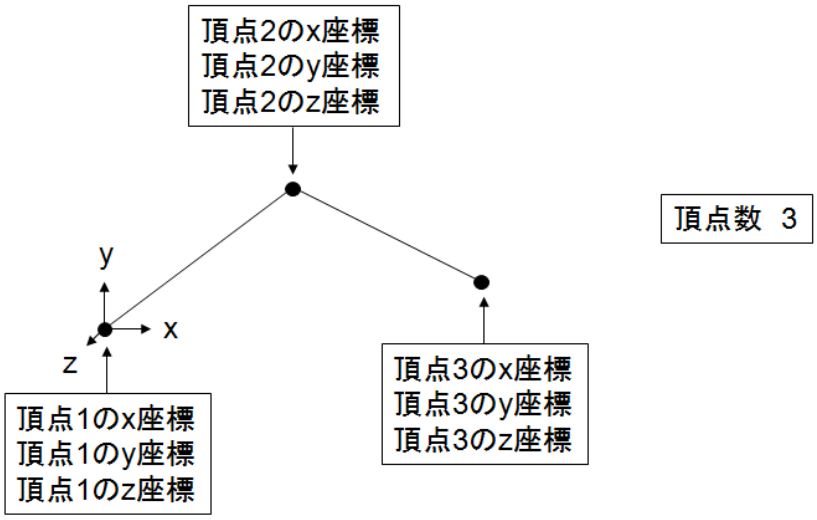
線分を描画するには，glVertex3d 関数を使用する．具体的には，線分のパラメータである始点と終点の X 座標，Y 座標と Z 座標をそれぞれ設定し，2 点間を結ぶ直線を引くことで描画する．

4.6.3　折線

本節では，折線の描画方法について説明する．

幾何情報

折線の幾何情報を次に示す．

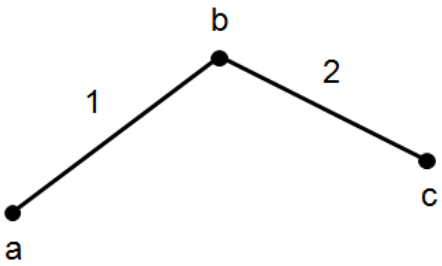


位相情報

本システムでは，折線の位相情報として，頂点と稜線に関する情報を保持する．折線の位相情報を次に示す．

パラメータ	型	説明
m_hEdge	Hashtable	稜線に関する情報

本システムでは，各頂点の座標を頂点とし，各頂点を結ぶ線分を稜線とする．折線の位相情報を次に示す．

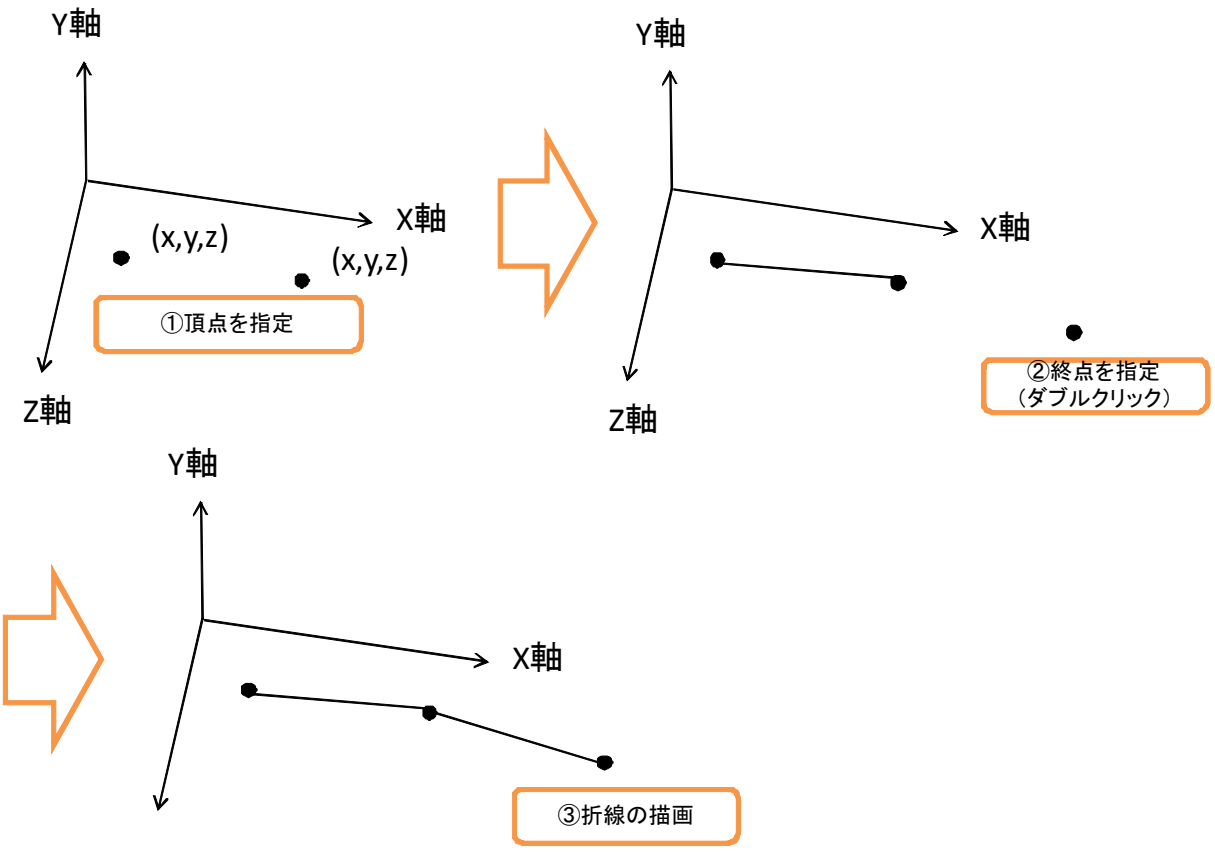


折線の各頂点の座標を位相情報の頂点に定義し，各頂点間を結ぶ稜線を定義する．折線の稜線と頂点の対応を次に示す．

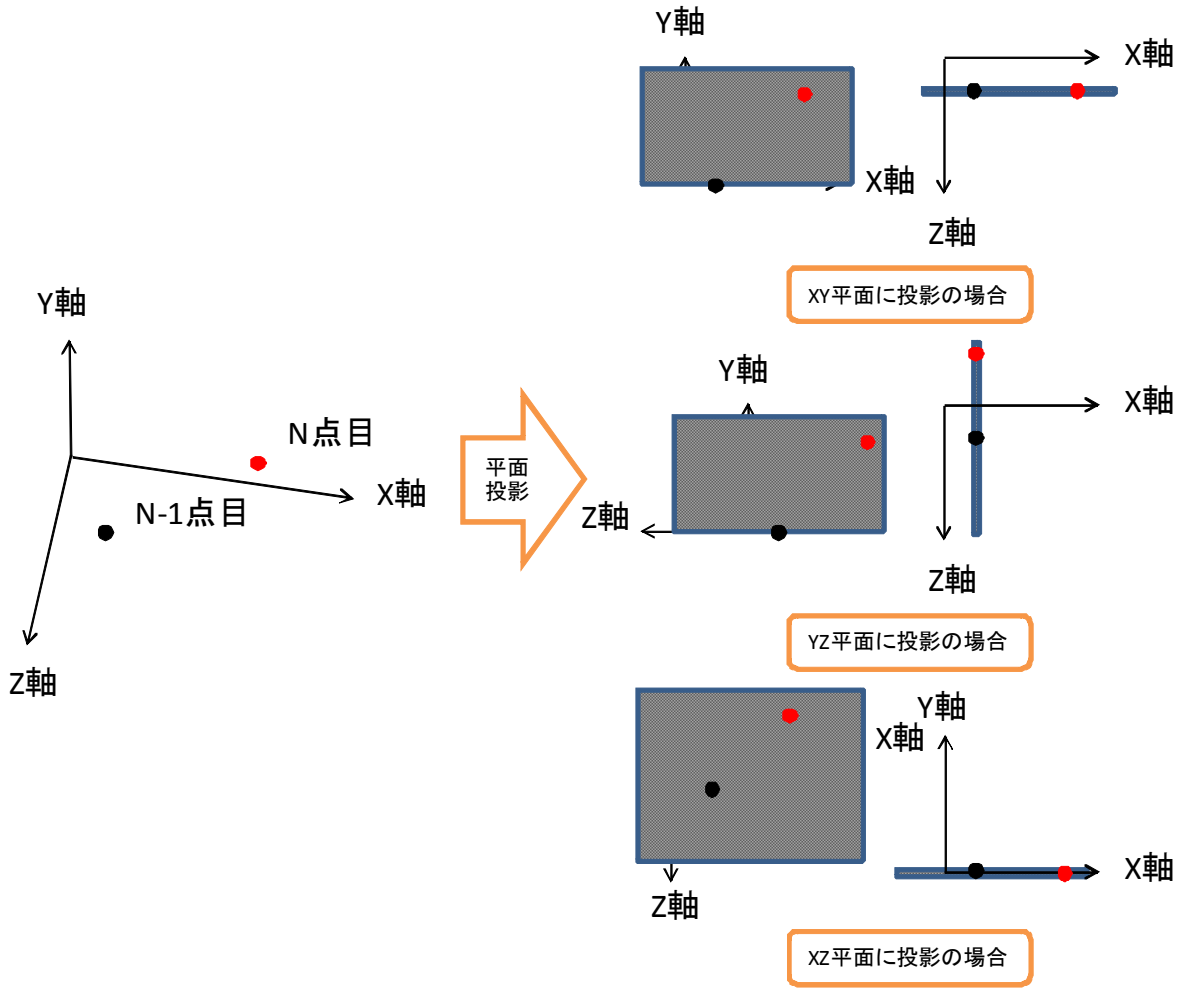
稜線番号	頂点番号	
	始点	終点
1	a	b
2	b	c

描画方法

本システムでは，モデル空間上においてマウス操作で折線を描画する．折線の描画手順を次に示す．



本システムでは，まず，各頂点（X ,Y ,Z）をマウス操作で指定する．次に，終点（X ,Y ,Z）をマウス操作（ダブルクリックイベントを使用）で指定する．最後に，各頂点を結ぶ線分を描画する．折線については，投影する平面を状況に応じて変更する．ただし，平面は，各座標軸からなる平面ではなく，１つ前の頂点の座標を通り，各座標軸から平面と平行な平面に投影する．折線における頂点座標の平面のイメージを次に示す．



また，折線の描画の際，折線のラバーバンド表示を行う．折線のラバーバンド表示では，確定した頂点座標の折線を描画し，現在，指定中の頂点は線分と同様に，マウスの移動イベントを検出し，移動する度にマウスの 3 次元座標を算出し，折線を描画する．

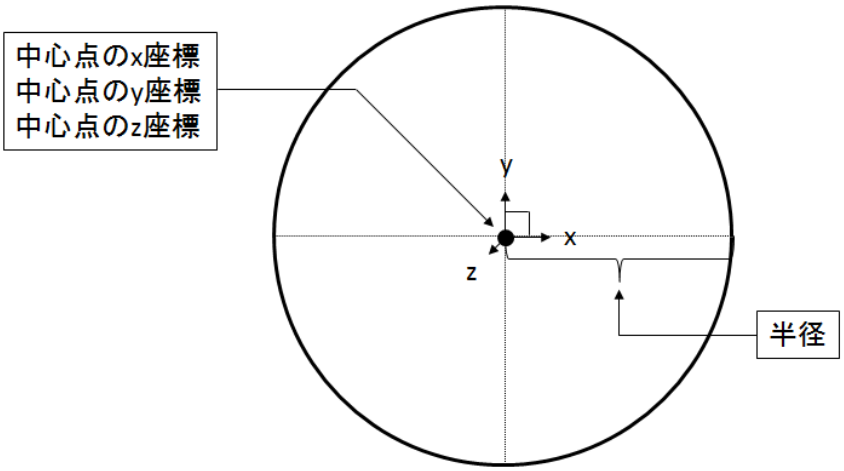
折線を描画するには，glVertex3d 関数を使用する．具体的には，折線は，頂点数と頂点数に応じた数の X 座標，Y 座標と Z 座標を設定し，各頂点間を結ぶ直線を引くことで描画する．

4.6.4　円

本節では，円の描画方法について説明する．

幾何情報

円の幾何情報を次に示す．

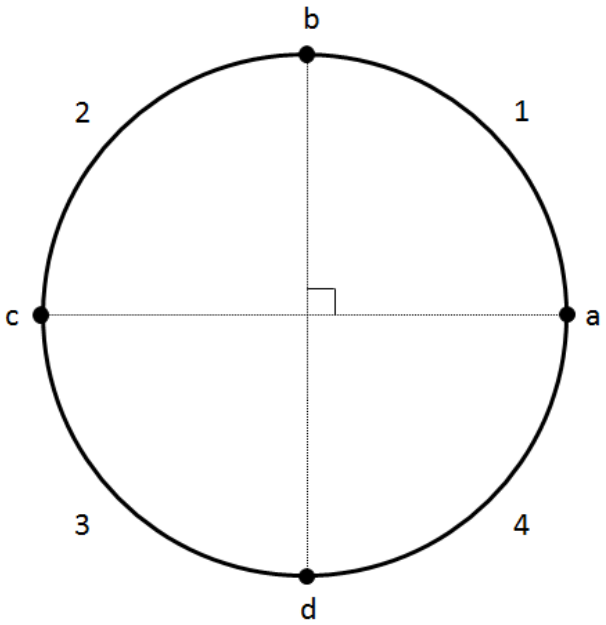


位相情報

本システムでは，円の位相情報として，頂点と稜線に関する情報を保持する．円の位相情報を次に示す．

パラメータ	型	説明
m_hEdge	Hashtable	稜線に関する情報

本システムでは，円周上に頂点を設定し，各頂点を結ぶ線分を稜線とする．円の位相情報を次に示す．



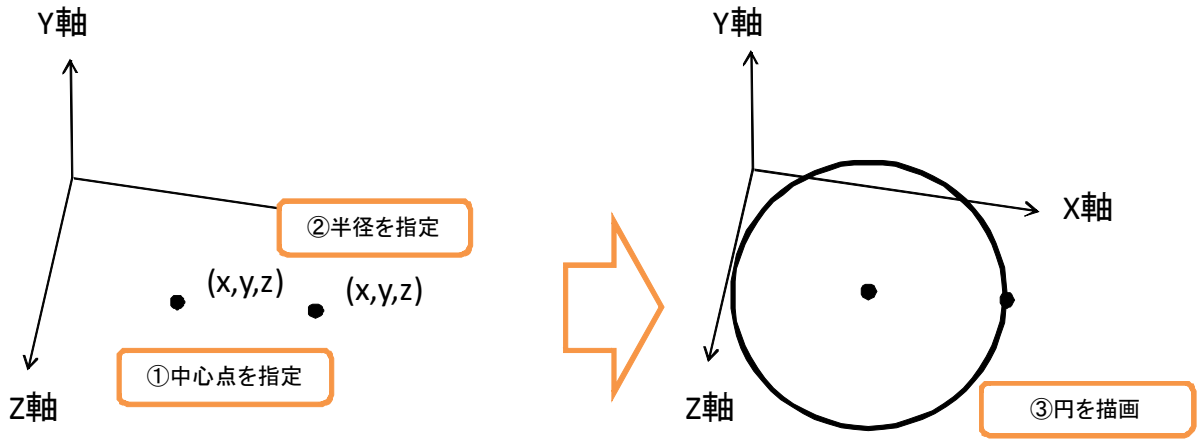
円の頂点は，始角から 90 度毎に頂点を定義し，各頂点間を結ぶ稜線を定義する．円の稜

線と頂点の対応を次に示す.

稜線番号	頂点番号	
	始点	終点
1	a	b
2	b	c
3	c	d
4	d	a

描画方法

本システムでは，モデル空間上においてマウス操作で中心点と半径を指定することで円を描画する．円の描画手順を次に示す.



本システムでは，まず，中心点（X ,Y ,Z）をマウス操作で指定する．次に，半径を指定するため，モデル空間上で点（X ,Y ,Z）をマウス操作で指定する．最後に，中心点と半径を指定するための点の距離を半径とする円を描画する．ただし，円の中心点と半径を表す点は同一平面上に投影するものとする．また，円の描画の際，円のラバーバンド表示を行う．円のラバーバンド表示では，中心点を指定後，半径を指定する際，マウスの移動イベントを検出し，円を描画する.

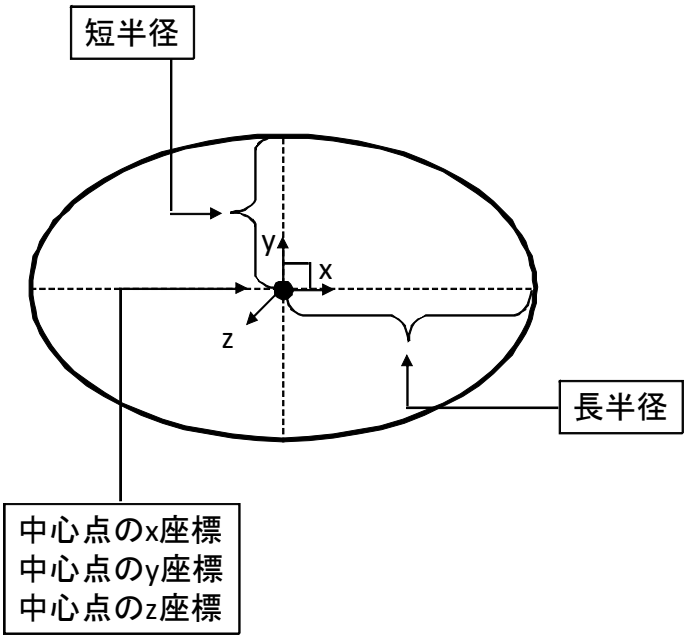
OpenGL には，円を描画するための関数が用意されていないため，円を描画するには，円の中心点と半径から円周上の頂点座標を算出し，折線に近似する．なお，円周上の頂点の数は 1080 とする.

4.6.5　楕円

本節では，楕円の描画方法について説明する.

幾何情報

楕円の幾何情報を次に示す.

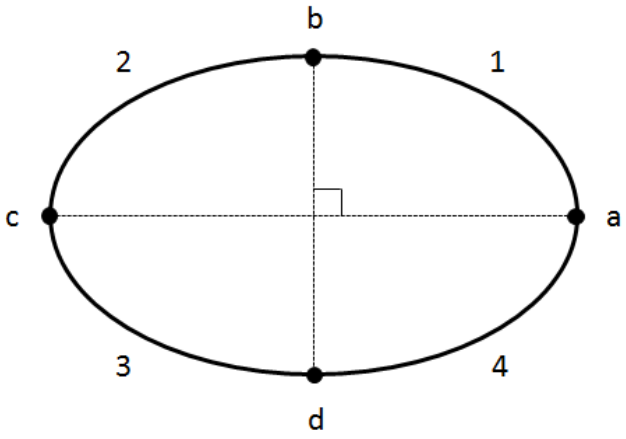


位相情報

本システムでは，楕円の位相情報として，頂点と稜線に関する情報を保持する．楕円の位相情報を次に示す.

パラメータ	型	説明
m_hEdge	Hashtable	稜線に関する情報

本システムでは，楕円の周上に頂点を設定し，各頂点を結ぶ線分を稜線とする．楕円の位相情報を次に示す.

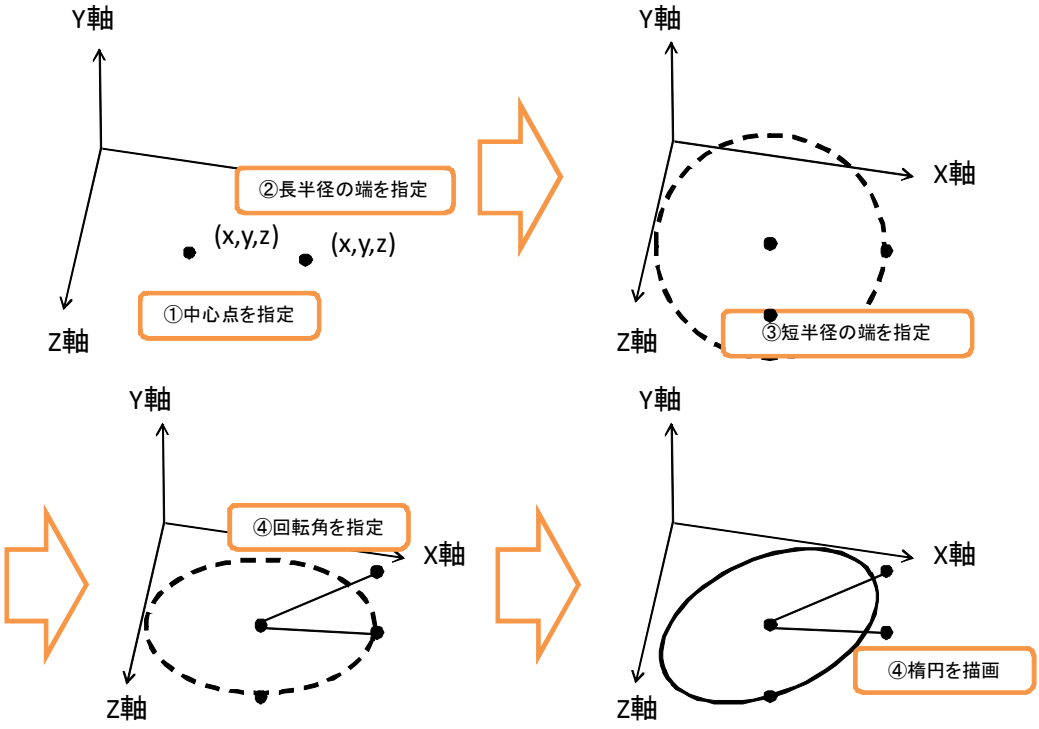


楕円の頂点は，始角から 90 度毎に頂点を定義し，各頂点間を結ぶ稜線を定義する．楕円の稜線と頂点の対応を次に示す．

稜線番号	頂点番号	
	始点	終点
1	a	b
2	b	c
3	c	d
4	d	a

描画方法

本システムでは，モデル空間上においてマウス操作で中心点，長半径と短半径を指定し，回転角を指定することで楕円を描画する．楕円の描画手順を次に示す．



本システムでは，まず，中心点（X ,Y ,Z）をマウス操作で指定する．次に，長半径と短半径を表す点をモデル空間上にマウス操作で指定する．そして，楕円の回転角を指定する．最後に，中心点，長半径と短半径を表す点と回転角の情報から楕円を描画する．ただし，楕円の中心点，長半径と短半径を表す点と回転角を表す点は同一平面上に投影するものとする．また，楕円の描画の際，楕円のラバーバンド表示を行う．楕円のラバーバンド表示では，中心点の指定後，マウスの移動を検出し，長半径を指定するまで，長半径を半径する円を描画する．次に，長半径を指定後，短半径を指定するまで，長半径と短半径から楕円を描画する．最後に，回転角を指定するまで回転した楕円を描画する．

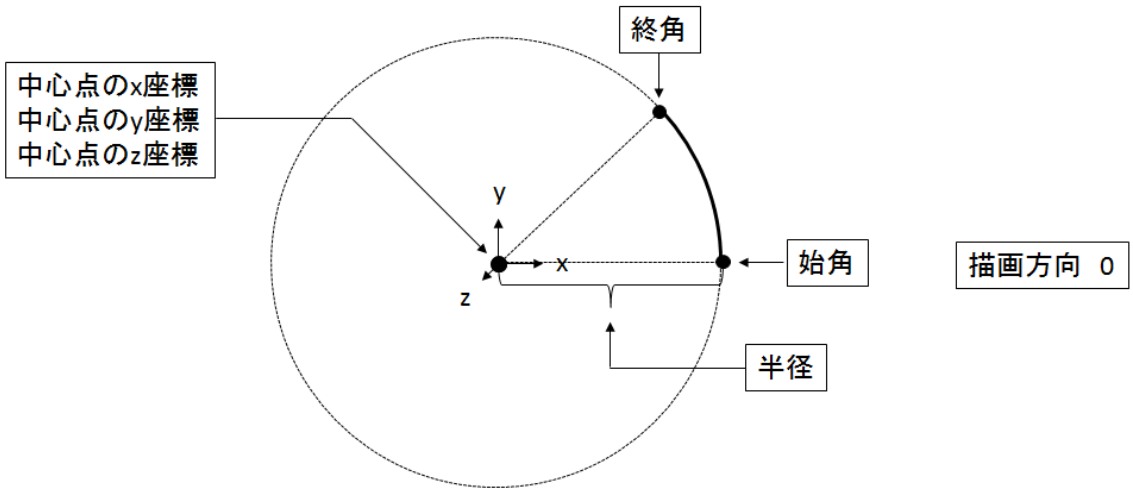
OpenGL には楕円を描画するための関数が用意されていないため，楕円を描画するには，円の場合と同様に，中心点，長半径，短半径と回転角から楕円の周上の頂点座標を算出し，折線に近似する．なお，楕円の周上の頂点の数は 1080 とする．

4.6.6 円弧

本節では，円弧の描画方法について説明する．

幾何情報

円弧の幾何情報を次に示す．

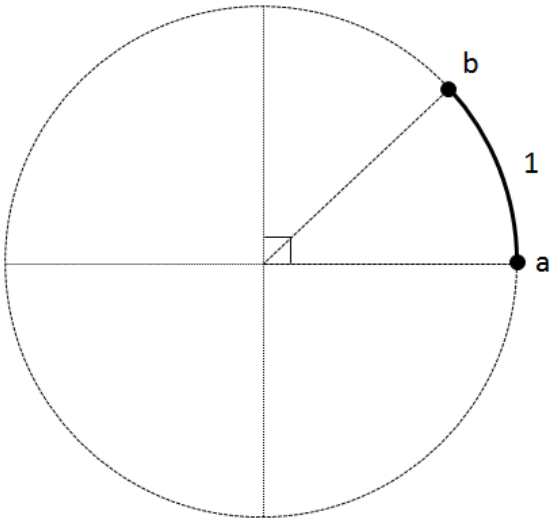


位相情報

本システムでは，円弧の位相情報として，頂点と稜線に関する情報を保持する．円弧の位相情報を次に示す．

パラメータ	型	説明
m_hEdge	Hashtable	稜線に関する情報

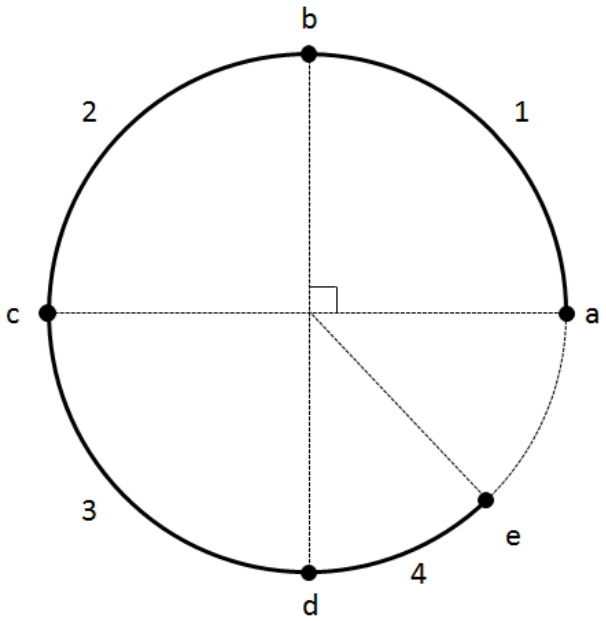
本システムでは，円弧の周上に頂点を設定し，各頂点を結ぶ線分を稜線とする．位相の頂点は，指定した円弧の角度によって位相情報の定義が異なる．始角と終角の差が90度未満の円弧の位相情報を次に示す．



円弧の頂点は，円周上の始角と終角を頂点と定義し，頂点間を結ぶ稜線を定義する．円弧の稜線と頂点の対応を次に示す．

稜線番号	頂点番号	
	始点	終点
1	a	b

始角と終角の差が 90 度以上の円弧の位相情報を次に示す．

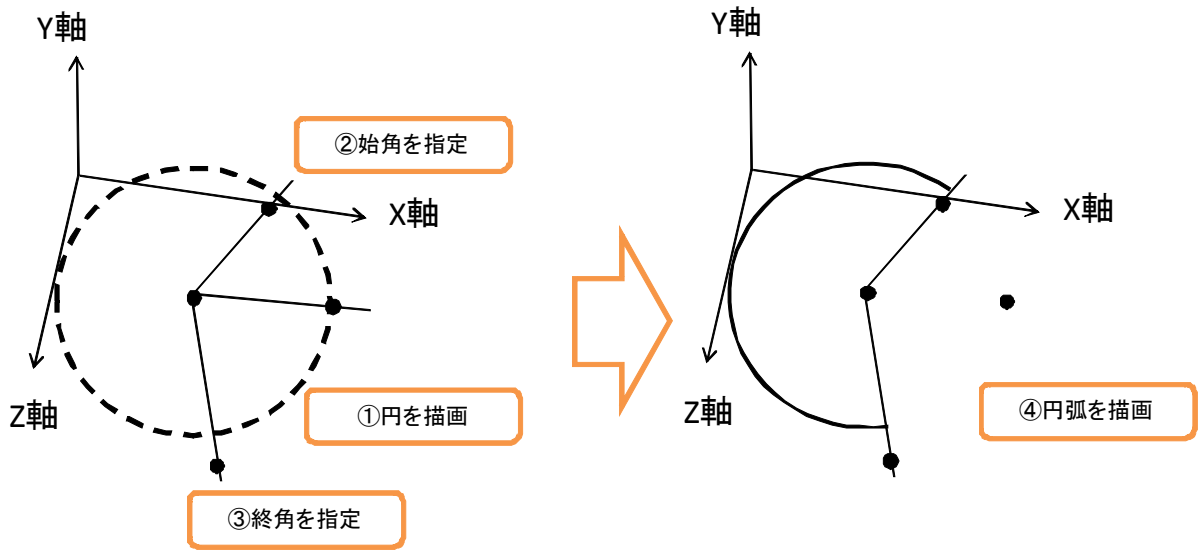


円弧の頂点は，円周上の始角と終角を頂点と定義し，さらに，始角から 90 度毎の頂点を定義する．そして，各頂点間を結ぶ稜線を定義する．円弧の稜線と頂点の対応を次に示す．

稜線番号	頂点	
	始点	終点
1	a	b
2	b	c
3	c	d
4	d	e

描画方法

本システムでは，モデル空間上においてマウス操作で中心点，半径，始角と終角を指定することで円弧を描画する．円弧の描画手順を次に示す．



本システムでは，まず，中心点（X ,Y ,Z）をマウス操作で指定する．次に，半径を表す点をモデル空間上にマウス操作で指定する．そして，円弧の始角と終角の順にマウス操作で指定する．最後に，これらの情報から円弧を描画する．ただし，円弧の中心点，半径を表す点，始角と終角を表す点は同一平面上に投影するものとする．また，円弧の描画の際，円弧のラバーバンド表示を行う．円弧のラバーバンドでは，中心点を指定後，半径を指定するまで，マウスの移動を検出し，マウスの 3 次元座標に基づいた円を描画する．次に，始角を指定するまで，円弧の周上の点から始角を結ぶ弦を描画する．最後に，終角を指定するまで，始角から終角を結ぶ弦を描画する．

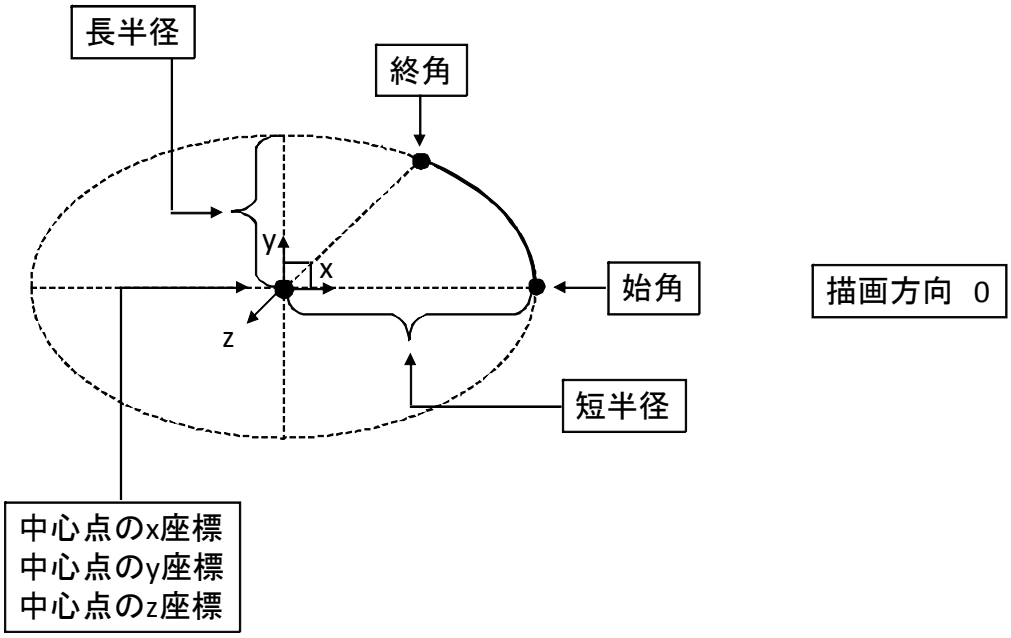
OpenGL には円弧を描画するための関数が用意されていないため，円弧を描画するには，円の場合と同様に，中心点，半径，始角，終角から円弧の周上の頂点座標を算出し，折線に近似する．なお，円弧の周上の頂点の数は 1080 とする．

4.6.7　楕円弧

本節では，楕円弧の描画方法について説明する．

幾何情報

楕円弧の幾何情報を次に示す．

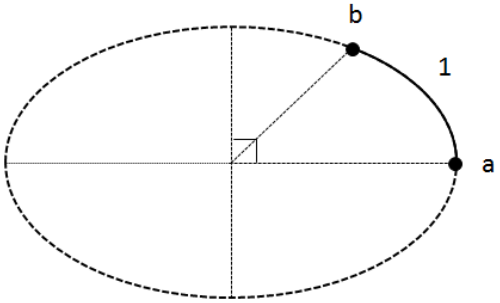


位相情報

本システムでは，楕円弧の位相情報として，頂点と稜線に関する情報を保持する．楕円弧の位相情報を次に示す.

パラメータ	型	説明
m_hEdge	Hashtable	稜線に関する情報

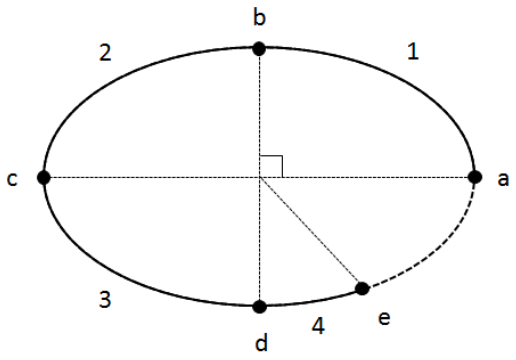
本システムでは，楕円弧の周上に頂点を設定し，各頂点を結ぶ線分を稜線とする．位相の頂点は，指定した楕円弧の角度によって位相情報の定義が異なる．始角と終角の差が90度未満の楕円弧の位相情報を次に示す.



楕円弧の頂点は，円周上の始角と終角を頂点と定義し，頂点間を結ぶ稜線を定義する．円弧の稜線と頂点の対応を次に示す.

稜線番号	頂点番号	
	始点	終点
1	a	b

始角と終角の差が 90 度以上の楕円弧の位相情報を次に示す.

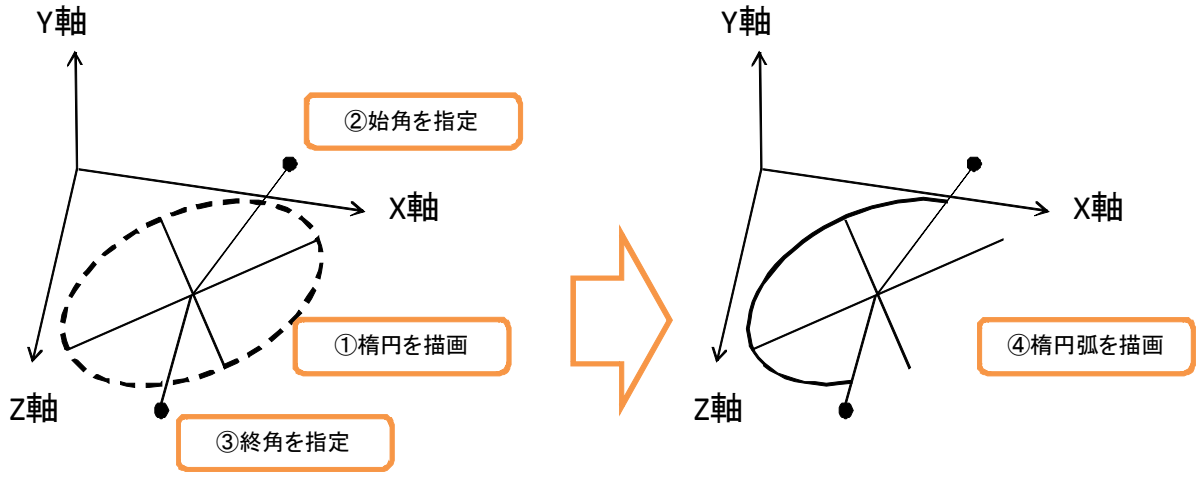


楕円弧の頂点は，楕円弧の周上の始角と終角を頂点と定義し，さらに，始角から 90 度毎の頂点を定義する．そして，各頂点間を結ぶ稜線を定義する．楕円弧の稜線と頂点の対応を次に示す.

稜線番号	頂点番号	
	始点	終点
1	a	b
2	b	c
3	c	d
4	d	e

描画方法

本システムでは，モデル空間上においてマウス操作で中心点，長半径と短半径，回転角，始角と終角を指定することで楕円弧を描画する．楕円弧の描画手順を次に示す.



本システムでは，まず，楕円を描画するための情報を指定する．次に，楕円弧の始角と終角を表す点をモデル空間上にマウス操作で指定する．最後に，これらの情報から楕円弧を描画する．ただし，楕円と楕円弧の始角と終角を表す点は同一平面上に投影するものとする．また，楕円弧の描画の際，楕円弧のラバーバンド表示を行う．楕円弧のラバーバンドでは，まず，楕円の情報を指定するまでは，楕円と同様のラバーバンド表示を行う．次に，始角を指定するまで，楕円弧の周上の点から始角を結ぶ弦を描画する．最後に，終角を指定するまで，始角から終角を結ぶ弦を描画する．

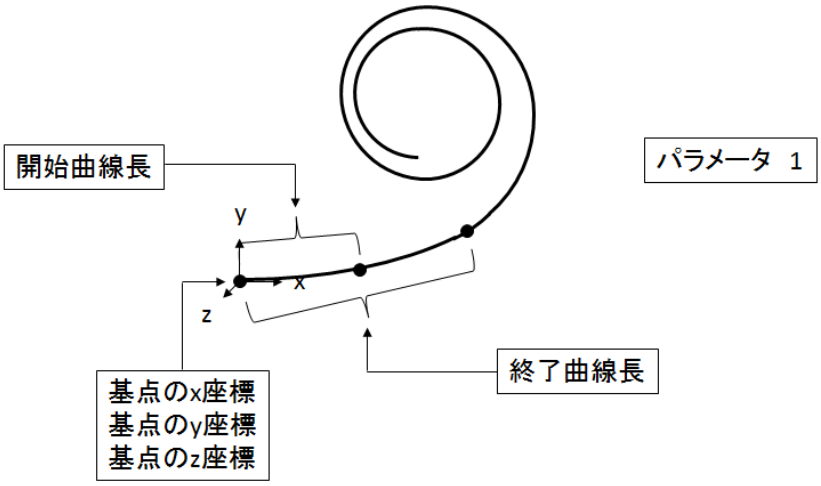
OpenGL には楕円弧を描画するための関数が用意されていないため，楕円弧を描画するには，円の場合と同様に，楕円弧の始角と終角の間の周上の頂点座標を算出し，折線に近似する．なお，楕円弧の周上の頂点の数は 1080 とする．

4.6.8　クロソイド

本節では，クロソイドの描画方法について説明する．

幾何情報

クロソイドの幾何情報を次に示す．

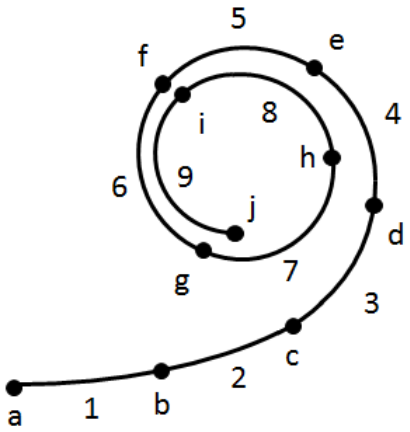


位相情報

本システムでは，クロソイドの位相情報として，頂点と稜線に関する情報を保持する．クロソイドの位相情報を次に示す．

パラメータ	型	説明
m_hEdge	Hashtable	稜線に関する情報

本システムでは，位相の頂点を基にクロソイドの位相情報を定義する．位相の頂点は，各線分の始点座標と終点座標に定義する．クロソイドでは，位相情報を定義するにあたり，クロソイドを任意の頂点数の折線に近似する．折線に近似する際，頂点数が|(開始曲線長-終了曲線長)|/10+1 個になる折線に近似する．クロソイドの位相の頂点は，折線を構成する各線分の始点座標と終点座標に定義する．折線同様，稜線は各線分の始点座標と終点座標から定義する．2 つの位相の頂点から稜線を設定する処理を繰り返すことで，位相情報を定義する．クロソイドの位相情報のを次に示す．

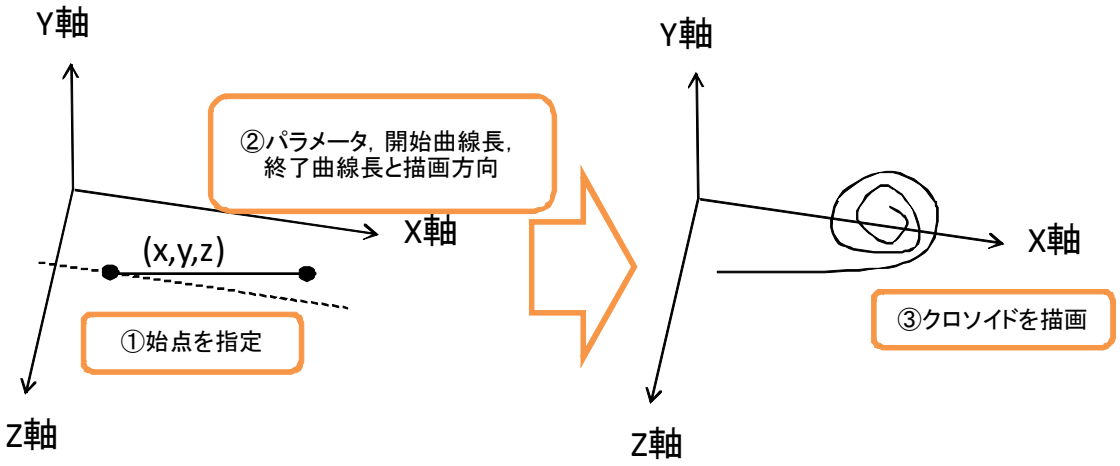


クロソイドの稜線と頂点の対応を次に示す.

稜線番号	頂点番号	
	始点	終点
1	a	b
2	b	c
3	c	d
4	d	e
5	e	f
6	f	g
7	g	h
8	h	i
9	i	j

描画方法

本システムでは，モデル空間上においてマウス操作で始点を指定し，ダイアログからパラメータ，開始曲線長，終了曲線長と描画方向を指定することでクロソイドを描画する．クロソイドの描画手順を次に示す.



本システムでは，まず，始点（X ,Y ,Z）をマウス操作で指定する．次に，ダイアログ上でパラメータ，開始曲線長，終了曲線長と描画方向を指定する．最後に，始点，パラメータ，開始曲線長，終了曲線長と描画方向からクロソイドを描画する．

OpenGL には，クロソイドを描画するための関数が用意されていません．そのため，クロソイドを描画するには，クロソイドの線上の頂点座標を算出し，折線に近似する．クロソイドの線上の頂点座標の算出方法を次に示す．

まず，曲線長と半径から成立する関係式を式 1 に示す．

$$\tau = L / 2R \qquad (1)$$

τ は，X 軸とクロソイド上の点の接線とのなす角を表す．クロソイド上の座標（x，y，z）を算出する式を式 2 に示す．

$$\begin{aligned} x &= A\sqrt{2\tau}\left(1-\frac{1}{2!\times 5}\tau^2+\frac{1}{4!\times 9}\tau^4-\frac{1}{6!\times 13}\tau^6+\ldots\right) \\ y &= A\tau\sqrt{2\tau}\left(\frac{1}{3}-\frac{1}{3!\times 7}\tau^2+\frac{1}{5!\times 11}\tau^4-\frac{1}{7!\times 15}\tau^6+\ldots\right) \end{aligned} \qquad (2)$$

ただし，A はパラメータを表す．式 2 を一般化した式を式 3 に示す．

$$\begin{aligned} &A\sqrt{2\tau}\left((-1)^{k-1}\times\frac{1}{(2(k-1))!\times(4k-3)}\right)\times\tau^{2(k-1)} \\ &A\tau\sqrt{2\tau}\left((-1)^{k-1}\times\frac{1}{(2k-1)!\times(4k-1)}\right)\times\tau^{2(k-1)} \end{aligned} \qquad (3)$$

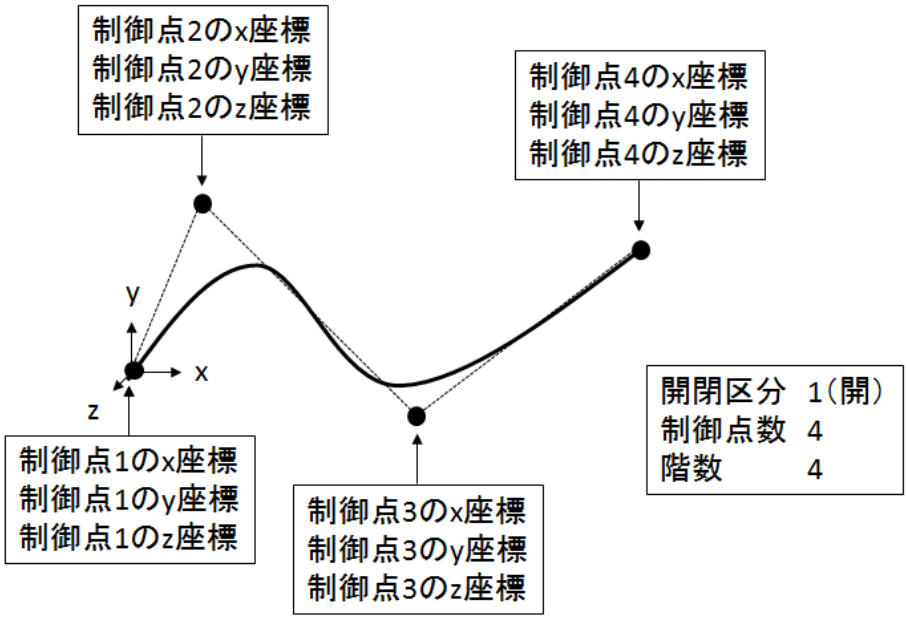
次に，式 2 と式 3 を使用して，クロソイドを構成する座標の具体的な算出方法を説明する．まず，式 1 からτを算出する．次に，式 3 の k を「1，2，3，…」と変え，その時の座標を算出する．そして，k と k-1 の時の座標の差が 0.001 になった時点で計算を終了する．これらの処理を曲線長の値をΔL ずつ加算し，クロソイド上の全ての座標を算出する．

4.6.9　ベジェ曲線

本節では，ベジェ曲線の描画方法について説明する．

幾何情報

ベジェ曲線の幾何情報を次に示す．



開閉区分とは，ベジェ曲線の開閉を指定する値である．開閉区分が 0 の場合，閉じたベジェ曲線を描画する．開閉区分が 1 の場合，開いたベジェ曲線を描画する．制御点数とは，作成するベジェ曲線全体で必要となる頂点の数である．階数とは，1 次のベジェ曲線を作成する際に必要となる頂点の数である．N 次の階数で構成されるベジェ曲線を M 個作成する場合，制御点数は，(N-1)*M+1 個になる．

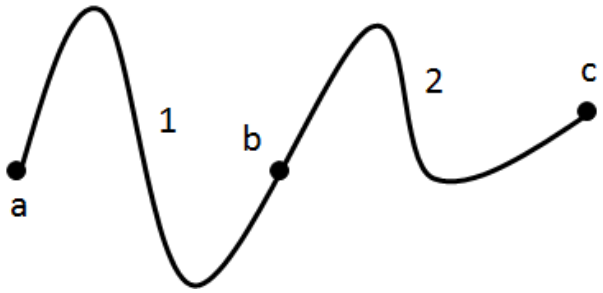
位相情報

本システムでは，ベジェ曲線の位相情報として，頂点と稜線に関する情報を保持する．

ベジェ曲線の位相情報を次に示す.

パラメータ	型	説明
m_hEdge	Hashtable	稜線に関する情報

本システムでは，ベジェ曲線の始点座標と終点座標を位相情報の頂点に定義し，各頂点および各頂点からなる稜線を保持する．ベジェ曲線の位相情報を次に示す.

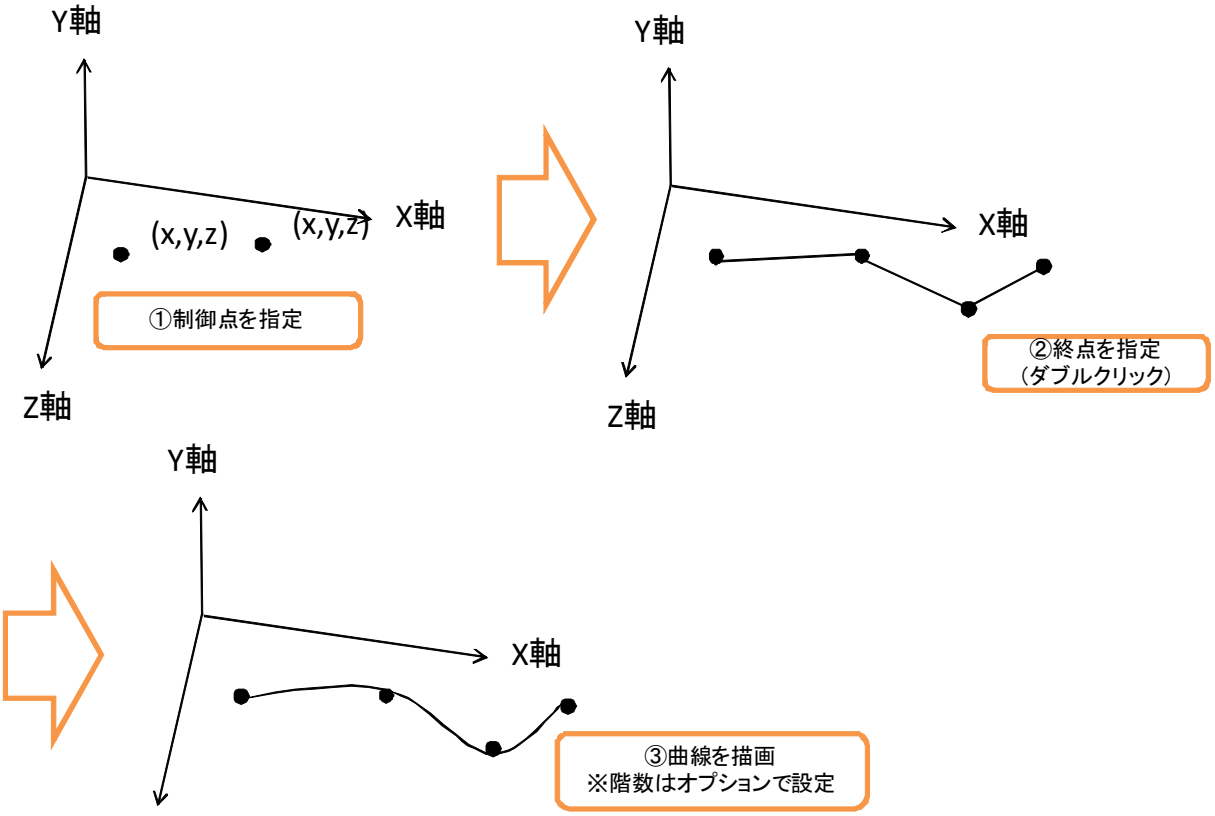


ベジェ曲線の稜線と頂点の対応を次に示す.

稜線番号	頂点番号	
	始点	終点
1	a	b
2	b	c

描画方法

本システムでは，モデル空間上においてマウス操作で各制御点を指定することベジェ曲線を描画する．ベジェ曲線の描画手順を次に示す.



本システムでは，まず，各制御点 (X ,Y ,Z) をマウス操作で指定する．次に，終点 (X ,Y ,Z) をマウス操作（ダブルクリックイベントを使用）で指定する．最後に，各制御点からベジェ曲線を描画する．ただし，各制御点の投影方法は，折線と同様の方法とする．また，ベジェ曲線の階数については，オプションで指定するものとする．また，ベジェ曲線を描画する際，ラバーバンド表示を行う．ベジェ曲線のラバーバンド表示では，折線と同様のラバーバンド表示を行う．具体的には，終点を指定するまで，ベジェ曲線を描画するのではなく，制御点を頂点とする折線を描画する．

ベジェ曲線を描画するには，glMap1d 関数，glEnable 関数と glEvalCoord1d 関数を使用する．まず，ベジェ曲線を構成する制御点の座標データを配列で用意する．次に，glMap1d 関数と glEnable 関数を使用してベジェ曲線を設定する．最後に，glEvalCoord1d 関数を使用して，有効化したベジェ曲線を描画する．

glMap1d 関数の定義を次に示す．

```
void glMap1d( GLenum target, GLdouble u1, GLdouble u2, Glint stride,GLint order,
              const GLdouble* points)
```

glMap1d 関数は 6 つの引数をとり，戻り値はない．glMap1d 関数の引数を次に示す．

引数名	型	引数の意味
Target	GLenum	頂点座標を示す定数
u1	GLdouble	glEvalCoord1d 関数の引数 u がとり得る値の範囲
u2	GLdouble	glEvalCoord1d 関数の引数 u がとり得る値の範囲
stride	GLint	ベジェ曲線を構成する制御点の間隔
order	GLint	ベジェ曲線を構成する制御点の数
points	GLdouble*	ベジェ曲線を構成する制御点の配列のポインタ

glMap1d 関数の引数 Target に GL_MAP1_VERTEX_3 を設定することで，3 次元空間上にベジェ曲線を描画する．引数 u1，u2 には，次に説明する glEvalCoord1d 関数の引数 u のとり得る値の範囲を設定する．引数 stride には，引数 points で参照される制御点のデータにおける各制御点の間隔を設定する．引数 order には，引数 points で参照される制御点のデータにおける各制御点の数を設定する．

glEvalCoord1d 関数の定義を次に示す．

```
void glEvalCoord1d(GLdouble u)
```

glEvalCoord1d 関数は 1 つの引数をとり，戻り値はない．glEvalCoord1d 関数の引数を次に示す．

引数名	型	引数の意味
u	GLdouble	ベジェ曲線を構成する頂点データ

glEvalCoord1d 関数の引数 u は，glMap1d 関数の引数 u1，u2 に設定された範囲内の値を設定する．本システムでは，引数 u1 は 0，引数 u2 は 1 に設定する．そのため，引数 u に 0 を設定するとベジェ曲線の始点が描画される．また，引数 u に 1 を設定するとベジェ曲線の終点が描画される．したがって，引数 u の数が多いと，ベジェ曲線を構成する頂点が多くなるため，滑らかな曲線になる．ベジェ曲線の描画方法の例を次に示す．

```
// ベジェ曲線の座標を格納する 2 次元配列の宣言
double **dCtlPoint;
// メモリの確保（1 次元目の配列のサイズ設定）
dCtlPoint=new double*[m_iNumber/(m_iOrder-1)];
// 2 次元目の配列のメモリを確保するための繰り返し
for(int i=0;i<m_iNumber/(m_iOrder-1);i++){
    // メモリの確保（2 次元目の配列のサイズ設定）
    dCtlPoint[i]=new double[(m_iOrder)*3];
}
```

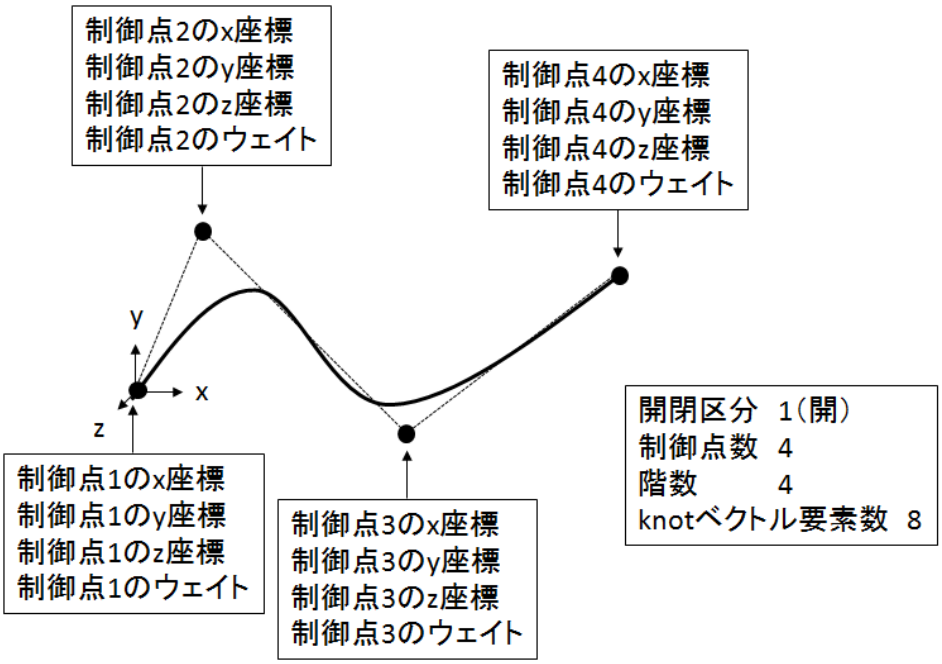
```
// 配列に値を格納するための繰り返し
for(int i=0;i<((m_iNumber)/(m_iOrder-1));i++){
    // 配列に値を格納するための繰り返し
    for(int j=0;j<(m_iOrder)*3;j++){
        // X 座標値を格納
        if(j%3 == 0) dCtlPoint[i][j] = (double)(alX[j/3+i*(m_iOrder-1)]);
        // Y 座標値を格納
        if(j%3 == 1) dCtlPoint[i][j] = (double)(alY[j/3+i*(m_iOrder-1)]);
        // Z 座標値を格納
        if(j%3 == 2) dCtlPoint[i][j] = (double)(alZ[j/3+i*(m_iOrder-1)]);
    }
}

// ベジェ曲線を複数描画するための繰り返し
for(int i=0;i<(m_iNumber/(m_iOrder-1));i++){
    // i 番目のベジェ曲線を設定
    glMap1d(GL_MAP1_VERTEX_3, 0, 1, 3, (m_iOrder), dCtlPoint[i]);
    // ベジェ曲線の有効化
    glEnable(GL_MAP1_VERTEX_3);
    // ベジェ曲線の描画の実行
    glBegin(GL_LINE_STRIP);
    // ベジェ曲線を描画するために分割回数繰り返し
    for(int t = 0; t <= iStep; ++t){
        // ベジェ曲線の描画
        glEvalCoord1d(t/(iStep*1.0));
    }
    // ベジェ曲線の描画の終了
    glEnd();
    glDisable(GL_MAP1_VERTEX_3);
}
```

4.6.10 NURBS曲線

本節では，NURBS 曲線の描画方法について説明する．

幾何情報

NURBS 曲線の幾何情報を次に示す．



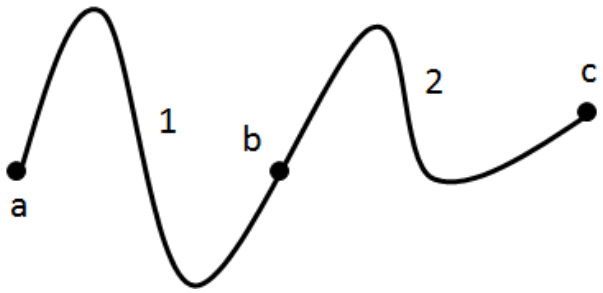
開閉区分とは，NURBS 曲線の開閉を指定する値である．開閉区分が 0 の場合，閉じた NURBS 曲線を描画する．開閉区分が 1 の場合，開いた NURBS 曲線を描画する．制御点数とは，作成する NURBS 曲線全体で必要となる頂点の数である．階数とは，1 次の NURBS 曲線を作成する際に必要となる頂点の数である．N 次の階数で構成される NURBS 曲線を M 個作成する場合，制御点数は，(N-1)*M+1 個になる．

位相情報

本システムでは，NURBS 曲線の位相情報として，頂点と稜線に関する情報を保持する．NURBS 曲線の位相情報を次に示す．

パラメータ	型	説明
m_hEdge	Hashtable	稜線に関する情報

本システムでは，ベジェ曲線同様，NURBS曲線の始点座標と終点座標を位相情報の頂点に定義し，頂点と各頂点からなる稜線を保持する．NURBS曲線の位相情報を次に示す．

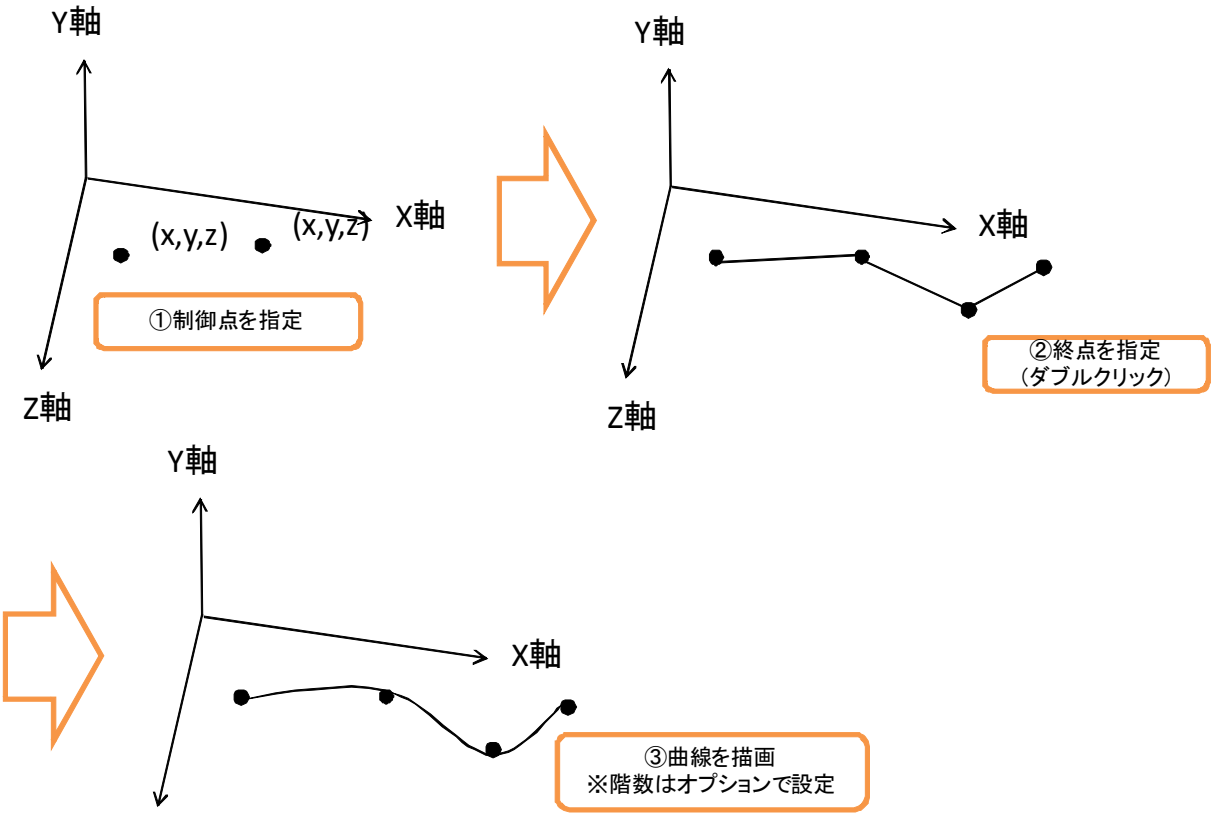


NURBS 曲線の稜線と頂点の対応を次に示す.

稜線番号	頂点番号	
	始点	終点
1	a	b
2	b	c

描画方法

本システムでは, モデル空間上においてマウス操作で各制御点を指定すること NURBS 曲線を描画する. NURBS 曲線の描画手順を次に示す.



本システムでは，まず，各制御点（X ,Y ,Z）をマウス操作で指定する．次に，終点（X ,Y ,Z）をマウス操作（ダブルクリックイベントを使用）で指定する．最後に，各制御点から NURBS 曲線を描画する．ただし，各制御点の投影方法は，折線と同様の方法とする．また，NURBS 曲線の階数については，オプションで指定するものとする．さらに，NURBS 曲線の描画の際，ラバーバンド表示を行う．NURBS 曲線のラバーバンド表示では，ベジェ曲線の同様に制御点を頂点とする折線を描画する．

NURBS 曲線を描画するには，gluNewNurbsRenderer 関数，gluNurbsProperty 関数，gluNurbsCurve 関数，gluBeginCurve 関数と gluEndCurve 関数を使用する．まず，gluNewNurbsRenderer 関数を使用して，NURBS オブジェクトを作成する．次に，gluNurbsProperty 関数を使用して，作成した NURBS オブジェクトの属性を設定する．そして，gluNurbsCurve 関数を使用して作成した NURBS オブジェクトの形状を定義し，NURBS 曲線を描画する. 最後に, gluBeginCurve 関数と gluEndCurve 関数を使用して描画する NURBS 曲線を設定する．

gluNewNurbsRenderer 関数の定義を次に示す.

```
GLUnurbsObj* gluNewNurbsRenderer (void)
```

gluNewNurbsRenderer 関数を使用することで，NUBRS オブジェクトを生成する．引数はなく，戻り値は，作成した NURBS オブジェクトを参照するポインタである．

gluNurbsProperty 関数の定義を次に示す.

```
void gluNurbsProperty ( GLUnurbs *nobj, GLenum property, GLfloat value)
```

gluNurbsProperty 関数は，3 つの引数をとり，戻り値はない．gluNurbsProperty 関数の引数を次に示す.

引数名	型	引数の意味
nobj	GLUnurbs*	NURBS オブジェクトを示す定数
property	GLenum	設定する属性を示す定数
value	GLfloat	Property で設定した属性の値

本システムでは，gluNurbsProperty 関数の引数である property は，GLU_SAMPLING _TOLERANCE を設定する．この場合，value は，NURBS 曲線の描画に使用する多角形の線分や辺の最大の長さを定義する. そのため, value の値が小さいほど滑らかな曲線を描画できる．本システムでは，gluNurbsProperty 関数の引数 value は，0.1 を設定する．

gluNurbsCurve 関数の定義を次に示す.

```
gluNurbsCurve( GLUnurbs* nobj, GLint nknots, GLfloat* knot, GLint stride,
                GLfloat* ctlarray, GLint order, GLenum type);
```

gluNurbsCurve 関数は，7 つの引数をとり，戻り値はない．gluNurbsCurve 関数の引数を次に示す.

引数名	型	引数の意味
nobj	GLUnurbsObj*	NURBS オブジェクトを示す定数
nknots	Glint	ノットベクトルの要素数
knot	GLfloat*	ノットベクトルの配列
stride	GLint	1 頂点に必要な要素数
ctlarray	GLfloat*	制御点の配列
order	GLint	制御点の数
type	GLenum	曲線の型を示す定数

gluNurbsCurve 関数の引数である type に GL_MAP1_VERTEX_3 を設定することで, NURBS 曲線を 3 次元空間上に描画する.

gluBeginCurve 関数と gluEndCurve 関数の定義を次に示す.

```
gluBeginCurve()

描画する NURBS 曲線

gluEndCurve()
```

gluBeginCurve 関数と gluEndCurve 関数は対になる．この 2 つの関数の間に設定した gluNurbsCurve 関数が NURBS 曲線を描画する．NURBS 曲線の描画方法の例を次に示す.

```
// Nurbs 曲線の座標を格納する配列の生成
float *fCtlPoint;
// メモリの確保
fCtlPoint=new float[m_iNumber*4];
// 配列に値を格納するための繰り返し
for(int i=0;i<m_iNumber*4;i++){
    // 配列に値を格納
    if(i%4 == 0) fCtlPoint[i] = (double)alX[i/4];
    if(i%4 == 1) fCtlPoint[i] = (double)alY[i/4];
    if(i%4 == 2) fCtlPoint[i] = (double)alZ[i/4];
    if(i%4 == 3) fCtlPoint[i] = (double)m_alWeight[i/4];
```

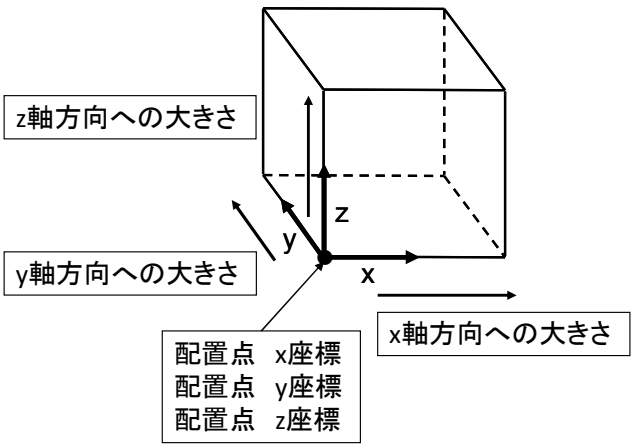
```
}

// knot ベクトルの値を格納する配列の生成
float *fKnot;
// knot ベクトルのメモリ確保
fKnot=new float[m_alKnot->Count];
// knot ベクトルに値を格納するための繰り返し
for(int i=0;i<m_alKnot->Count;i++){
      // knot ベクトルに値を格納
      fKnot[i] = (double)(m_alKnot[i]);
}

// Nurbs オブジェクトの宣言
GLUnurbsObj *NurbsObj;
// Nurbs オブジェクト作成
NurbsObj = gluNewNurbsRenderer();
//  サンプリング範囲の設定(小さいほど滑らか)
gluNurbsProperty(NurbsObj, GLU_SAMPLING_TOLERANCE,0.1);
gluNurbsCurve(NurbsObj, m_alKnot->Count, fKnot,
  4, fCtlPoint, m_iOrder, GL_MAP1_VERTEX_4);
```

4.6.11　直方体

本節では，直方体の描画方法について説明する．

幾何情報

直方体の幾何情報を次に示す．



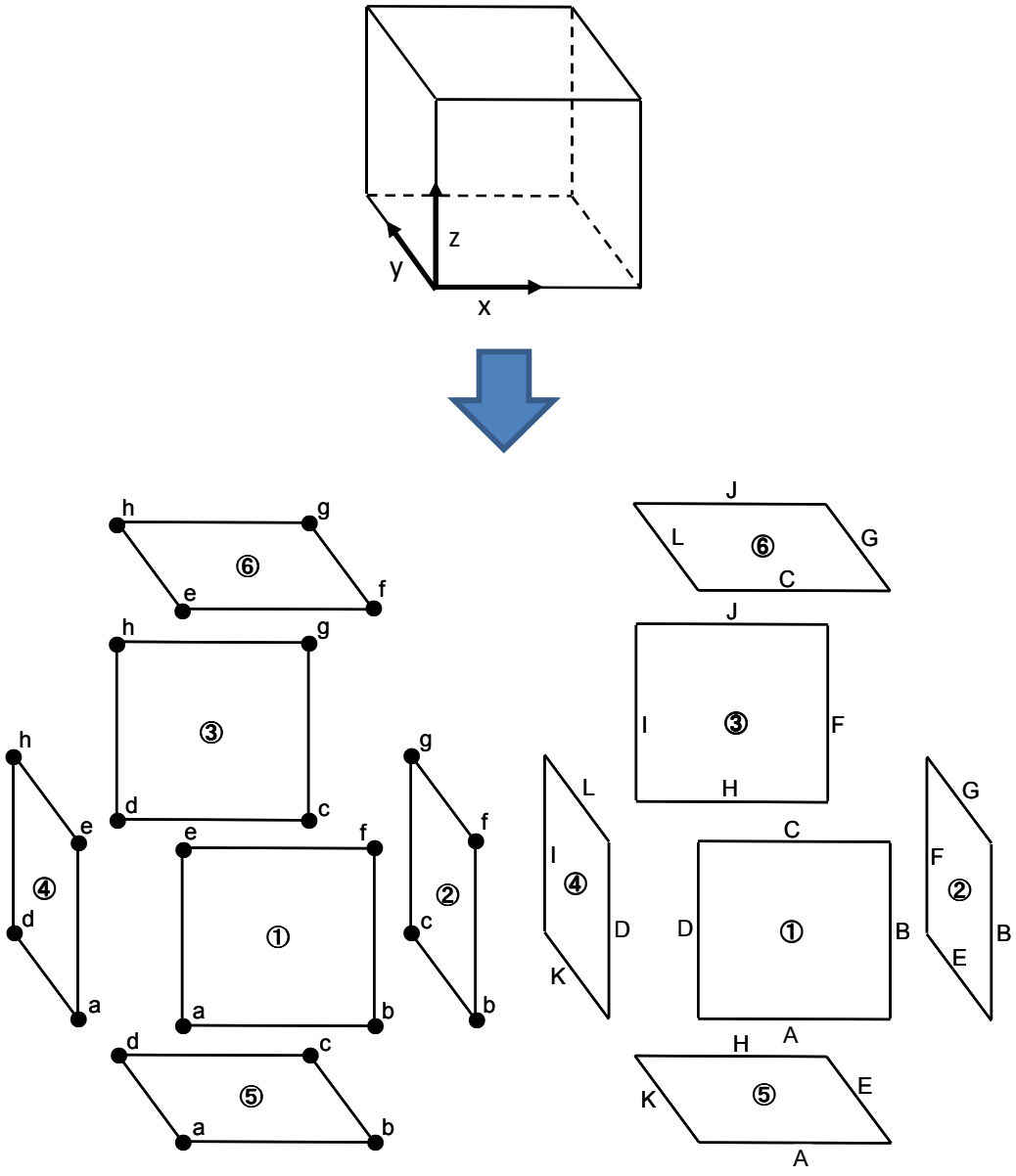
位相情報

本システムでは，直方体の位相情報として，頂点，稜線と面に関する情報を保持する．

直方体の位相情報を次に示す.

パラメータ	型	説明
m_hFace	Hashtable	面に関する情報

本システムでは，8 個の頂点，12 個の稜線と 6 個の面を基に直方体の位相情報を設定する．直方体の位相情報を次に示す.

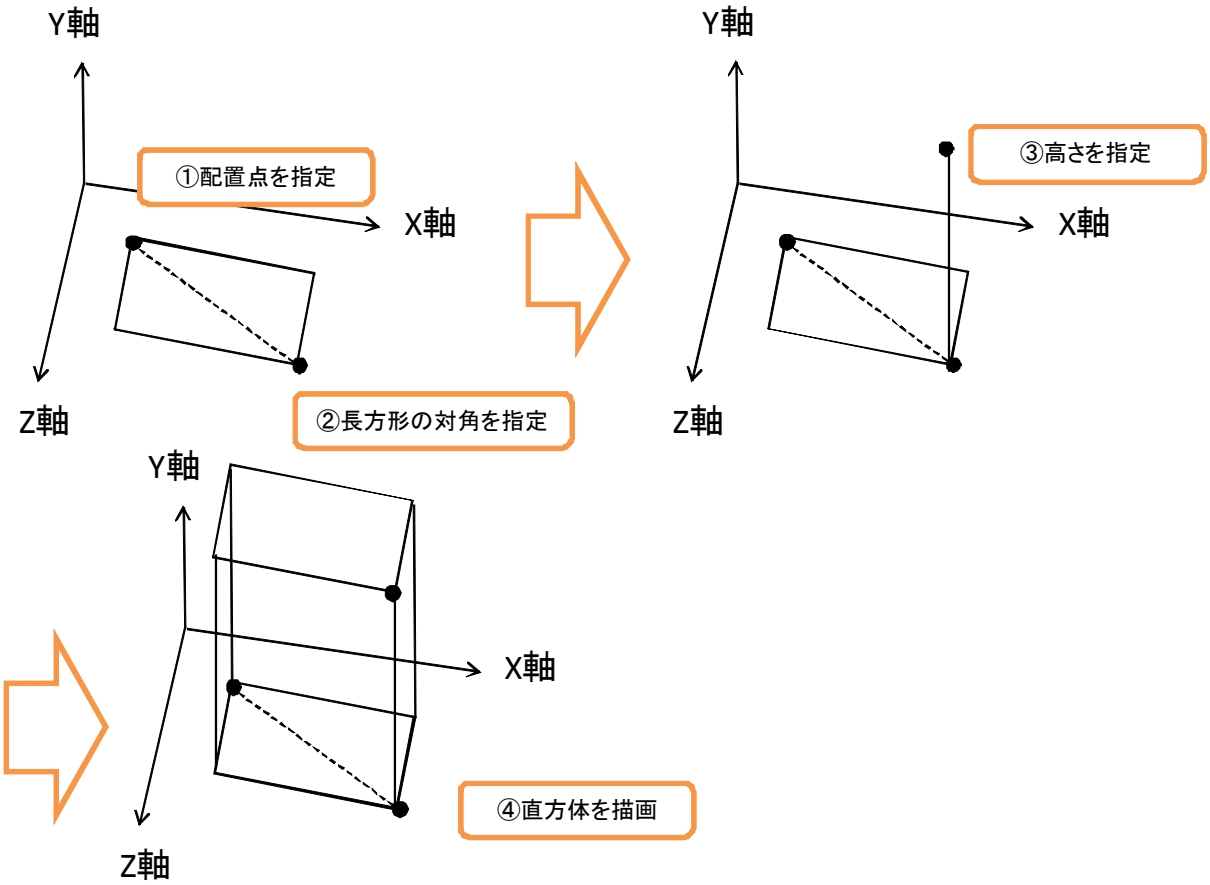


直方体の面，稜線と頂点の対応を次に示す.

面番号	稜線	頂点
1	A，B，C，D	a，b，f，e
2	E，F，G，B	b，c，g，f
3	H，I，J，F	c，d，h，g
4	K，D，L，I	d，h，e，a
5	A，E，H，K	a，b，c，d
6	C，G，J，L	d，a，e，h

描画方法

本システムでは，モデル空間上においてマウス操作で直方体を描画する．直方体の描画手順を次に示す．



本システムでは，まず，配置点（X ,Y ,Z）をマウス操作で指定する．次に，長方形の対角線上の点をマウス操作で指定する．そして，直方体の高さを表す点（X ,Y ,Z）をマウス操作で指定する．最後に，配置点，対角線上の点と高さを表す点から直方体を描画する．ただし，配置点と対角線上の点は同一平面上に投影し，高さを表す点は，対角線上の点を通り，配置点と対角線上の点からなる平面に垂直な平面に投影する．また，直方体を描画

する際，ラバーバンド表示を行う．直方体のラバーバンドでは，まず，配置点を指定後，長方形の対角を指定するまで，マウスの移動を検知し，マウスの 3 次元座標を算出し，長方形を描画する．次に，直方体の高さを指定するまで，同様にマウスの移動を検知し，マウスの 3 次元座標を算出し，直方体を描画する．

OpenGL には，直方体を描画する関数が用意されていない．そのため，本システムでは，6 個の面を描画することにより直方体を描画する．面の描画には，glVertex3d 関数を使用する．直方体の描画方法の例を次に示す．

```
// 正面
// 描画の開始
glBegin(GL_POLYGON);
    // 描画の実行
    glVertex3d(0, 0, 0);
    glVertex3d(m_dLengthX, 0, 0);
    glVertex3d(m_dLengthX, m_dLengthY, 0);
    glVertex3d(0, m_dLengthY, 0);
// 描画の終了
glEnd();

// 奥面
// 描画の開始
glBegin(GL_POLYGON);
    // 描画の実行
    glVertex3d(m_dLengthX, 0, m_dLengthZ);
    glVertex3d(0, 0, m_dLengthZ);
    glVertex3d(0, m_dLengthY, m_dLengthZ);
    glVertex3d(m_dLengthX, m_dLengthY, m_dLengthZ);
// 描画の終了
glEnd();

// 下面
// 描画の開始
glBegin(GL_POLYGON);
    // 描画の実行
    glVertex3d(0, 0, m_dLengthZ);
    glVertex3d(m_dLengthX, 0, m_dLengthZ);
    glVertex3d(m_dLengthX, 0, 0);
    glVertex3d(0, 0, 0);
// 描画の終了
glEnd();

// 上面
// 描画の開始
glBegin(GL_POLYGON);
    // 描画の実行
    glVertex3d(0, m_dLengthY, 0);
    glVertex3d(m_dLengthX, m_dLengthY, 0);
    glVertex3d(m_dLengthX, m_dLengthY, m_dLengthZ);
    glVertex3d(0, m_dLengthY, m_dLengthZ);
// 描画の終了
glEnd();

// 右面
```

```
// 描画の開始
glBegin(GL_POLYGON);
    // 描画の実行
    glVertex3d(m_dLengthX, 0, 0);
    glVertex3d(m_dLengthX, 0, m_dLengthZ);
    glVertex3d(m_dLengthX, m_dLengthY, m_dLengthZ);
    glVertex3d(m_dLengthX, m_dLengthY, 0);
// 描画の終了
glEnd();

// 左面
// 描画の開始
glBegin(GL_POLYGON);
    // 描画の実行
    glVertex3d(0, 0, m_dLengthZ);
    glVertex3d(0, 0, 0);
    glVertex3d(0, m_dLengthY, 0);
    glVertex3d(0, m_dLengthY, m_dLengthZ);
// 描画の終了
glEnd();
```

また，面要素を持つモデルは，表示方法をワイヤフレームモデルとサーフェスモデルに切り替えることができる. モデルの表示方法の切り替えには glPolygonMode 関数を使用する. glPolygonMode 関数の定義を次に示す.

```
void glPolygonMode( GLenum face, GLenum mode )
```

glPolygonMode 関数は 2 個の引数をとり，戻り値はない．glPolygonMode 関数の引数を次に示す.

引数名	型	引数の意味
face	GLenum	多角形の描画面
mode	GLenum	描画モード

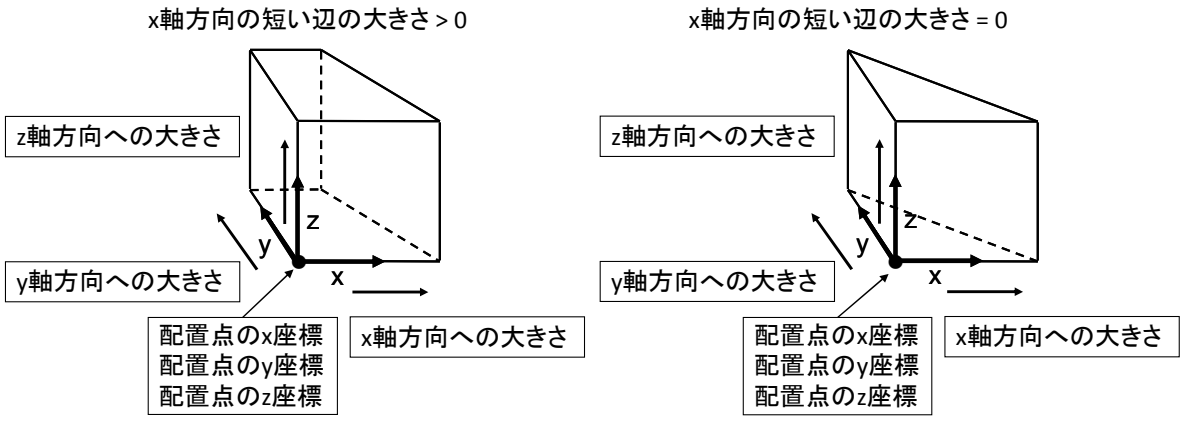
本システムでは，face には GL_FRONT_AND_BACK を設定する．また，mode には，ワイヤフレームモデルを描画する場合, GL_LINE を設定し, サーフェスモデルを描画する場合, GL_FILL を設定する.

4.6.12 ウェッジ

本節では，ウェッジの描画方法について説明する.

幾何情報

ウェッジの幾何情報を次に示す.

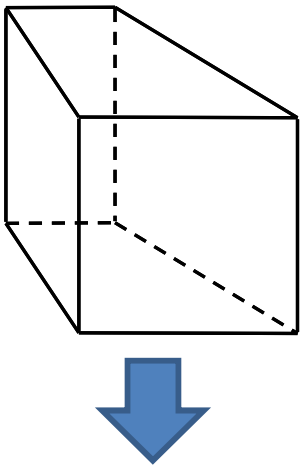


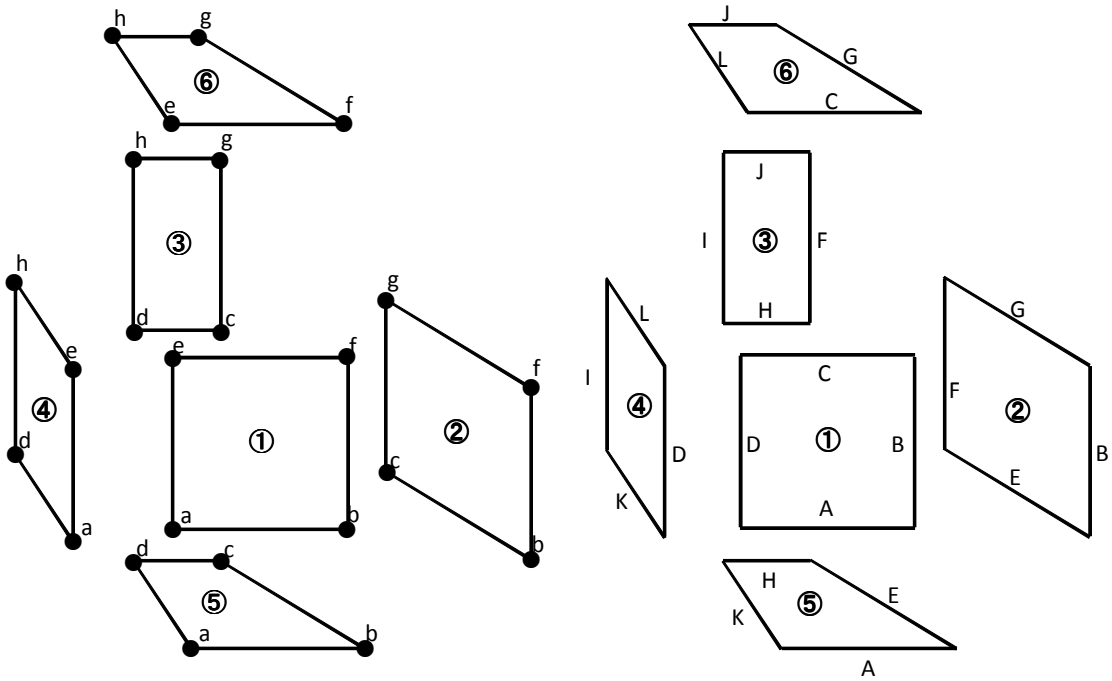
位相情報

本システムでは，ウェッジの位相情報として，頂点，稜線と面に関する情報を保持する．ウェッジの位相情報を次に示す.

パラメータ	型	説明
m_hFace	Hashtable	面に関する情報

本システムでは，8 個の頂点，12 個の稜線と 6 個の面を基にウェッジの位相情報を設定する．ウェッジの位相情報を次に示す.



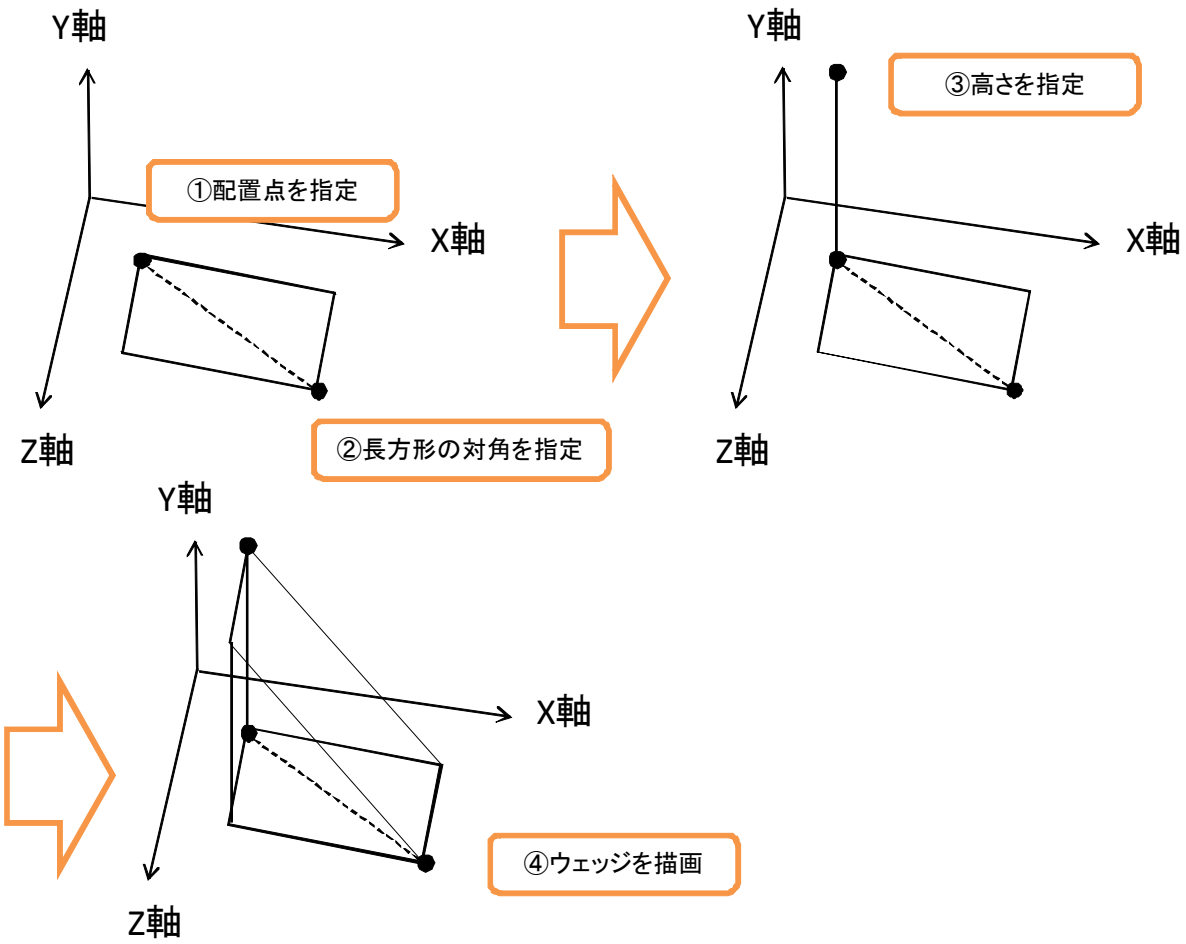


ウェッジの面，稜線と頂点の対応を次に示す.

面番号	稜線	頂点
1	A，B，C，D	a，b，f，e
2	E，F，G，B	b，c，g，f
3	H，I，J，F	c，d，h，g
4	K，D，L，I	d，a，e，h
5	A，E，H，K	a，b，c，d
6	C，G，J，L	e，f，g，h

描画方法

本システムでは，モデル空間上においてマウス操作でウェッジを描画する．ウェッジの描画手順を次に示す.



本システムでは，まず，配置点（X ,Y ,Z）をマウス操作で指定する．次に，長方形の対角線上の点をマウス操作で指定する．そして，ウェッジの高さを表す点（X ,Y ,Z）をマウス操作で指定する．最後に，配置点，対角線上の点と高さを表す点からウェッジを描画する．ただし，配置点と対角線上の点は同一平面上に投影し，高さを表す点は，配置点を通り，配置点と対角線上の点からなる平面に垂直な平面に投影する．また，ウェッジの描画の際，ラバーバンド表示を行う．ウェッジのラバーバンド表示では，まず，配置点を指定後，長方形の対角を指定するまで，マウスの移動を検知し，マウスの 3 次元座標を算出し，長方形を描画する．次に，ウェッジの高さを指定するまで，同様にマウスの移動を検知し，マウスの 3 次元座標を算出し，ウェッジを描画する．

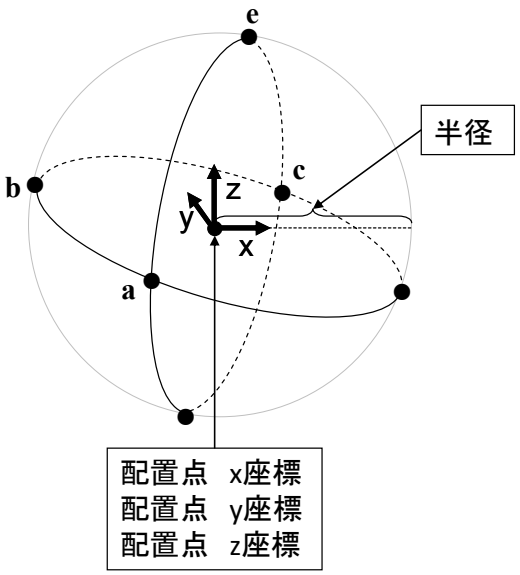
ウェッジの描画方法として，OpenGL には，ウェッジを描画する関数が用意されていない．そのため，本システムでは，直方体の描画と同様に 6 個の面を描画することによりウェッジを描画する．また，ウェッジにおいても，ワイヤフレームモデルとサーフェスモデルに切り替えを可能にする．

4.6.13　球

本節では，球の描画方法について説明する．

幾何情報

球の幾何情報を次に示す．

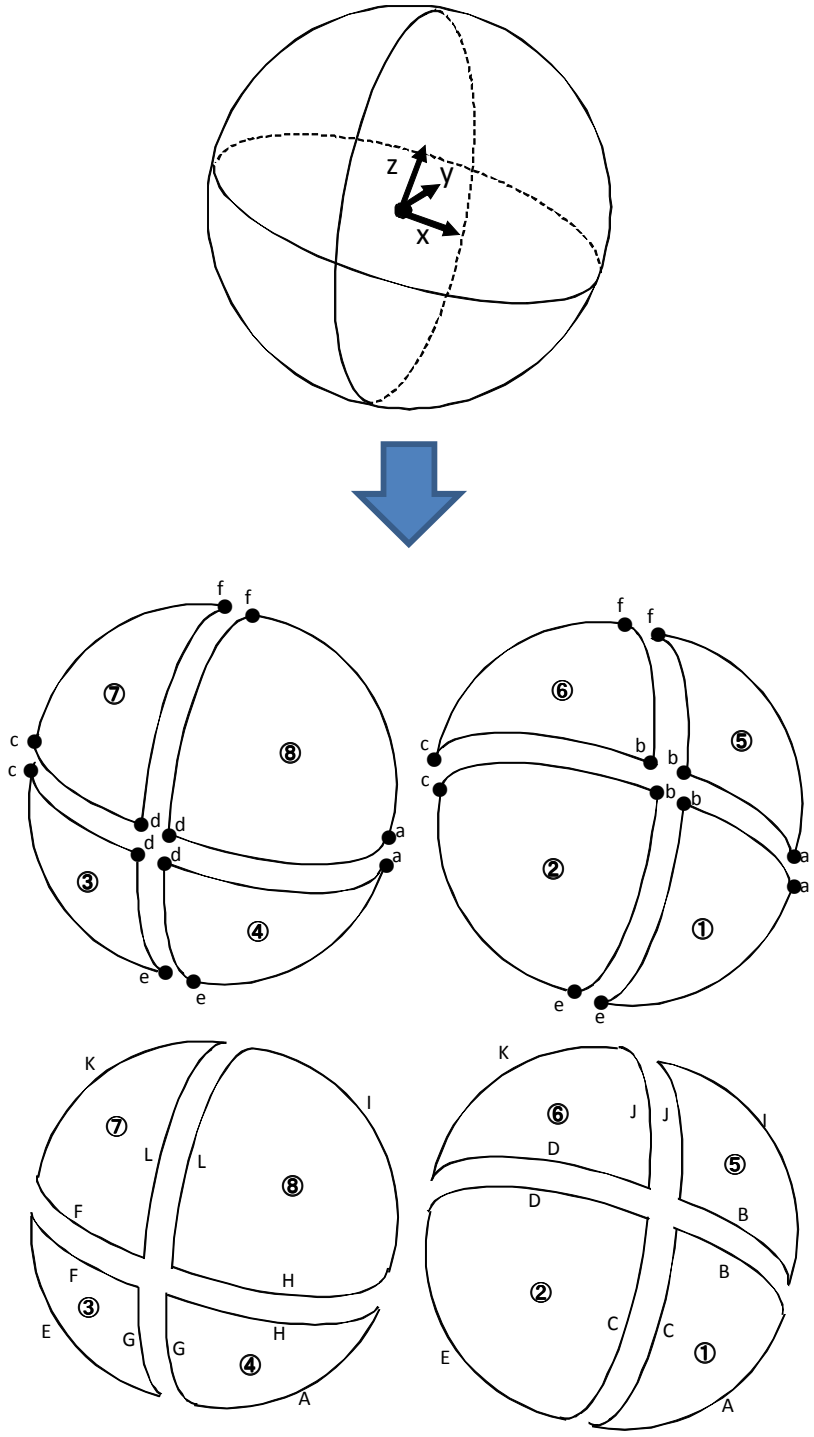


位相情報

本システムでは，球の位相情報として，頂点，稜線と面に関する情報を保持する．球の位相情報を次に示す．

パラメータ	型	説明
m_hFace	Hashtable	面に関する情報

球では，6 個の頂点，12 個の稜線と 8 個の面の位相情報を設定する．球の位相情報を次に示す．



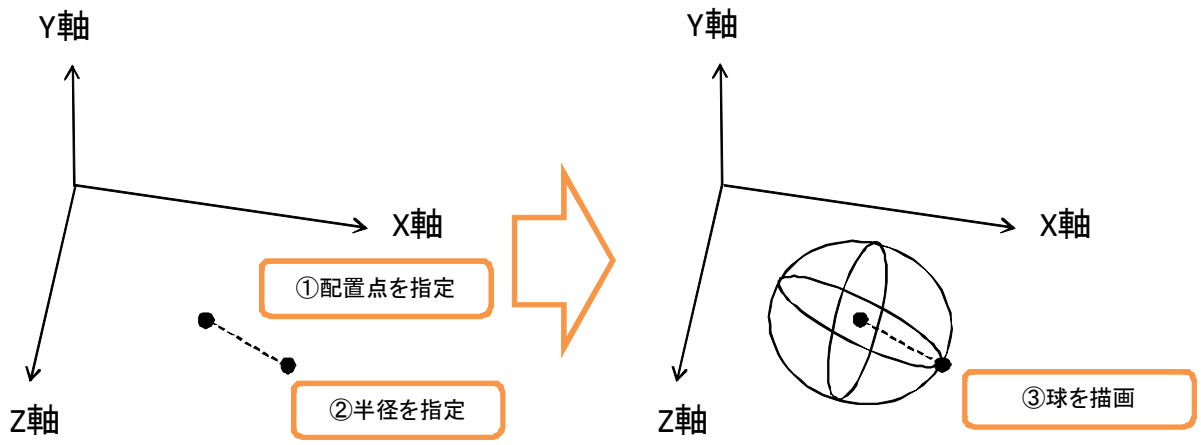
球の面，稜線と頂点の対応を次に示す.

面番号	稜線	頂点
1	A，B，C	e，a，b
2	C，D，E	e，b，c

3	E，F，G	e，c，d
4	G，H，A	e，d，a
5	I，B，J	f，a，b
6	J，D，K	f，b，c
7	K，F，L	f，c，d
8	L，H，I	f，d，a

描画方法

本システムでは，モデル空間上においてマウス操作で球を描画する．球の描画手順を次に示す．



本システムでは，まず，配置点（X ,Y ,Z）をマウス操作で指定する．次に，球の半径を表す点をマウス操作で指定する．最後に，配置点と半径を表す点から球を描画する．ただし，配置点は，座標軸から構成される平面に投影するが，半径を表す点は特に投影する必要はない．また，球を描画する際，ラバーバンド表示を行う．球のラバーバンド表示では，配置点を指定後，球の半径を指定するまで，マウスの移動を検知し，3 次元座標を算出し，球を描画する．

球の描画方法として，gluSphere 関数を使用して球を描画する．gluSphere 関数の定義を次に示す．

```
void gluSphere(GLUquadricObj *qobj, GLdouble radius, GLint slices, GLint stacks)
```

gluSphere 関数は 4 個の引数をとり，戻り値はない．gluSphere 関数の引数を次に示す．

引数名	型	引数の意味

qobj	GLUquadricObj*	球描画用オブジェクト
radius	GLdouble	球の半径
slices	GLint	経度方向の分割数
stacks	GLint	緯度方向の分割数

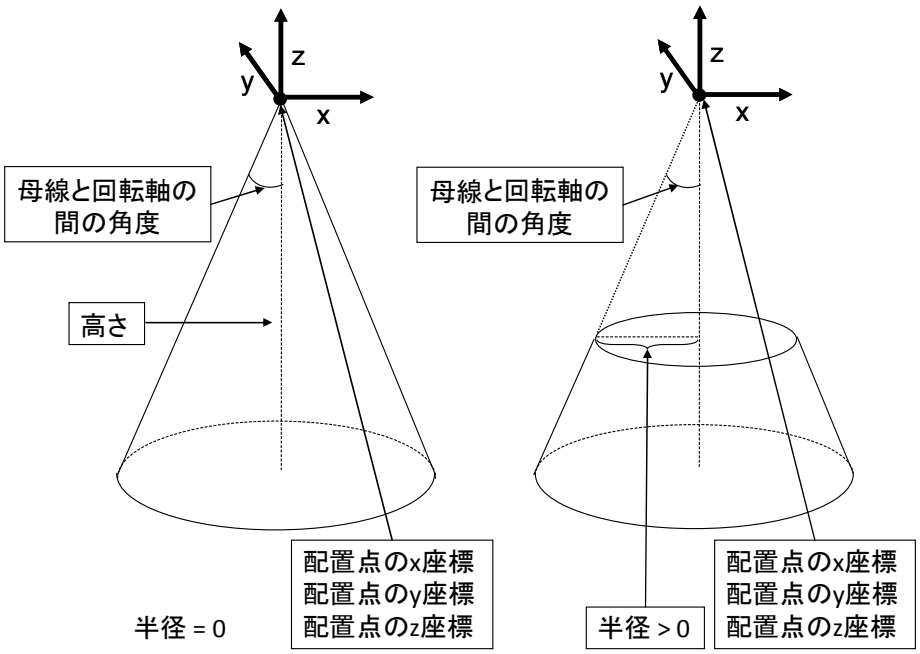
また，球においても，ワイヤフレームモデルとサーフェスモデルに切り替えを可能にする．

4.6.14 円錐

本節では，円錐の描画方法について説明する．

幾何情報

円錐の幾何情報を次に示す．

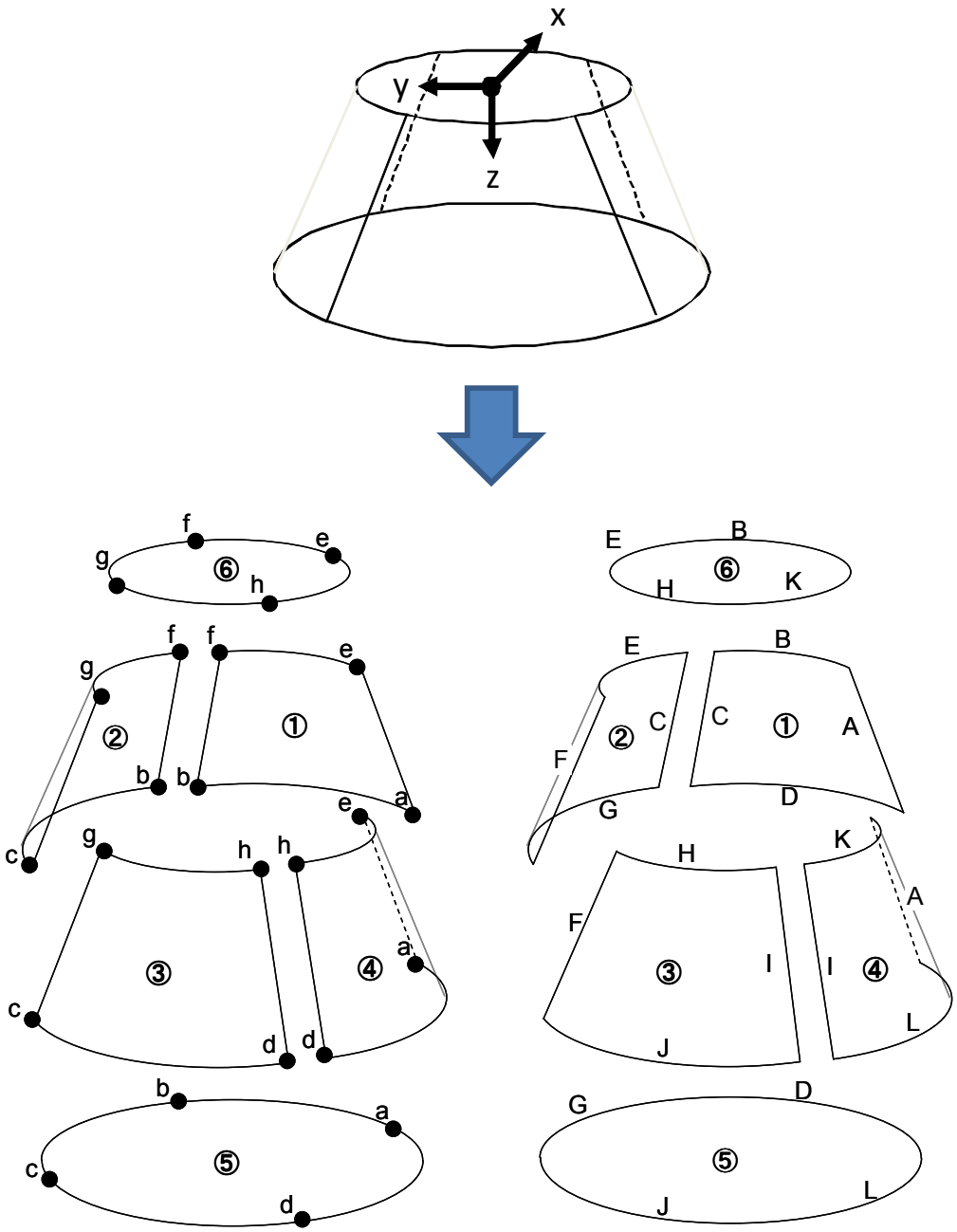


位相情報

本システムでは，円錐の位相情報として，頂点，稜線と面に関する情報を保持する．円錐の位相情報を次に示す．

パラメータ	型	説明
m_hFace	Hashtable	面に関する情報

円錐では，8 個の頂点，12 個の稜線と 6 個の面の位相情報を設定する．円錐の位相情報を次に示す．



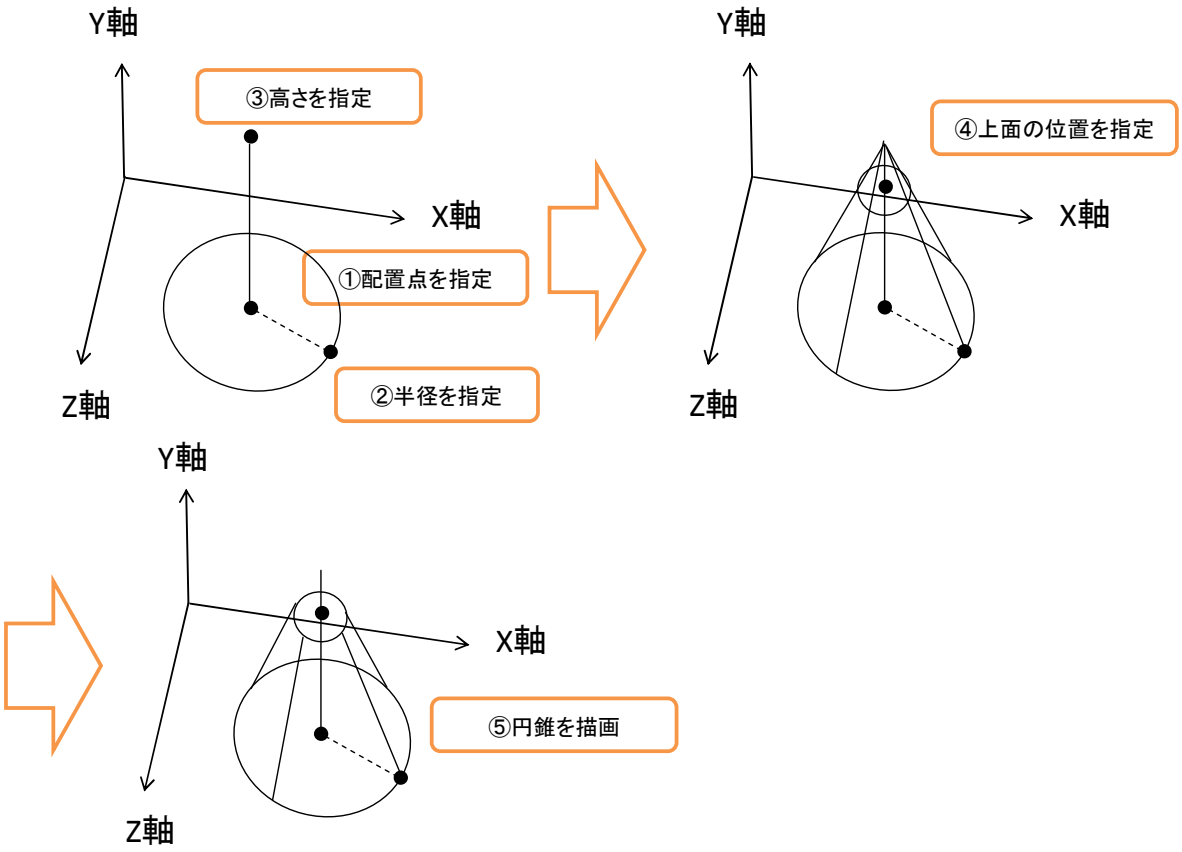
円錐の面，稜線と頂点の対応を次に示す．

面番号	稜線	頂点
1	D，C，B，A	a，b，f，e

2	G，F，E，C	b，c，g，f
3	J，I，H，F	c，d，h，g
4	L，A，K，I	d，a，e，h
5	D，G，J，L	a，b，c，d
6	B，E，H，K	e，f，g，h

描画方法

本システムでは，モデル空間上においてマウス操作で円錐を描画する．円錐の描画手順を次に示す．



本システムでは，まず，配置点（X ,Y ,Z）をマウス操作で指定する．次に，円錐の底面の半径を表す点をマウス操作で指定する．そして，円錐の高さを表す点をマウス操作で指定する．次に，円錐の上面を表す点をマウス操作で指定する．最後に，これらの情報を基に円錐を描画する．ただし，配置点は，座標軸から構成される平面に投影し，半径を表す点は，配置点と同一平面に投影する．そして，高さと上面の位置を表す点は，配置点を投影した平面に直行するようにする．また，円錐を描画する際，ラバーバンド表示を行う．

円錐のラバーバンド表示では，まず，配置点を指定後，半径を指定するまで，マウスの移動を検出し，マウスの 3 次元情報を算出し，円を描画する．次に，高さを指定するまで，同様に，マウスの移動を検出し，マウスの 3 次元情報を算出し，円錐を描画する．最後に，上面の位置を指定するまで，同様に，マウスの移動を検出し，マウスの 3 次元情報を算出し，円錐を描画する．

円錐の描画方法として，gluCylinder 関数を使用して円錐を描画する．gluCylinder 関数の定義を次に示す．

```
void gluCylinder(GLUquadricObj *qobj, GLdouble baseRadius, GLdouble topRadius, GLdouble height, GLint slices, GLint stacks)
```

gluCylinder 関数は 6 個の引数をとり，戻り値はない．gluCylinder 関数の引数を次に示す．

引数名	型	引数の意味
qobj	GLUquadricObj*	オブジェクトポインタ
baseRadius	GLdouble	底面の半径
topRadius	GLdouble	上面の半径
height	GLdouble	円錐の高さ
slices	GLint	経度方向の分割数
stacks	GLint	緯度方向の分割数

また，gluCylinder 関数は，円錐の底面と上面を描画することができないため，glVertex3d 関数を使用して円錐の底面と上面を描画する．

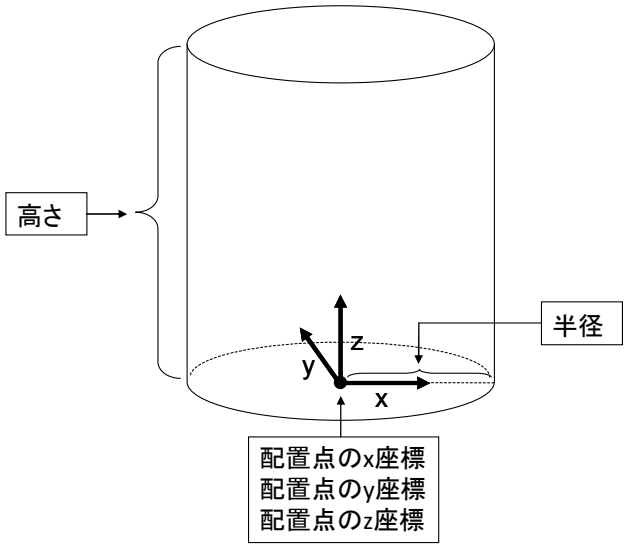
さらに，円錐においても，ワイヤフレームモデルとサーフェスモデルに切り替えを可能にする．

4.6.15　円柱

本節では，円柱の描画方法について説明する．

幾何情報

円柱の幾何情報を次に示す．

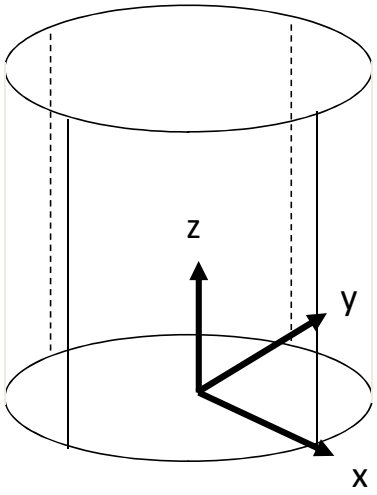


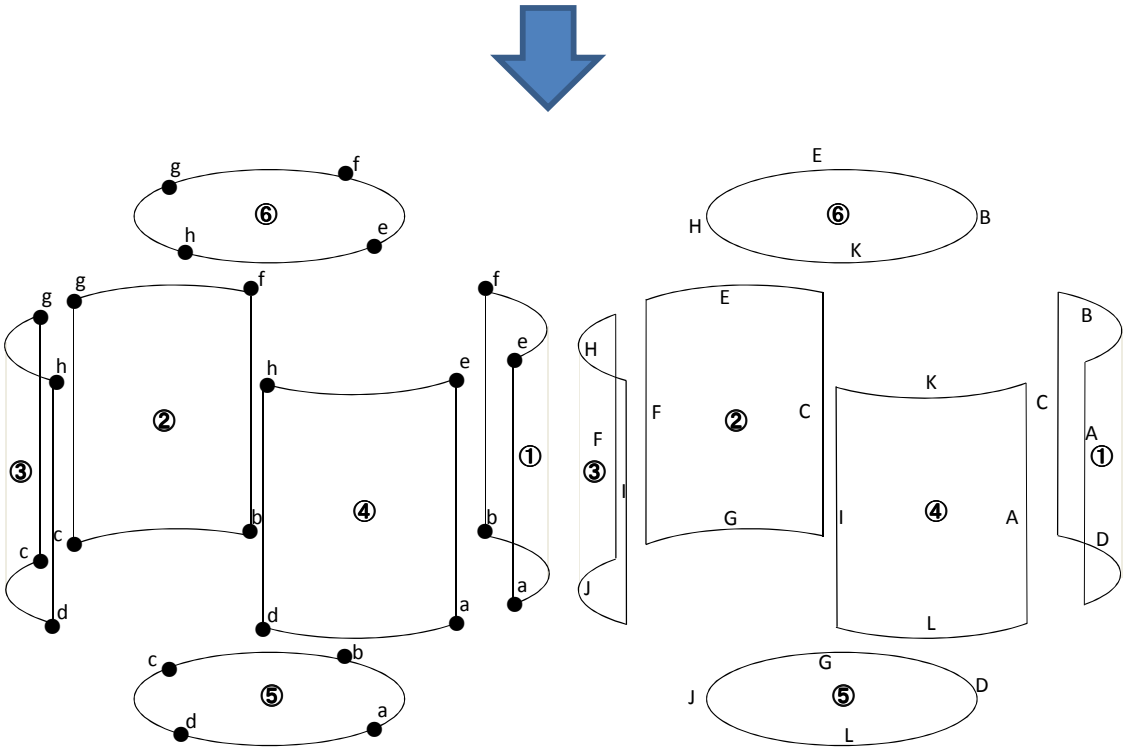
位相情報

本システムでは，円柱の位相情報として，頂点，稜線と面に関する情報を保持する．円柱の位相情報を次に示す.

パラメータ	型	説明
m_hFace	Hashtable	面に関する情報

円柱では，8 個の頂点，12 個の稜線と 6 個の面の位相情報を設定する．円柱の位相情報を次に示す.



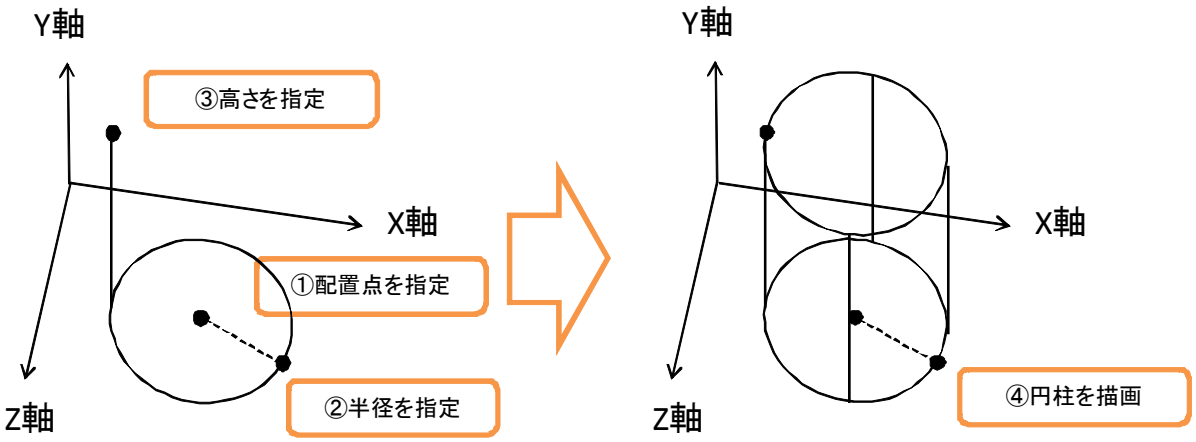


円柱の面，稜線と頂点の対応を次に示す.

面番号	稜線	頂点
1	D，C，B，A	a，b，f，e
2	G，F，E，C	b，c，g，f
3	J，I，H，F	c，d，h，g
4	L，A，K，I	d，a，e，h
5	D，G，J，L	a，b，c，d
6	B，E，H，F	e，f，g，h

描画方法

本システムでは，モデル空間上においてマウス操作で円柱を描画する．円柱の描画手順を次に示す.



本システムでは，まず，配置点（X ,Y ,Z）をマウス操作で指定する．次に，円柱の底面の半径を表す点と円柱の高さを表す点をマウス操作で指定する．最後に，これらの情報を基に円柱を描画する．ただし，配置点は，座標軸から構成される平面に投影し，半径は配置点と同一平面に投影する．そして，高さを表す点は，配置点を投影した平面に直行するようにする．また，円柱を描画する際，ラバーバンド表示を行う．円柱のラバーバンドでは，まず，配置点を指定後，半径を指定するまで，マウスの移動を検出し，マウスの 3 次元座標を算出する．そして，配置点と半径を用いて円を描画する．次に，高さを指定するまで，同様に，マウスの移動を検出し，マウスの 3 次元座標を算出する．そして，高さの値を用いて円柱を描画する．

円柱の描画方法として，gluCylinder 関数を使用して円柱を描画する．ただし，gluCylinder 関数は，円柱の底面と上面を描画することができないため，glVertex3d 関数を使用して円柱の底面と上面を描画する．

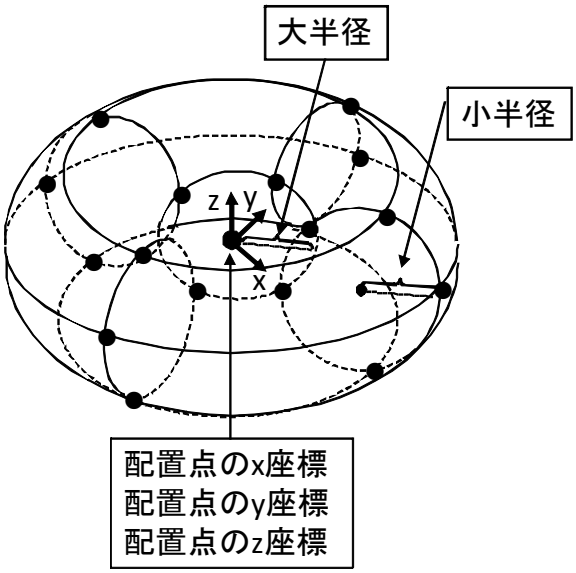
また，円柱においても，ワイヤフレームモデルとサーフェスモデルに切り替えを可能にする．

4.6.16　トーラス

本節では，トーラスの描画方法について説明する．

幾何情報

トーラスの幾何情報を次に示す．

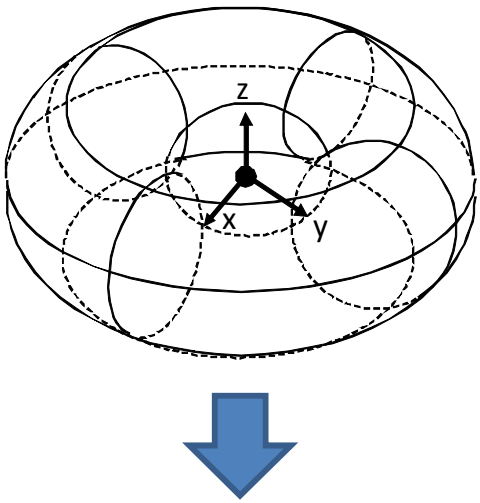


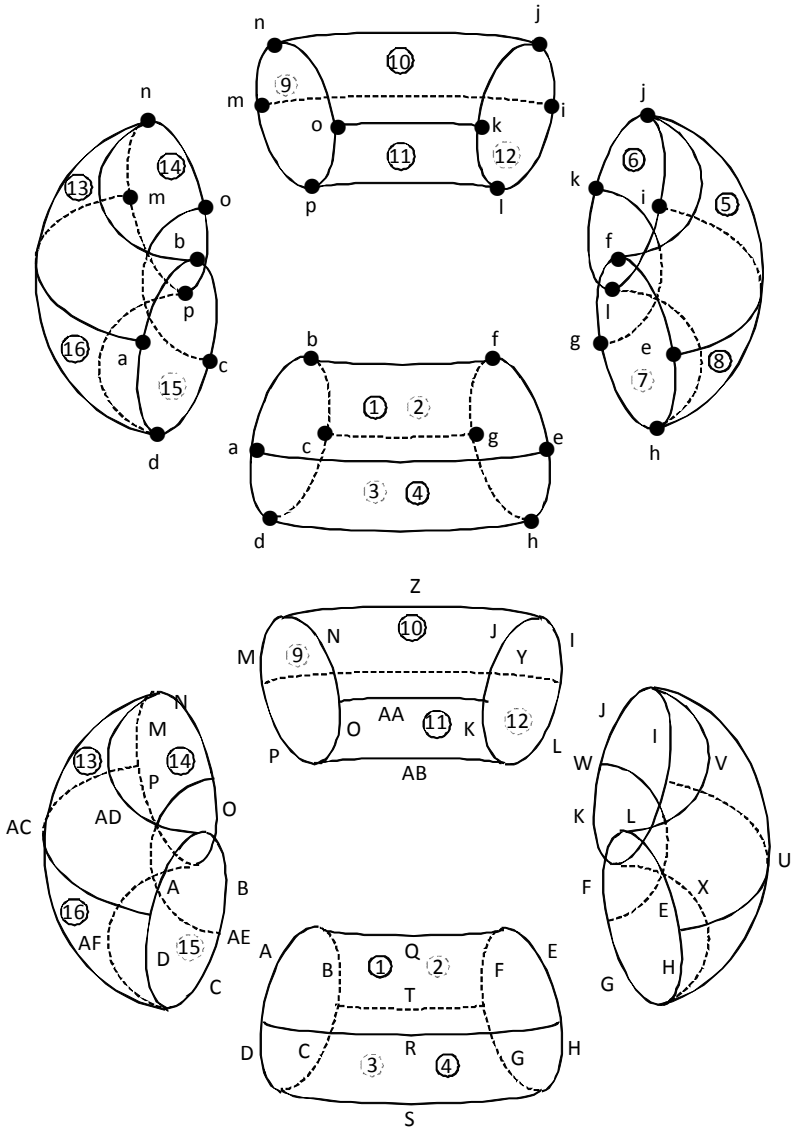
位相情報

本システムでは，トーラスの位相情報として，頂点，稜線と面に関する情報を保持する．トーラスの位相情報を次に示す．

パラメータ	型	説明
m_hFace	Hashtable	面に関する情報

トーラスでは，16 個の頂点，32 個の稜線と 16 個の面の位相情報を設定する．トーラスの位相情報を次に示す．





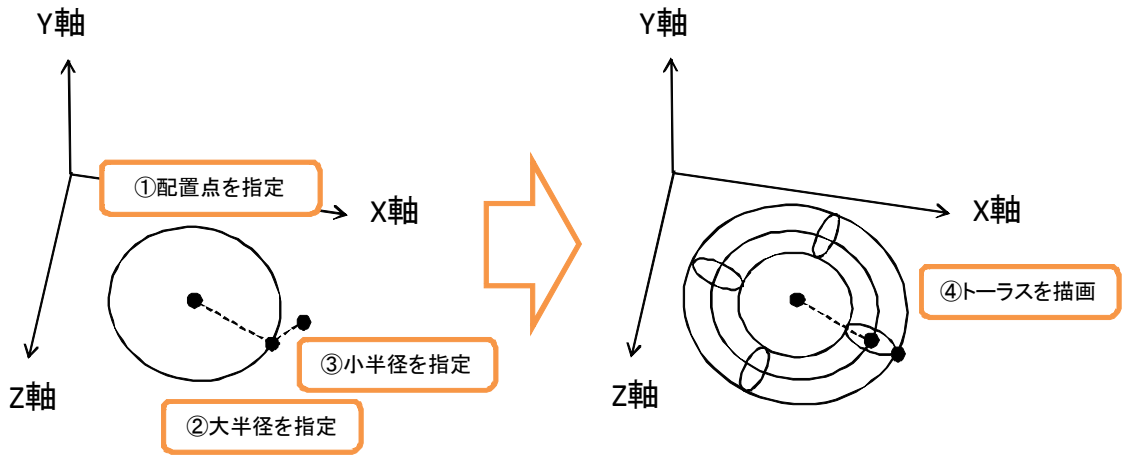
トーラスの面，稜線と頂点の対応を次に示す.

面番号	稜線	頂点
1	A, Q, E, R	a, b, f, e
2	B, T, F, Q	b, c, g, f
3	C, S, G, T	c, d, h, g
4	D, R, H, S	d, a, e, h
5	E, V, I, U	e, f, j, i
6	F, W, J, V	f, g, k, j
7	G, X, K, W	g, h, l, k
8	H, U, L, X	h, e, i, l

9	I, Z, M, Y	i, j, n, m
10	J, AA, N, Z	j, k, o, n
11	K, AB, O, AA	k, l, p, o
12	L, Y, P, AB	l, i, m, p
13	M, AD, A, AC	m, n, b, a
14	N, AE, B, AD	n, o, c, b
15	O, AF, C, AE	o, p, d, c
16	P, AC, D, AF	p, m, a, d

描画方法

本システムでは，モデル空間上においてマウス操作でトーラスを描画する．トーラスの描画手順を次に示す．



本システムでは，まず，配置点（X ,Y ,Z）をマウス操作で指定する．次に，大半径を表す点と小半径を表す点をマウス操作で指定する．最後に，これらの情報を基にトーラスを描画する．ただし，配置点，大半径と小半径を表す点は，座標軸上から構成される平面に投影する．また，トーラスの描画の際，ラバーバンド表示を行う．トーラスのラバーバンド表示では，まず，配置点を指定後，大半径を指定するまで，マウスの移動を検知し，マウスの 3 次元座標を算出する．そして，配置点と大半径の値を用いて円を描画する．次に，小半径を指定するまで，同様に，マウスの移動を検出し，マウスの 3 次元座標を算出する．そして，配置点，大半径と小半径の値を用いて，トーラスを描画する．

次に，トーラスの描画方法として，glutSolidTorus 関数を使用してトーラスを描画する．glutSolidTorus 関数の定義を次に示す．

```
void glutSolidTorus (GLdouble innerRadius, GLdouble outerRadius, GLint sides, GLint rings)
```

glutSolidTorus 関数は 4 個の引数をとり，戻り値はない．glutSolidTorus 関数の引数を次に示す．

引数名	型	引数の意味
innerRadius	GLdouble	配置点から回転面の中心までの半径
outerRadius	GLdouble	回転面の半径
sides	GLint	経度方向の分割数
rings	GLint	緯度方向の分割数

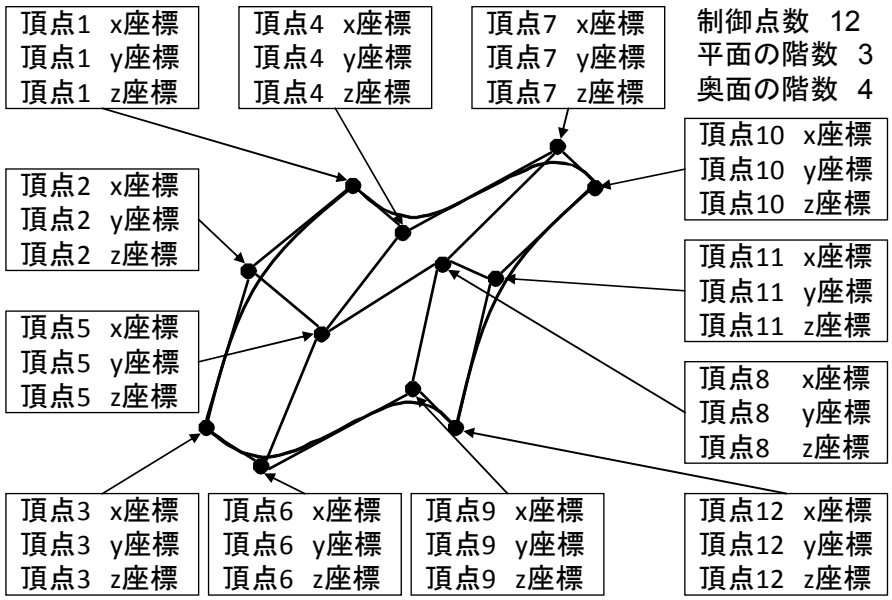
また，トーラスにおいても，ワイヤフレームモデルとサーフェスモデルに切り替えを可能にする．

4.6.17　ベジェ曲面

本節では，ベジェ曲面の描画方法について説明する．

幾何情報

ベジェ曲面の幾何情報を次に示す．



制御点数とは，作成するベジェ曲線全体で必要となる頂点の数である．階数とは，1 次のベジェ曲線を作成する際に必要となる頂点の数である．N 次の階数で構成されるベジェ曲線

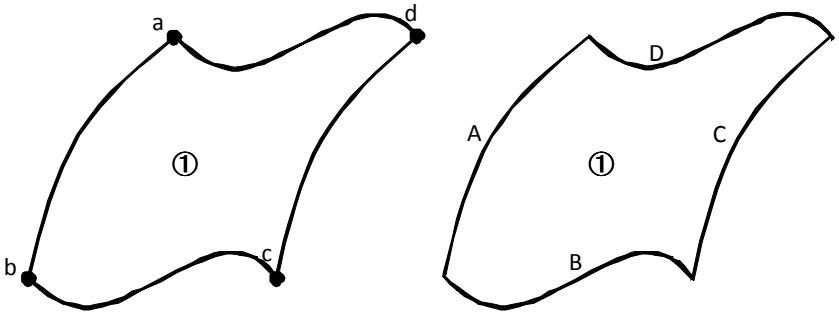
を M 個作成する場合，制御点数は，(N-1)*M+1 個になる．

位相情報

本システムでは，ベジェ曲面の位相情報として，頂点，稜線と面に関する情報を保持する．ベジェ曲面の位相情報を次に示す．

パラメータ	型	説明
m_hFace	Hashtable	面に関する情報

本システムでは，ベジェ曲面の位相情報として，頂点，稜線と面に関する情報を保持する．ベジェ曲面の位相情報を次に示す．

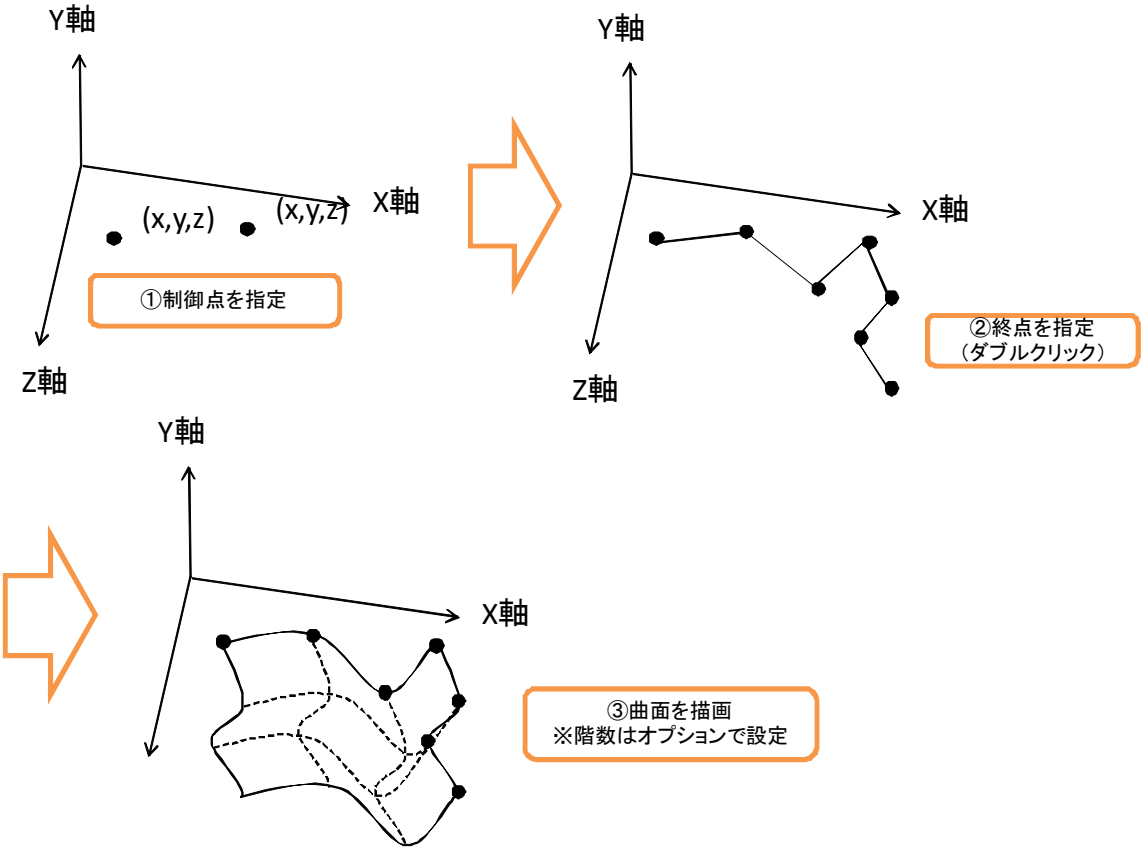


ベジェ曲面の面，稜線と頂点の対応を次に示す．

面番号	稜線	頂点
1	A，B，C，D	a，b，c，d

描画方法

本システムでは，モデル空間上においてマウス操作でベジェ曲面を描画する．ベジェ曲面の描画手順を次に示す．



本システムでは，(平面の階数+奥面の階数)-1 の数の制御点を指定することで，ベジェ曲面を描画する．まず，各制御点（X ,Y ,Z）をマウス操作で指定する．次に，終点（X ,Y ,Z）をマウス操作（ダブルクリックイベントを使用）で指定する．最後に，各制御点からベジェ曲面を描画する．ただし，各制御点の投影方法は，折線と同様の方法とする．また，ベジェ曲面の階数については，オプションで指定するものとする．また，ベジェ曲面を描画する際，ラバーバンド表示を行う．ベジェ曲面のラバーバンド表示では，折線と同様のラバーバンド表示を行う．具体的には，終点を指定するまで，ベジェ曲面を描画するのではなく，制御点を頂点とする折線を描画する．

ベジェ曲面の描画方法として，glMap2d 関数を使用してベジェ曲面を描画する．glMap2d 関数の定義を次に示す．

```
void glMap2d( GLenum target, GLdouble u1, GLdouble u2, GLint ustride, GLint uorder,
              GLdouble v1, GLdouble v2, GLint vstride, GLint vorder, const GLdouble * points)
```

glMap2d 関数は 10 個の引数をとり，戻り値はない．glMap2d 関数の引数を次に示す．

引数名	型	引数の意味

target	GLenum	エバリュータで生成する値の種類
u1	GLdouble	平面方向のパラメータの計算方法
u2	GLdouble	平面方向のパラメータの計算方法
ustride	Glint	平面方向の曲線の始点と終点に用いる要素
uorder	Glint	平面方向の階数
v1	GLdouble	奥行方向のパラメータの計算方法
v2	GLdouble	奥行方向のパラメータの計算方法
vstride	GLint	奥行方向の曲線の始点と終点に用いる要素
vorder	GLint	奥行方向の階数
points	GLdouble	全ての制御点の配列

次に, glMap2d 関数で設定した曲面を有効化するために glEnable 関数を使用する. glEnable 関数の定義を次に示す.

```
void glEnable(GLenum target)
```

glEnable 関数は 1 個の引数を取り，戻り値はない．glEnable 関数の引数を次に示す.

引数名	型	引数の意味
target	GLenum	OpenGL の特定機能を示す定数

本システムでは，target に GL_MAP2_VERTEX_3 と GL_MAP2_VERTEX_4 を設定することで，頂点を生成する際に平面の法線ベクトルを算出する.

最後に，glEvalCoord2d 関数を利用して，ベジェ曲面を描画する．glEvalCoord2d 関数の u と v には 0 から 1 までの数字が入る．それぞれ，0 を設定すると始点，1 を設定すると終点を描画する．glEvalCoord2d 関数の定義を次に示す.

```
void glEvalCoord2d( GLdouble u, GLdouble v)
```

glEvalCoord2d 関数は 2 つの引数を取り，戻り値はない．glEvalCoord2d 関数の引数を次に示す.

引数名	型	引数の意味
u	GLdouble	平面方向の制御点の始点と終点
v	GLdouble	奥行方向の制御点の始点と終点

ベジェ曲面の描画方法の例を次に示す.

```
// ベジェ曲面を複数描画するための繰り返し
for(int i=0;i<(((m_iNumber/m_iOrder1)-1)/(m_iOrder2-1));i++){
    // i 番目のベジェ曲面を設定
    glMap2d(GL_MAP2_VERTEX_3, 0, 1, 3, m_iOrder1, 0, 1,
                3*(m_iOrder1), m_iOrder2, dCtlPoint[i]);
    // ベジェ曲面の有効化
    glEnable(GL_MAP2_VERTEX_3);

    // 分割の指定
    double slice=0.05;

    glBegin(GL_QUADS);
    // 奥行方向描画の繰り返し
    for(double v = 0; v < 1; v += slice){
        // 平面方向の描画の繰り返し
        for(double u = 0; u < 1; u += slice){
            // 四角形 ABCD の四隅を指定
            glEvalCoord2d(u,v);             // 頂点 A
            glEvalCoord2d(u+slice,v);    // 頂点 B
            glEvalCoord2d(u+slice,v+slice);// 頂点 C
            glEvalCoord2d(u,v+slice);    // 頂点 D
        }
    }
    // 描画の終了
    glEnd();
    glDisable(GL_MAP2_VERTEX_3);
}
```

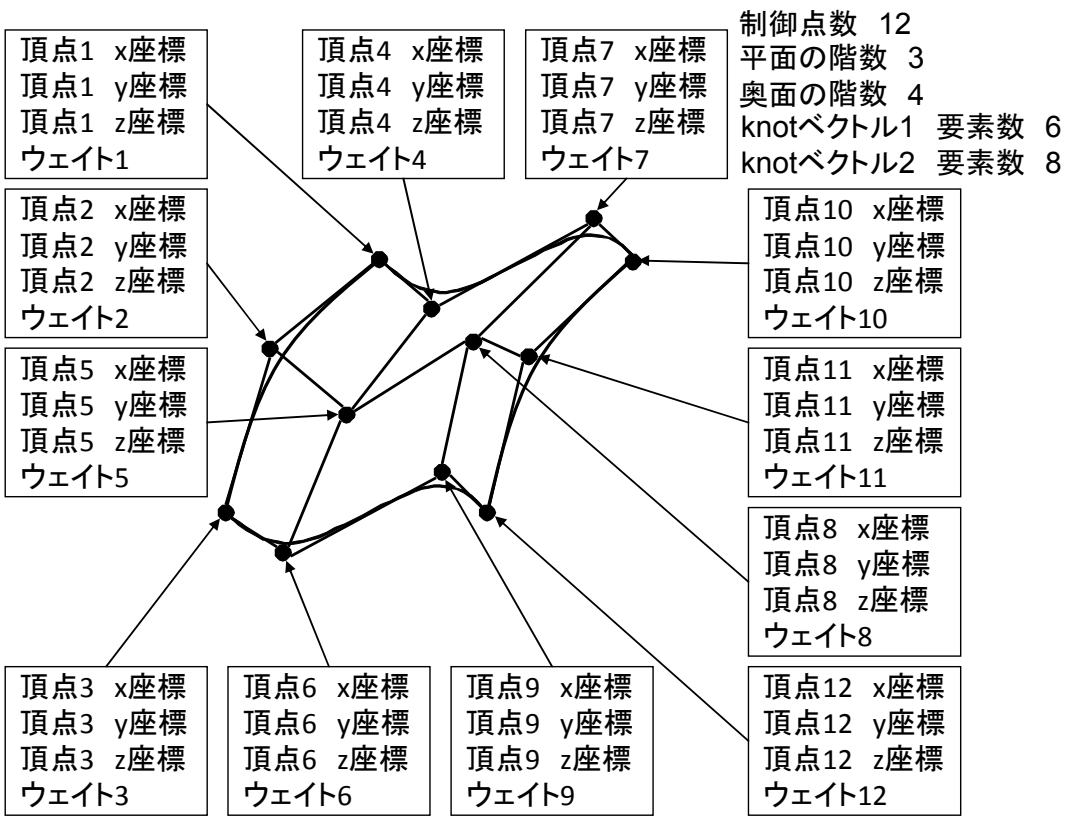
また，ベジェ曲面においても，ワイヤフレームモデルとサーフェスモデルに切り替えを可能にする.

4.6.18 NURBS曲面

本節では，NURBS 曲面の描画方法について説明する.

幾何情報

NURBS 曲面の幾何情報を次に示す.

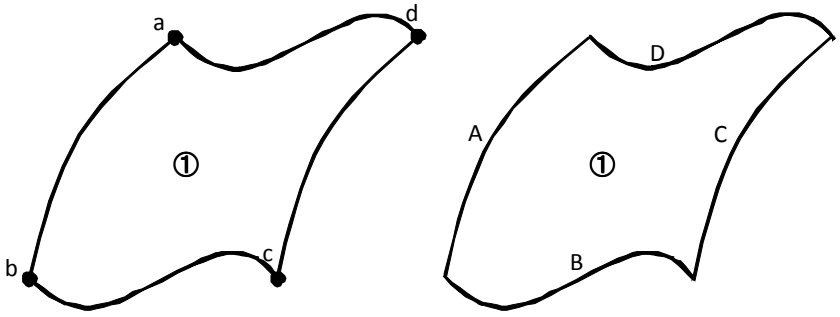


位相情報

本システムでは，NURBS 曲面の位相情報として，頂点，稜線と面に関する情報を保持する．NURBS 曲面の位相情報を次に示す.

パラメータ	型	説明
m_hFace	Hashtable	面に関する情報

本システムでは，NURBS 曲面の位相情報として，頂点，稜線と面に関する情報を保持する．NURBS 曲面の位相情報を次に示す.

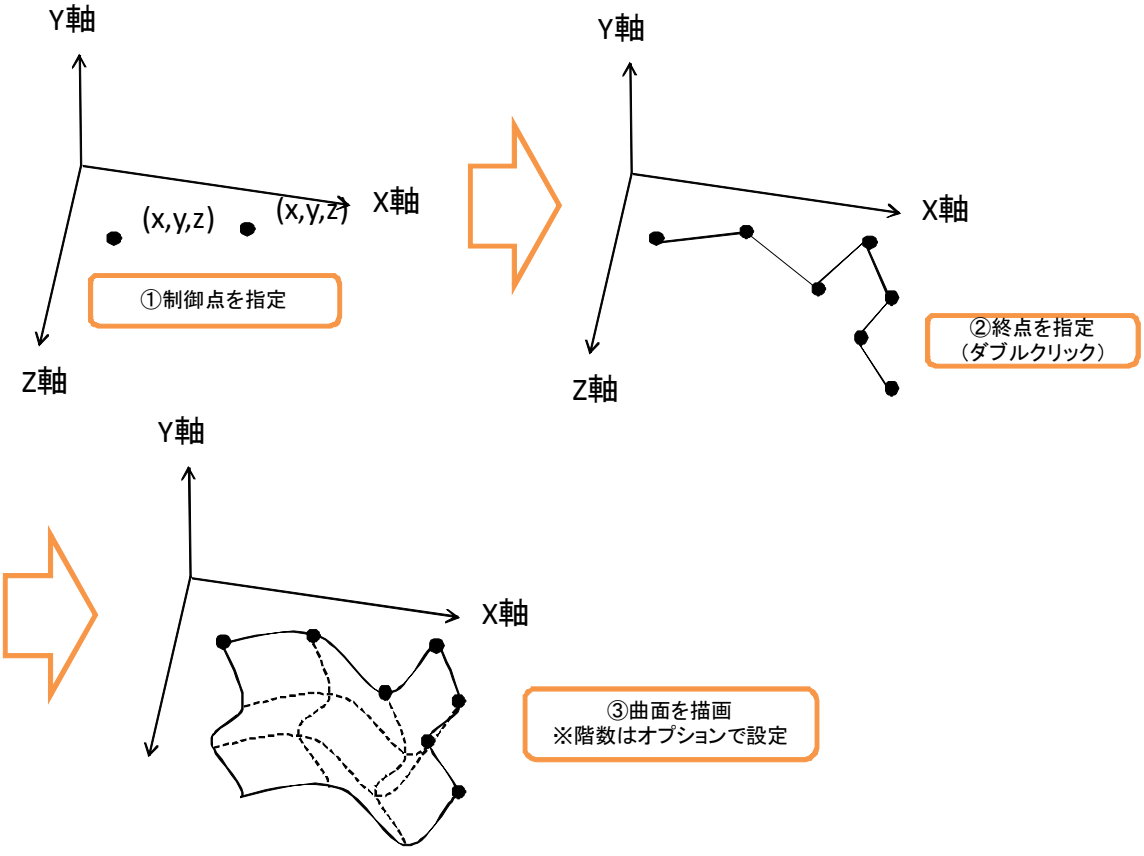


NURBS 曲面の面，稜線と頂点の対応を次に示す．

面番号	稜線	頂点
1	A，B，C，D	a，b，c，d

描画方法

本システムでは，モデル空間上においてマウス操作で NURBS 曲面を描画する．NURBS 曲面の描画手順を次に示す．



本システムでは，(平面の階数+奥面の階数)-1 の数の制御点を指定することで，NURB 曲面を描画する．まず，各制御点（X ,Y ,Z）をマウス操作で指定する．次に，終点（X ,Y ,Z）をマウス操作（ダブルクリックイベントを使用）で指定する．最後に，各制御点から NURBS 曲面を描画する．ただし，各制御点の投影方法は，折線と同様の方法とする．また，NURBS 曲面の階数については，オプションで指定するものとする．また，NURBS 曲面を描画する際，ラバーバンド表示を行う．NURBS 曲面のラバーバンド表示では，折線と同様のラバー

バンド表示を行う. 具体的には, 終点を指定するまで, NURBS 曲面を描画するのではなく, 制御点を頂点とする折線を描画する.

NURBS 曲面の描画方法として，gluNurbsProperty 関数を使用して NURBS 曲面を描画する．gluNurbsProperty 関数の定義を次に示す.

```
void gluNurbsProperty(GLUnurbs *nobj, GLenum property, GLfloat value);
```

gluNurbsProperty 関数は 3 個の引数をとり，戻り値はない．gluNurbsProperty 関数の引数を次に示す.

引数名	型	引数の意味
nobj	GLUnurbs	NURBS オブジェクトの指定
property	GLenum	設定するプロパティ
value	GLfloat	設定したプロパティの値

本システムでは，property に GLU_U_STEP と GLU_V_STEP を設定することで，平面方向と奥行方向のサンプル点の個数を指定する．また，GLU_SAMPLING_TOLERANCE を設定することでサンプリング範囲を指定する．サンプリングの値が小さいほど滑らかな曲面を描画する.

また，gluNurbsSurface 関数を利用して，NURBS 曲面を描画する．gluNurbsSurface 関数の定義を次に示す.

```
void gluNurbsSurface(GLUnurbs *nobj, GLint sKnotCount, float *sKnots, GLint tKnotCount,
                GLfloat *tKnots, GLint sStride, GLint tStride, GLfloat *control,
                GLint sOrder, GLint tOrder, GLenum type);
```

gluNurbsSurface 関数は 11 個の引数をとり，戻り値はない．gluNurbsSurface 関数の引数を次に示す.

引数名	型	引数の意味
nobj	GLUnurbs	NURBS オブジェクトの指定
sKnotCount	GLint	平面方向のノット数を指定
sKnots	GLfloat	平面方向のノットの配列を指定
tKnotCount	GLint	奥行方向のノット数を指定
tKnots	GLfloat	奥行方向のノットの配列を指定
sStride	GLint	平面方向に連続する制御点の間のオフセットを指定

tStride	GLint	奥行方向に連続する制御点の間のオフセットを指定
control	GLfloat	NURBS 曲面の制御点の配列を指定
sOrder	GLint	平面方向の階数の指定
tOrder	GLint	奥行方向の階数の指定
type	GLenum	曲面の種類を指定

NURBS 曲面の描画方法の例を次に示す.

```
// Nurbs オブジェクトの宣言
GLUnurbsObj *theNurb;
// Nurbs オブジェクト作成
theNurb = gluNewNurbsRenderer();

// u 方向のサンプリング範囲の指定
gluNurbsProperty(theNurb, GLU_U_STEP,0.1);
// v 方向のサンプリングの指定
gluNurbsProperty(theNurb, GLU_V_STEP,0.1);

// Nurbs オブジェクトを指定し描画
gluBeginSurface(theNurb);
// Nurbs オブジェクトの設定
gluNurbsSurface(theNurb,m_alKnot1->Count,fKnot1,m_alKnot2->Count,

        fKnot2,4,4*(m_iOrder1),fCtlPoint,m_iOrder1,m_iOrder2,GL_MAP2_VERTEX_4);
// Nurbs オブジェクトの描画の終了
gluEndSurface(theNurb);
```

また，NURBS 曲面においても，ワイヤフレームモデルとサーフェスモデルに切り替えを可能にする.

4.7　特殊のモデルの作成

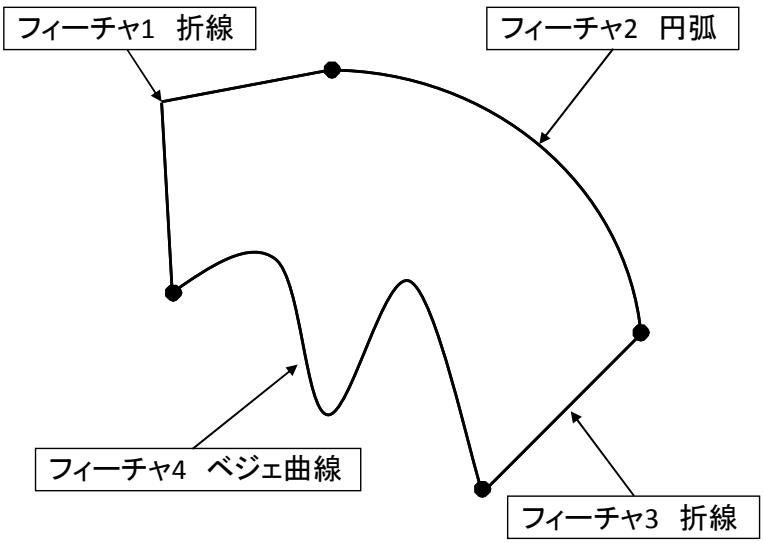
本章では，特殊なモデルの作成として，複合曲線，押出面，掃引面と回転面の作成方法について説明する.

4.7.1　複合曲線

本節では，複合曲線の作成方法について説明する．本節では，まず，複合曲線の幾何情報について説明する．次に，複合曲線の位相情報について説明する．最後に，複合曲線の描画方法について説明する.

幾何情報

複合曲線の幾何情報を次に示す.



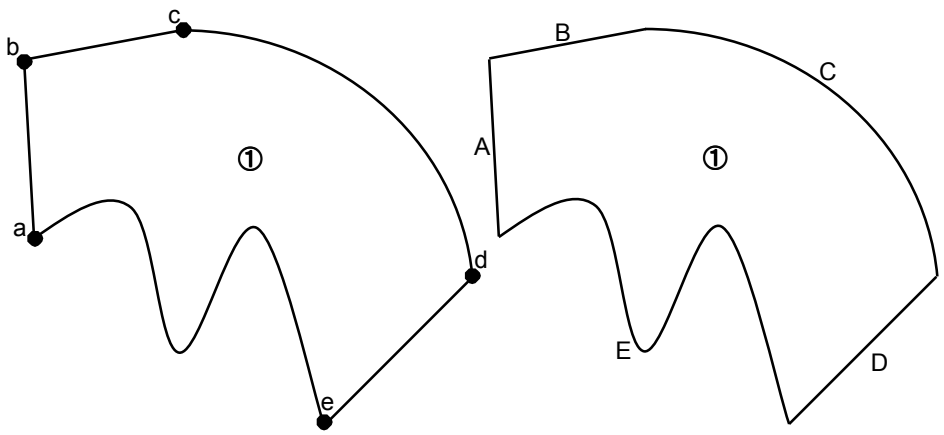
複合曲線は，複数のフィーチャを組み合わせて面を描画する．複合曲線で利用できるモデルは，折線，円弧，楕円弧，ベジェ曲線とする．

位相情報

本システムでは，複合曲線の位相情報として，頂点，稜線と面に関する情報を保持する．複合曲線の位相情報を次に示す．

パラメータ	型	説明
m_hFace	Hashtable	面に関する情報

複合曲線では，各フィーチャの位相情報を統合する．例えば，折線が 2 本とベジェ曲線が 2 本の場合の複合曲線の位相情報を次に示す．

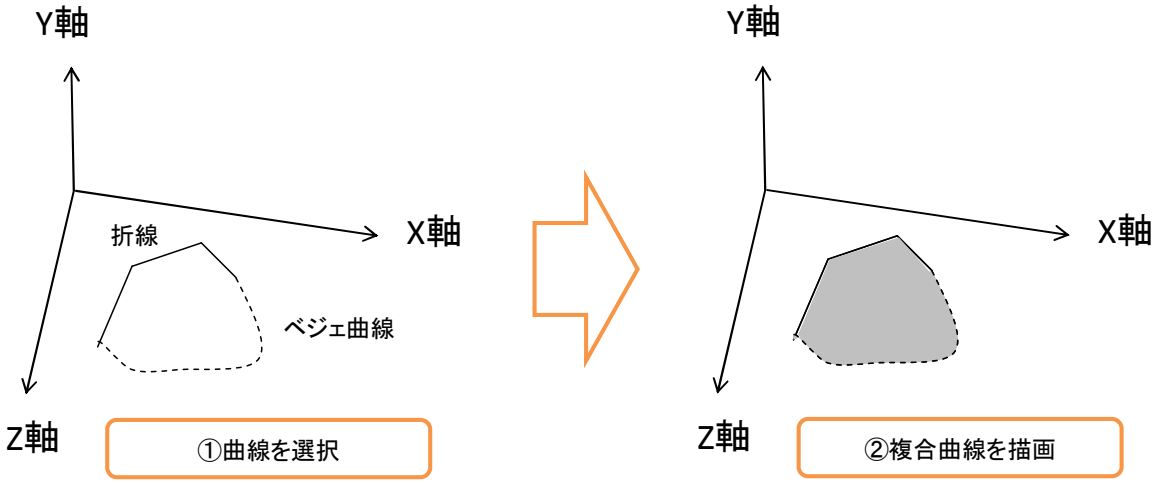


複合曲線の面，稜線と頂点の対応を次に示す．

面番号	稜線	頂点
1	A，B，C，D，E	a，b，c，d，e

描画方法

複合曲線の作成方法として，上記で記述した曲線を選択することで面を描画する．複合曲線の作成方法を次に示す．



本システムでは，複合曲線を構成する曲線をマウス操作で選択する．選択終了後（ダブルクリックで判定とする．ただし，ダブルクリック時はモデル空間上でよいとする．），選択した曲線で複合曲線を描画する．

本システムにおける複合曲線では，すべての曲線を折線に近似した頂点座標を算出し，

glVertex3d 関数を使用して描画する．

また，複合曲線は，描画する面を正確に分割する必要がある．そのため，本システムでは，OpenGL の GLUtesselator を利用して面を分割する．GLUtesselator を利用するには，gluNewTess 関数を利用して GLUtesselator オブジェクトを生成する． gluNewTess 関数の定義を次に示す．

```
GLUtesselator* gluNewTess();
```

gluNewTess 関数は引数を必要とせず，戻り値に GLUtesselator オブジェクトを返す．さらに，gluTessCallback 関数を利用して面をどのように分割するか設定する．gluTessCallback 関数の定義を次に示す．

```
void gluTessCallback(GLUtriangulatorObj* tobj, GLenum which, void(*fn)());
```

gluTessCallback 関数は 3 個の引数をとり，戻り値はない．gluTessCallback 関数の引数を次に示す．

引数名	型	引数の意味
tobj	GLUtriangulatorObj	GLUtesselator オブジェクトの指定
which	GLenum	定義するコールバックの指定
fn	void*	呼び出される関数を指定

gluTessBeginPolygon 関数と gluTessEndPolygon 関数を利用して面が凸, 非凸や自己干渉かを定義する．gluTessBeginPolygon 関数の定義を次に示す．

```
void gluTessBeginPolygon(GLUtriangulatorObj* tobj, void* data);
```

gluTessBeginPolygon 関数は２個の引数をとり，戻り値はない．gluTessBeginPolygon 関数の引数を次に示す．

引数名	型	引数の意味
tobj	GLUtriangulatorObj	GLUtesselator オブジェクトの指定
data	void	ユーザ指定のポリゴンデータを指すポインタ

gluTessEndPolygon 関数の定義を次に示す.

```
void gluTessEndPolygon (GLUtriangulatorObj* tobj);
```

gluTessEndPolygon 関数は 1 個の引数をとり，戻り値はない．gluTessEndPolygon 関数の引数を次に示す.

引数名	型	引数の意味
tobj	GLUtriangulatorObj	GLUtesselator オブジェクトの指定

gluTessBeginPolygon 関数と gluTessEndPolygon 関数の間には 1 個以上 gluTessBeginContour 関数と gluTessEndContour 関数を呼び出す必要がある．gluTessBeginContour 関数の定義を次に示す.

```
void gluTessBeginContour (GLUtriangulatorObj* tobj);
```

gluTessBeginContour 関数は 1 個の引数をとり，戻り値はない．gluTessBeginContour 関数の引数を次に示す.

引数名	型	引数の意味
tobj	GLUtriangulatorObj	GLUtesselator オブジェクトの指定

gluTessEndContour 関数の定義を次に示す.

```
void gluTessEndContour (GLUtriangulatorObj* tobj);
```

gluTessEndContour 関数は 1 個の引数をとり，戻り値はない．gluTessEndContour 関数の引数を次に示す.

引数名	型	引数の意味
tobj	GLUtriangulatorObj	GLUtesselator オブジェクトの指定

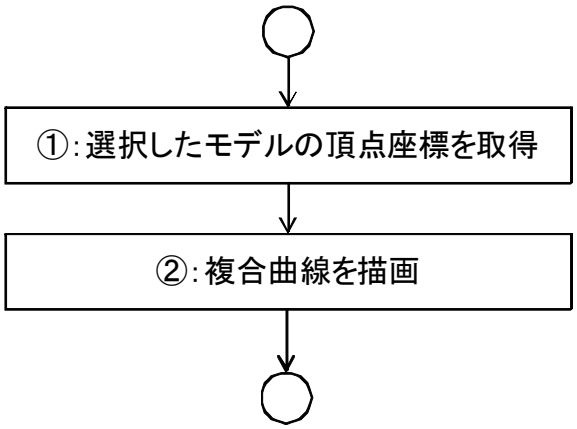
gluTessBeginContour 関数と gluTessEndContour 関数の間で gluTessVertex 関数を利用して面を正確に分割する．gluTessVertex 関数の定義を次に示す.

```
void gluTessVertex( GLUtriangulatorObj* tobj, GLdouble v[3], void* data);
```

gluTessVertex 関数は３個の引数をとり，戻り値はない．gluTessVertex 関数の引数を次に示す．

引数名	型	引数の意味
tess	GLUtriangulatorObj	GLUtesselator オブジェクトの指定
v	GLdouble	頂点の位置を指定
data	void	ユーザ指定のポリゴンデータを指すポインタの指定

複合曲線の描画の処理フローを次に示す．



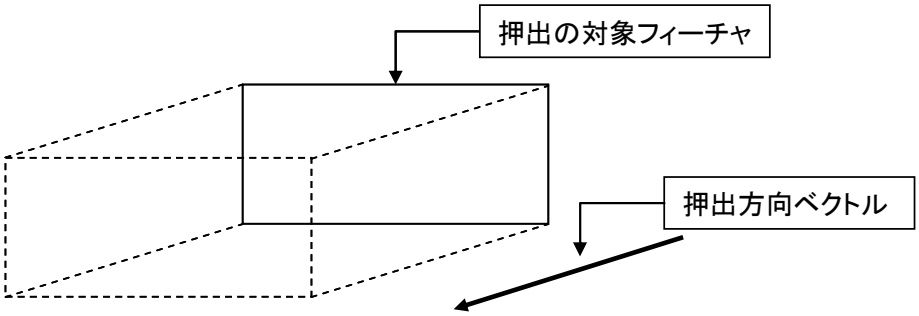
複合曲線を描画するには，まず，選択したモデルを折線に近似し，頂点座標を取得する．次に，頂点座標を基に曲面を作成することで複合曲線を描画する．ただし，選択したモデルが連結してない場合は，強制的に頂点数が 2 の折線を追加し，モデルを連結させることとする．

4.7.2 押出面

本節では，押出面の作成方法について説明する．本節では，まず，押出面の幾何情報について説明する．次に，押出面の位相情報について説明する．最後に，押出面の描画方法について説明する．

幾何情報

押出面の幾何情報を次に示す．

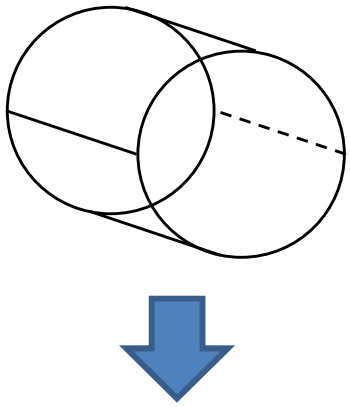


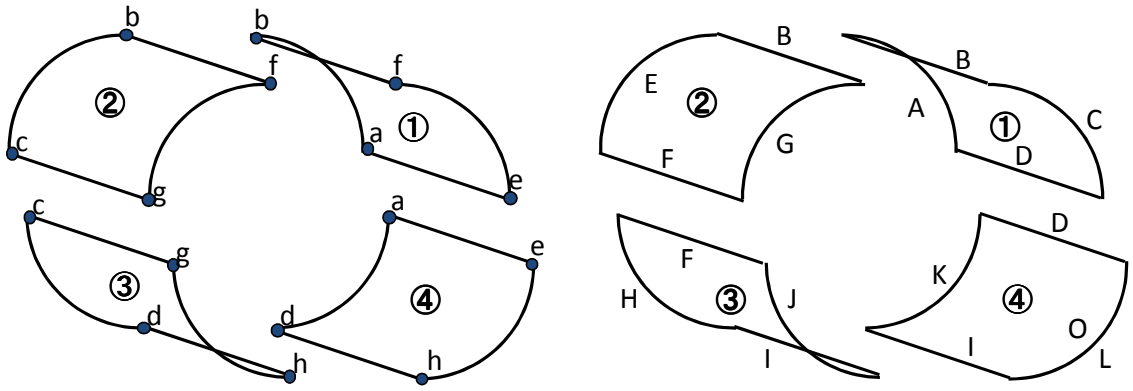
位相情報

本システムでは，押出面の位相情報として，頂点，稜線と面に関する情報を保持する．押出面の位相情報を次に示す．

パラメータ	型	説明
m_hFace	Hashtable	面に関する情報

押出面では，押出の対象フィーチャが持つ位相情報と押出を行い作成した立体の上面の位相情報を基に押出面の位相情報を設定する．押出面の位相情報の例を次に示す．



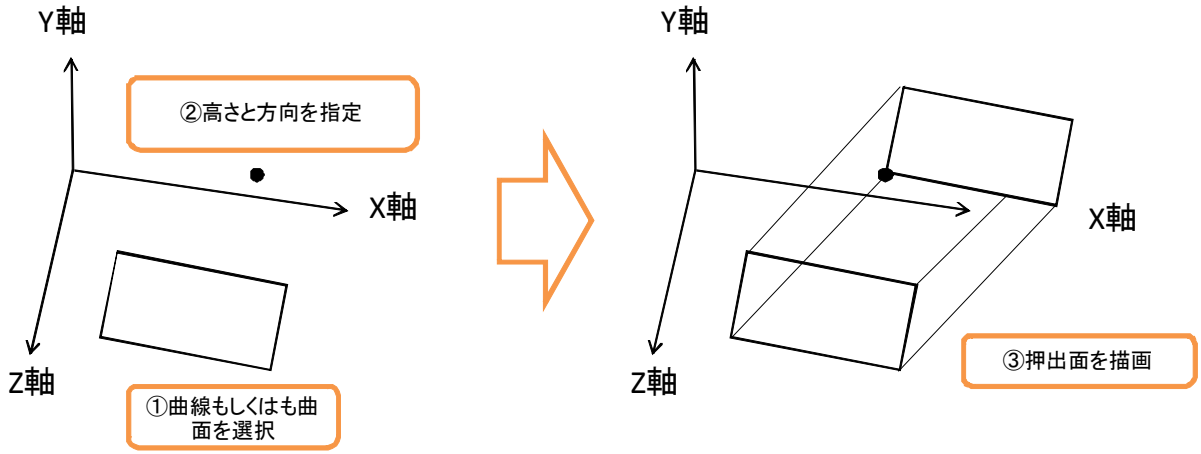


押出面の面，稜線と頂点の対応を次に示す．

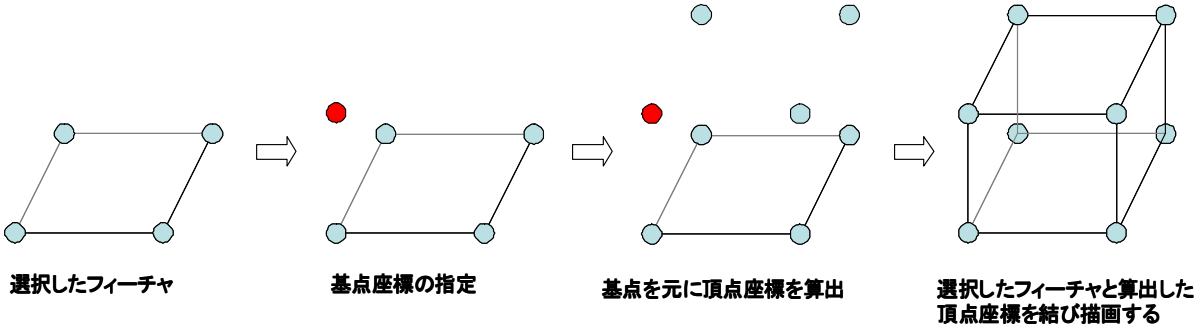
面番号	稜線	頂点
1	A，B，C，D	a，b，f，e
2	E，F，G，B	b，c，g，f
3	H，I，J，F	c，d，h，g
4	K，D，L，I	d，a，e，h

描画方法

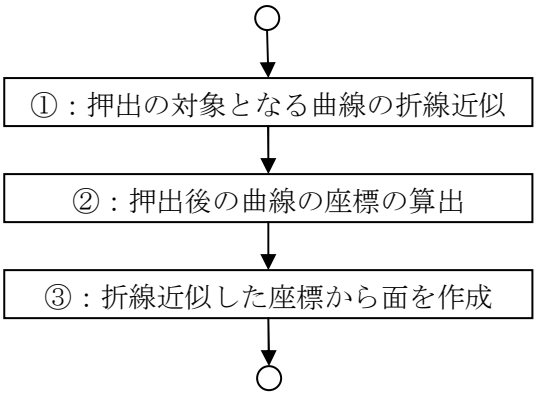
押出面の描画方法として，曲線（面）要素を選択し，押し出す方向と距離を指定することで面を描画する．押出面の描画方法を次に示す．



本システムにおける押出面の描画方法は，選択したモデルから頂点座標を算出する．そして，各頂点を基に面を作成することで描画する．押出面の描画方法を次に示す．



押出面の描画の処理フローを次に示す.



本システムでは，まず，押出の対象となる曲線（曲面）をマウス操作で指定する．次に，押し出し先をマウス操作で指定する．

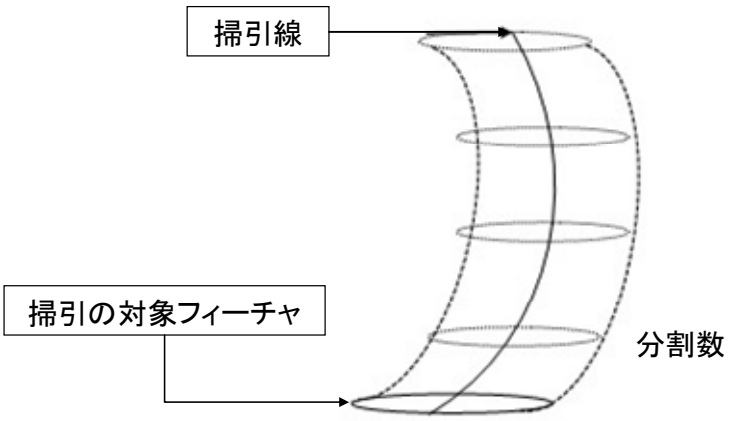
押出面の描画方法は，まず，押出の対象となる曲線の頂点座標を算出する．次に，指定した座標から押出の対象となる曲線（曲面）と平行になる曲線（曲面）を考えて頂点を生成する．そして，生成した各頂点と押出の対象となる曲線（曲面）の頂点から面を作成していき，押出面を構成する．また，押出面の描画の際，ラバーバンド表示を行う．押出面のラバーバンド表示では，押出の対象となる曲線（曲面）を指定後，高さと方向を指定するまで，マウスの移動を検出し，マウスの 3 次元座標を算出する．そして，高さと方向から押出面を描画する．

4.7.3　掃引面

本節では，掃引面の作成方法について説明する．本節では，まず，掃引面の幾何情報について説明する．次に，掃引面の位相情報について説明する．最後に，掃引面の描画方法について説明する．

幾何情報

掃引面の幾何情報を次に示す.

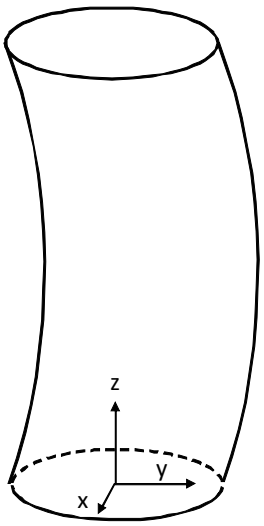


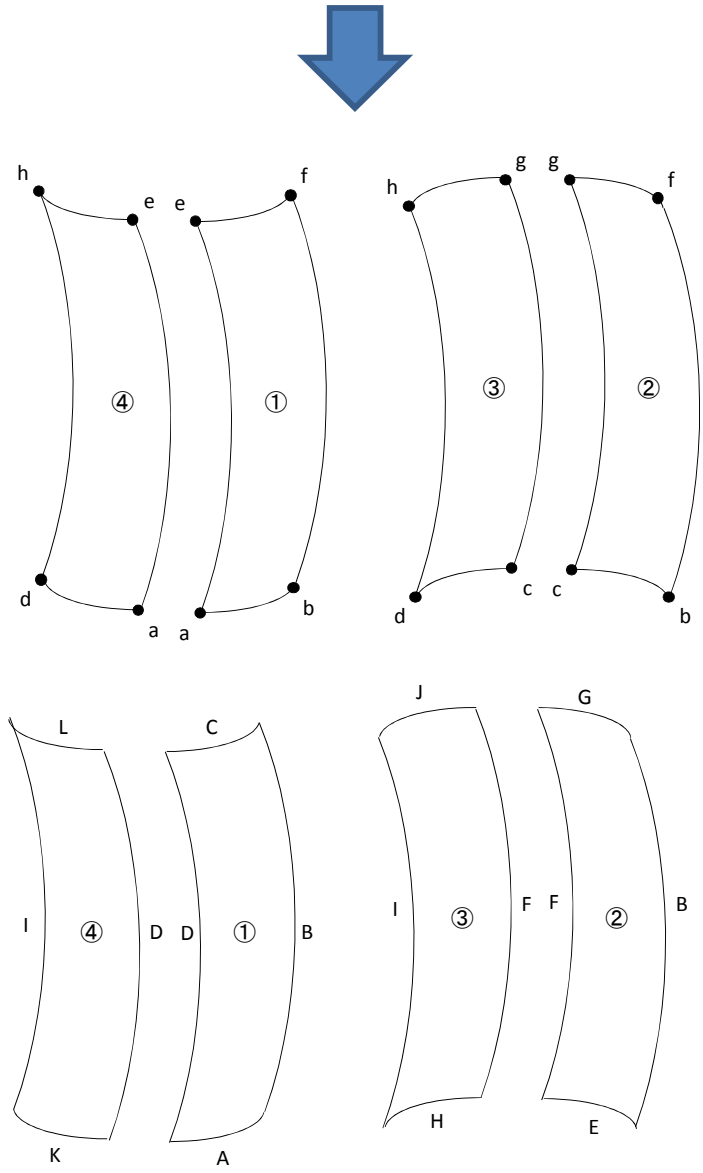
位相情報

本システムでは，掃引面の位相情報として，頂点，稜線と面に関する情報を保持する．掃引面の位相情報を次に示す.

パラメータ	型	説明
m_hFace	Hashtable	面に関する情報

掃引面では，掃引の対象フィーチャが持つ位相情報と掃引を行って作成した立体の上面の位相情報を基に掃引面の位相情報を設定する．掃引面の位相情報の例を次に示す.



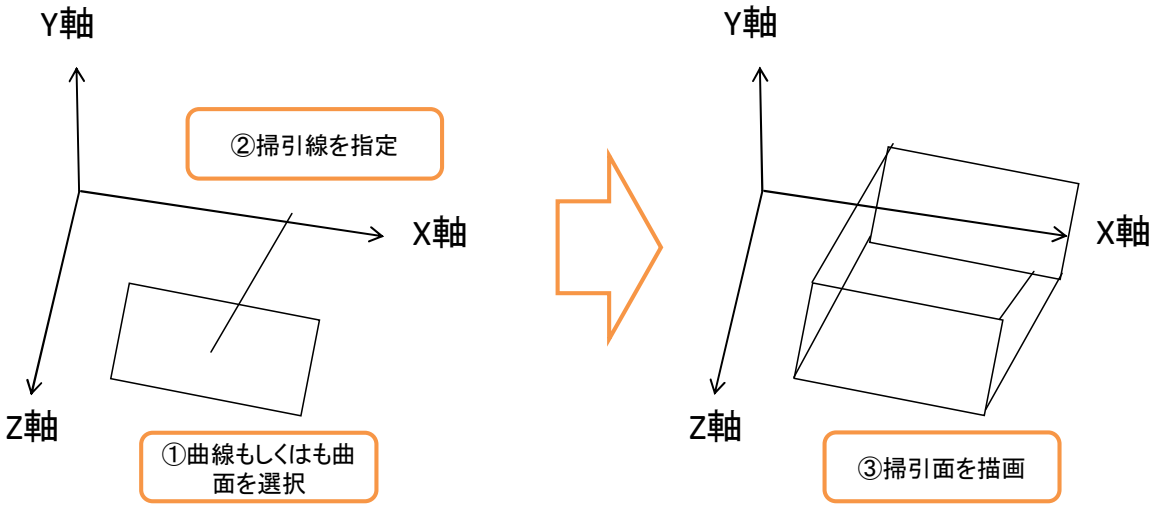


掃引面の面，稜線と頂点の対応を次に示す.

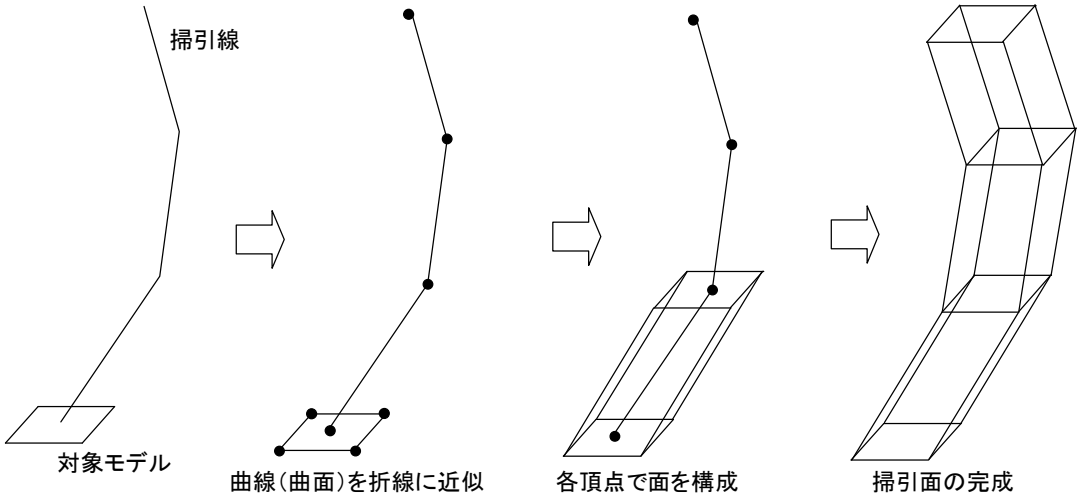
面番号	稜線	頂点
1	A，B，C，D	a，b，f，e
2	E，F，G，B	d，c，g，f
3	H，I，J，F	c，d，h，g
4	K，D，L，I	d，a，e，h

描画方法

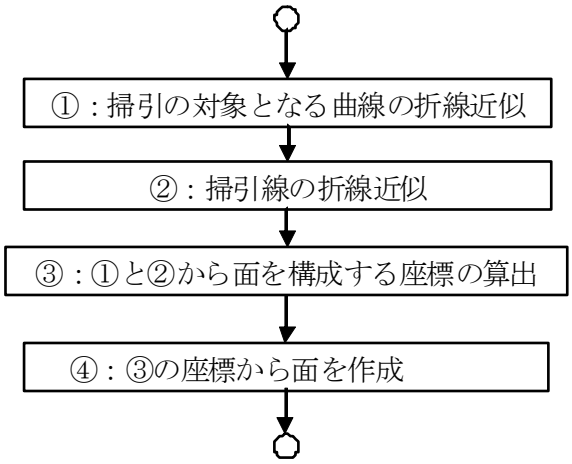
掃引面の描画方法として，曲線（面）要素と掃引線を選択することで面を描画する．掃引面の描画方法を次に示す.



本システムにおける掃引面の描画方法は，選択したモデルと掃引線の頂点座標を算出する．そして，それらの頂点座標を基に面を作成することで描画する．掃引面の描画方法を次に示す．



掃引面の描画の処理フローを次に示す．



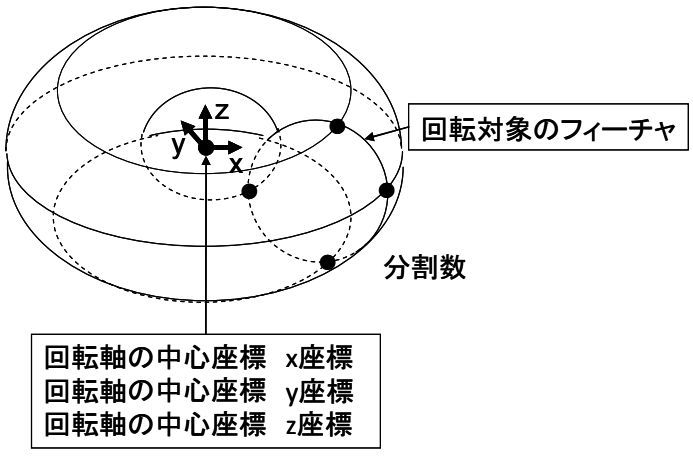
本システムでは，まず，掃引面となる曲線（曲面）と掃引線となる曲線をマウス操作で指定する．次に，掃引線を折線に近似し，折線の各頂点の座標を算出する．そして，折線の頂点と掃引面となる曲線（曲面）から構成される面を作成する．この処理をすべての近似した折線の頂点で行うことで掃引面を描画する．

4.7.4　回転面

本節では，回転面の作成方法について説明する．本節では，まず，回転面の幾何情報について説明する．次に，回転面の位相情報について説明する．最後に，回転面の描画方法について説明する．

幾何情報

回転面の幾何情報を次に示す．

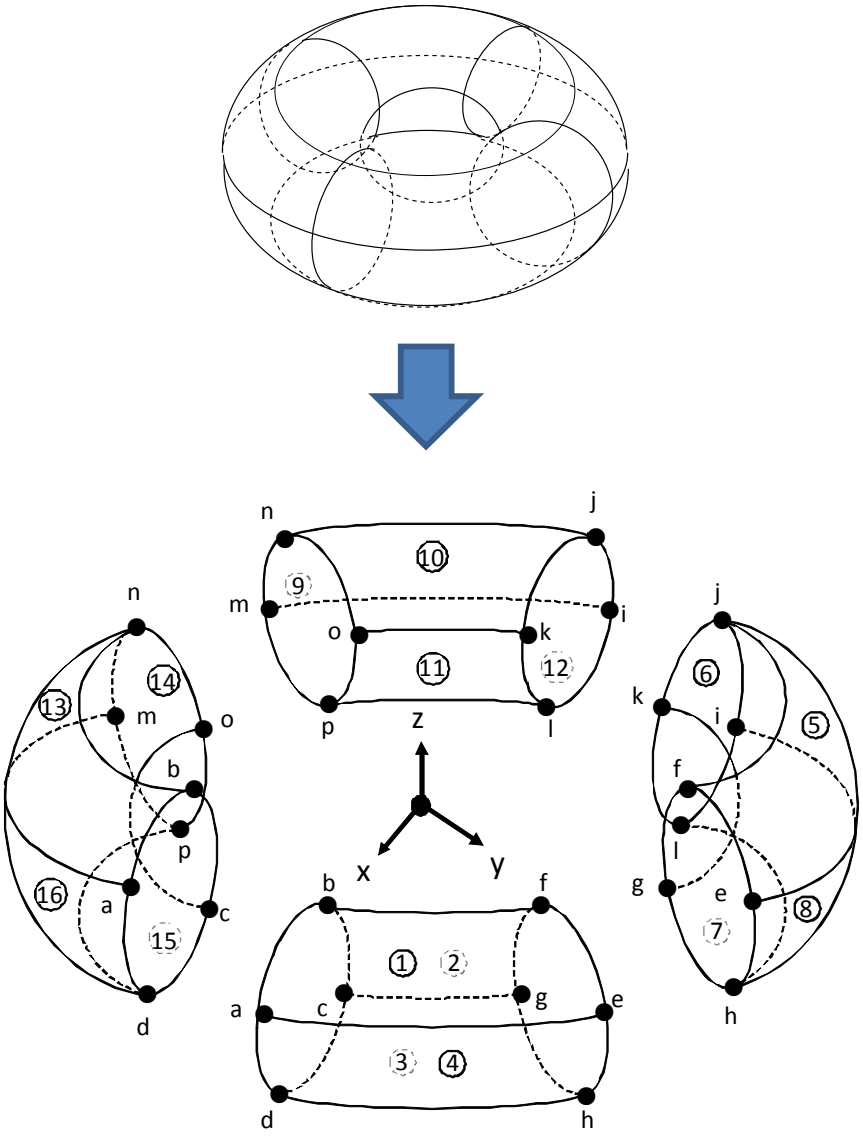


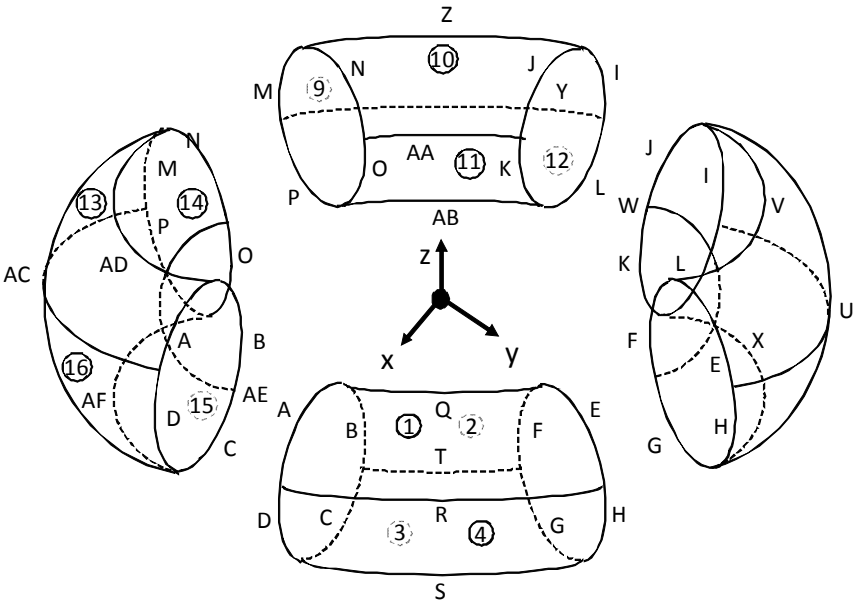
位相情報

本システムでは，回転面の位相情報として，頂点，稜線と面に関する情報を保持する．回転面の位相情報を次に示す．

パラメータ	型	説明
m_hFace	Hashtable	面に関する情報

掃引面では，回転の対象フィーチャが持つ位相情報に加え，90 度ずつ回転を行い作成した立体の上面の位相情報を基に回転面の位相情報を設定する．回転面の位相情報の例を次に示す．



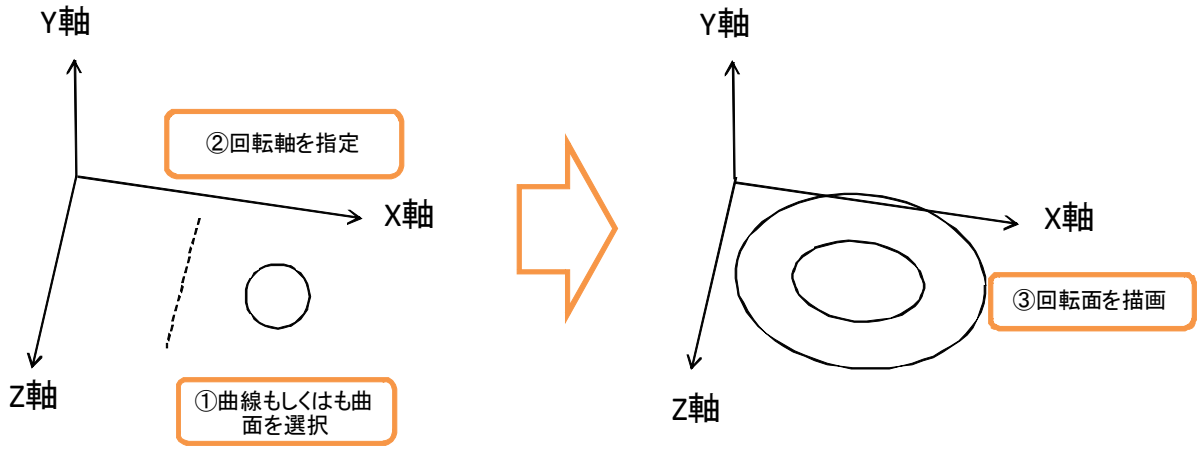


回転面の面，稜線と頂点の対応を次に示す.

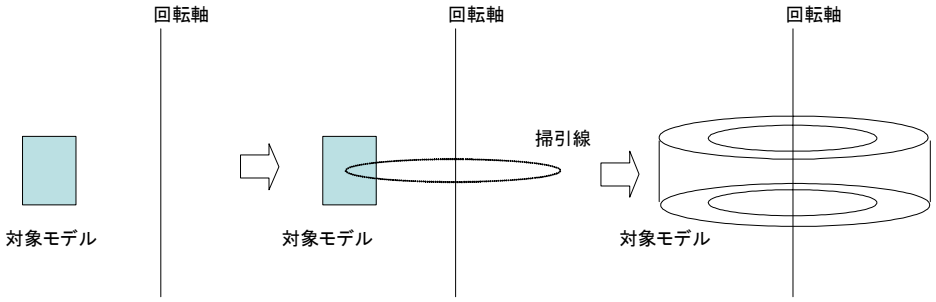
面番号	稜線	頂点
1	A, Q, E, R	a, b, f, e
2	B, T, F, Q	b, c, g, f
3	C, S, G, T	c, d, h, g
4	D, R, H, S	d, a, e, h
5	E, V, I, U	e, f, j, i
6	F, W, J, V	f, g, k, j
7	G, X, K, W	g, h, l, k
8	H, U, L, X	h, e, i, l
9	I, Z, M, Y	i, j, n, m
10	J, AA, N, Z	j, k, o, n
11	K, AB, O, AA	k, l, p, o
12	L, Y, P, AB	l, i, m, p
13	M, AD, A, AC	m, n, b, a
14	N, AE, B, AD	n, o, c, b
15	O, AF, C, AE	o, p, d, c
16	P, AC, D, AF	p, m, a, d

描画方法

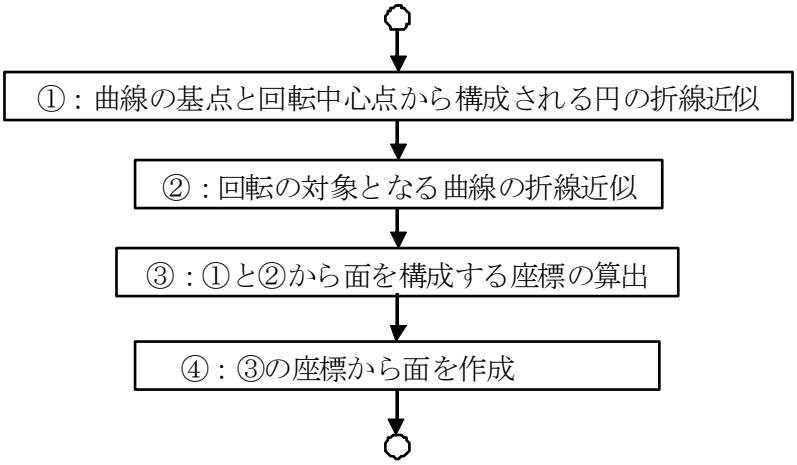
回転面の描画方法として，曲線（面）要素を選択し，回転中心と回転軸を指定することで面を描画する．回転面の描画方法を次に示す．



本システムにおける回転面の描画方法は，選択したモデルから頂点座標を算出する．指定した回転位置と回転軸から回転の対象となる曲線（面）の軌跡を掃引線と考え，掃引面を描画することで実現する．回転面の描画方法を次に示す．



回転面の描画の処理フローを次に示す．



本システムでは，まず，回転の対象となる曲線（曲面）をマウス操作で指定する．次に，マウス操作で回転軸と回転中心をマウス操作で指定する．ただし，回転軸の指定は，選択した曲線（曲面）の配置点を通り，各平面に平行な平面に投影する．そして，回転中心については，選択した曲線（曲面）の配置点から指定した回転軸に下ろした垂線の位置とする．

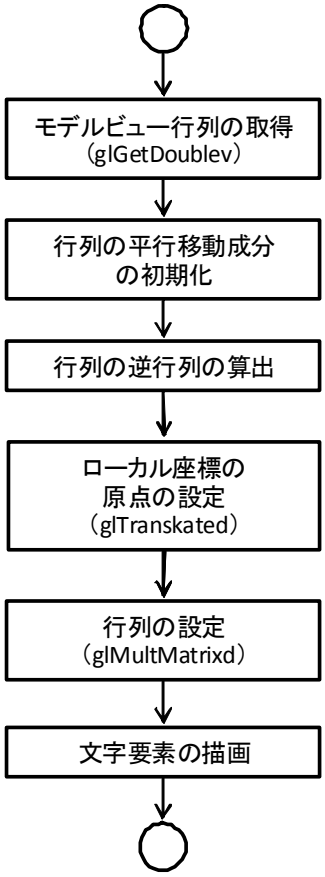
回転面を描画するには，まず，回転面となる曲線（曲面），回転軸，回転中心を指定する．次に，回転軸と回転中心から回転体となる曲線（曲面）の軌道を算出する．そして，算出した軌道を掃引線と考え，掃引面と同様の方法で回転体を描画する．また，回転面を描画する際，ラバーバンド表示を行う．回転面のラバーバンド表示では，回転面の対象となる曲線（曲面）を指定後，回転軸を選択する際，回転軸を線分で描画する．

4.8　注釈の挿入

本節では，注釈の挿入について説明する．本システムは，文字，直線寸法，半径寸法，直径寸法，角度寸法，弧長寸法，引出線とバルーンを対象とし，これらの挿入方法について説明する．本節では，まず，本システムにおける文字の表示について説明する．次に，各注釈について説明する．

4.8.1　文字の表示

本システムでは，各注釈の文字を常にカメラの方向を向くように表示する．文字を常にカメラの方向を向けるには，カメラの位置と傾きを使用することで実現する．文字の向きの算出方法を次に示す．



文字を常にカメラに向けるには，まず，モデルビュー行列を取得し，モデルビュー行列の平行移動成分を初期化（0）する．次に，取得した行列の逆行列を算出する．そして，文字のローカル座標を設定し，算出した逆行列を設定する．最後に，設定した環境下で文字を描画する．以上の処理を行うことで文字の向きを常にカメラの方向へ向ける．上記の処理の実装例を次に示す．

```
// 行列の要素を格納する変数
double dMatrix[16];
double dInverMatrix[16];
// ①モデルビュー行列の取得
glGetDoublev(GL_MODELVIEW_MATRIX, dMatrix);
// ②平行移動成分の初期化
dMatrix[12] = 0.0;
dMatrix[13] = 0.0;
dMatrix[14] = 0.0;
// ③逆行列の算出
Inverse(dInverMatrix, dMatrix, 4);
// 行列スタックの取得
glPushMatrix();
    // ④ローカル座標の原点の設定
    glTranslated(20,0,0);
    // ⑤行列の設定
```

```
    glMultMatrixd(dInverMatrix);
    // ⑥文字要素の描画
    ・・・

// 行列スタックの解放
glPopMatrix();
```

4.8.2 矢印の描画

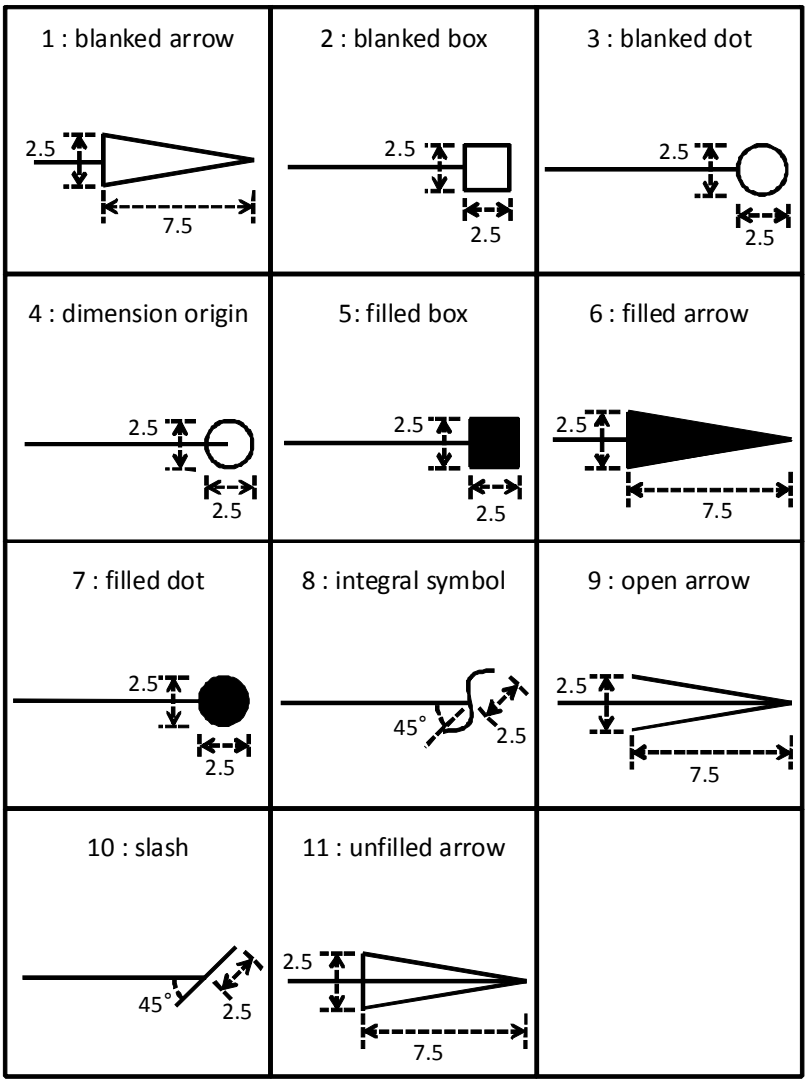
本節では，矢印の描画方法について説明する．本システムでは，全ての注釈で矢印を描画する必要がある．そして，本システムでの矢印は，各注釈のパラメータである矢印コードに応じて矢印の形状が異なる．本節では，まず，矢印コードと矢印の形状の対応について説明する．次に，本システムにおける矢印の描画方法について説明する．

矢印コード

本システムにおける矢印コードと矢印の対応を次に示す.

矢印コード	矢印の種類
1	blanked arrow
2	blanked box
3	blanked dot
4	dimension origin
5	filled box
6	filled arrow
7	filled dot
8	integral symbol
9	open arrow
10	slash
11	unfilled arrow

矢印コードに対応する各矢印の形状を次に示す.



矢印の描画方法

本項では，各矢印の描画方法について説明する．

blanked arrow

blanked arrow は，glVertex3d 関数を使用して 3 本の直線で三角形を描画する．blank arrow の実装例を次に示す．

```
// 描画の開始
glBegin(GL_LINE_LOOP);
    // 描画の実行
    glVertex3d(0,0,0);
    glVertex3d(iDirection*7.5,-1.25,0);
    glVertex3d(iDirection*7.5, 1.25,0);
// 描画の終了
glEnd();
```

glBegin 関数は，頂点データの始まりを宣言する関数であり，glEnd は，頂点データの終了を宣言する関数である．blank arrow は，三角形（閉じた多角形）であるため，glBegin 関数の引数に GL_LINE_LOOP を指定する．ただし，blank arrow は，三角形の内部に補助線を表示しないため，blank arrow の場合，補助線の始点（終点）の座標を「7.5」だけずらして補助線を描画する．

blanked box

blanked box は，glVertex3d 関数を使用して４本の直線で四角形を描画する．blank box の実装例を次に示す．

```
// 描画の開始
glBegin(GL_LINE_LOOP);
    // 描画の実行
    glVertex3d(-1.25,-1.25,0);
    glVertex3d(-1.25, 1.25,0);
    glVertex3d( 1.25, 1.25,0);
    glVertex3d( 1.25,-1.25,0);
// 描画の終了
glEnd();
```

glBegin 関数の引数に blanked arrow と同様で GL_LINE_LOOP を指定する．また，blanked box も blanked arrow と同様に四角形の内部に補助線を表示しないため，補助線の始点（終点）の座標を「1.25」だけずらして補助線を描画する．

blanked dot

blanked dot は，glVertex3d 関数を使用して円を折線に近似して描画することで実現する．blank dot の実装例を次に示す．

```
// 描画の開始
glBegin(GL_LINE_LOOP);
    // 円を描画するための繰り返し
    for(int i=0;i<iStep;i++){
        // X 座標値の算出
        dX = 1.25 *cos(2.0*PAI*(float)i/(float)iStep) + 0;
        // Y 座標値の算出
        dY = 1.25 *sin(2.0*PAI*(float)i/(float)iStep) + 0;
        // Z 座標値の算出
        dZ = 0;
        // 描画の実行
        glVertex3d(dX,dY,dZ);
    }
// 描画の終了
glEnd();
```

glBegin 関数の引数に blanked arrow と同様で GL_LINE_LOOP を指定する．また，blanked dot も円の内部に補助線を表示しないため，補助線の始点（終点）の座標を「1.25」だけずらして描画する．

dimension origin

dimension origin は，blank dot と同様に glVertex3d 関数を使用して円を折線に近似して描画することで実現する．dimension origin の実装例を次に示す．

filled box

filled box は，blanked box と同様に glVertex3d 関数を使用して 4 本の直線で四角形を描画することで実現する．ただし，filled box は，塗りつぶされた四角形であるため，glBegin 関数の引数には，GL_POLYGON を使用する．

filled arrow

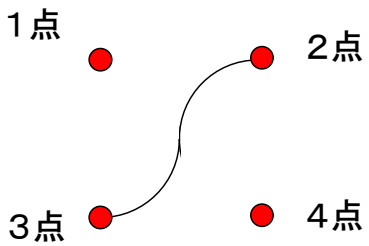
filled arrow は，blanked arrow と同様に glVertex3d 関数を使用して 3 本の直線で三角形を描画することで実現する．ただし，filled arrow は，filled box と同様，塗りつぶされた図形であるため，glBegin 関数の引数には，GL_ POLYGON を使用する．

filled dot

filled dot は，blank dot と同様に glVertex3d 関数を使用して円を折線に近似して描画することで実現する．ただし，filled dot は，filled box と同様，塗りつぶされた図形であるため，glBegin 関数の引数には，GL_ POLYGON を使用する．

integral symbol

integral symbol は，glMap1d 関数を使用してベジェ曲線を描画することで実現する．integral symbol を描画するためのベジェ曲線の制御点を次に示す．



integral symbol の実装例を次に示す．

```
    double dIntegral[13];
    dIntegral[0] = iDirection*(-2.5/sqrt(2.0));
    dIntegral[1] = (2.5/sqrt(2.0));
    dIntegral[2] = 0;
    dIntegral[3] = iDirection*(2.5/sqrt(2.0));
    dIntegral[4] = (2.5/sqrt(2.0));
    dIntegral[5] = 0;
    dIntegral[6] = iDirection*(-2.5/sqrt(2.0));
    dIntegral[7] = (-2.5/sqrt(2.0));
    dIntegral[8] = 0;
    dIntegral[9] = iDirection*(2.5/sqrt(2.0));
    dIntegral[10] =(-2.5/sqrt(2.0));
    dIntegral[12] = 0;
    // エバリュエータの定義
    glMap1d(GL_MAP1_VERTEX_3, 0, 1, 3, 4, dIntegral);
    glEnable(GL_MAP1_VERTEX_3);

    // 描画の開始
    glBegin(GL_LINE_STRIP);
        // 矢印描画のための繰り返し
        for(int t = 0; t <= iStep; ++t){
            // 描画の実行
            glEvalCoord1d(t/(iStep*1.0));
        }
    // 描画の終了
    glEnd();
```

open arrow

open arrow は, glVertex3d を利用して 2 本の直線を描画することで実現する. ただし, open arrow は，開いた図形であるため，glBegin 関数の引数には，GL_LINE_STRIP を指定する. open arrow の実装例を次に示す.

```
    // 描画の開始
    glBegin(GL_LINE_STRIP);
        // 描画の実行
        glVertex3d(iDirection*7.5,-1.25,0);
        glVertex3d(0,0,0);
        glVertex3d(iDirection*7.5, 1.25,0);
    // 描画の終了
    glEnd();
```

slash

slash は，glVertex3d を利用して直線を描画することで実現する．slash の実装例を次に示す.

```
    // 描画の開始
    glBegin(GL_LINE_STRIP);
```

```
        // 描画の実行
        glVertex3d(iDirection*(-2.5/sqrt(2.0)),(2.5/sqrt(2.0)),0);
        glVertex3d(iDirection*(2.5/sqrt(2.0)),(-2.5/sqrt(2.0)),0);
    // 描画の終了
    glEnd();
```

unfilled arrow

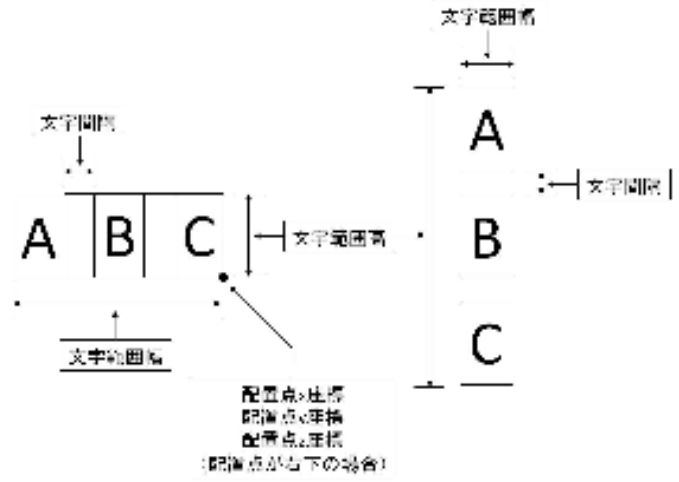
unfilled arrow は，filled arrow と同様で glVertex3d を利用して 3 本の直線で三角形を描画することで実現する．ただし，unfilled arrow は，塗りつぶされた図形でないため，glBegin 関数の引数には，GL_LINE_LOOP を指定する．

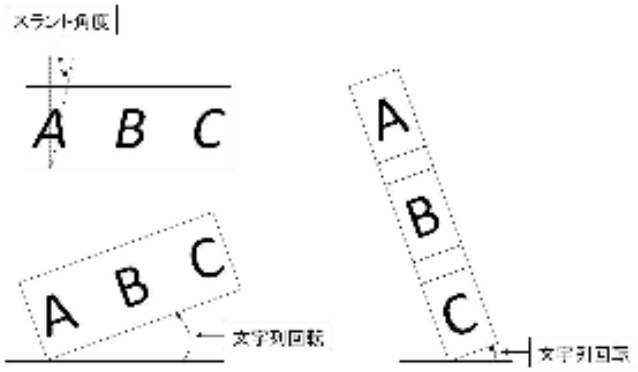
4.8.3 文字

本節では，文字の挿入方法について説明する．

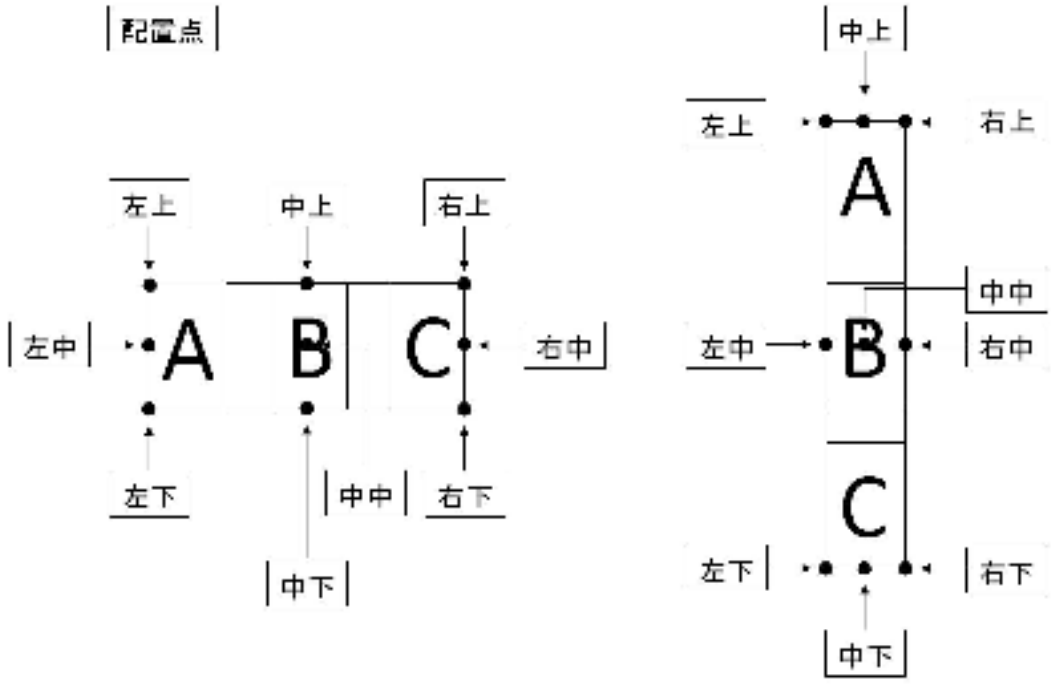
幾何情報

文字の幾何情報を次に示す．


文字間隔
A
B
C
文字範囲高
文字範囲幅
配置点>左揃
配置点>左揃
(開始点が右下の場合)
文字範囲高
A
B
C
文字間隔

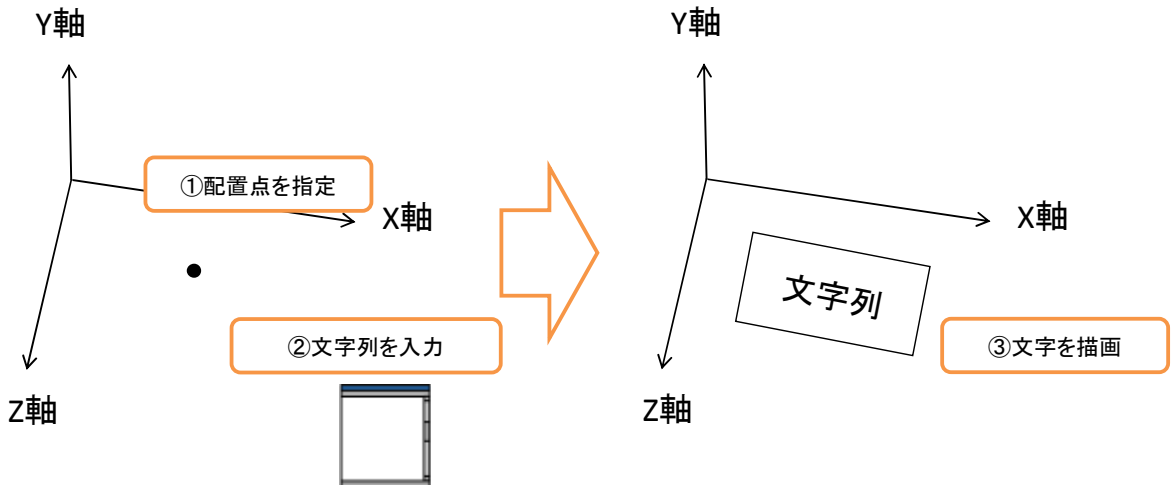

スラント角度
A B C
A B C
文字列回転
A
B
C
文字列回転

SXF 仕様より参照



挿入方法

本システムでは，モデル空間上においてマウス操作で文字を描画する．文字の描画手順を次に示す．



本システムでは，まず，配置点（X ,Y ,Z）をマウス操作で指定する．次に，ダイアログ上で文字列を入力する．最後に，配置点を基点に文字を描画する．ただし，文字の配置点

については，第 5.1 節で説明した 3 次元座標の算出方法で取得した 3 次元座標上とする．また，文字列範囲の高さ，文字間隔，スラント角度，文字方向と文字列の配置位置については，文字列を入力するダイアログ上で設定する．そして，文字列範囲の幅については，文字数と文字列範囲の高さから算出する．

描画方法

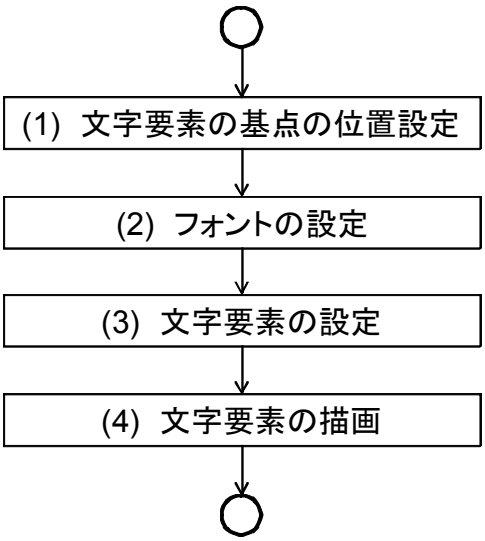
次に，文字の描画方法について説明する．本システムでは WGL を使用して文字を描画する．WGL には，文字を描画する関数として，wglUseFontBitmaps 関数と wglUseFontOutlines 関数が用意されている．これらのうち，wglUseFontBitmaps 関数は日本語フォントに対応していないため，本システムでは wglUseFontOutlines 関数を使用する．wglUseFontOutlines 関数の定義を次に示す．

```
void wglUseFontOutlines(HDC HDC, DWORD FontCode, DWORD List, DWORD baseID, float gap,
                        float push,int FontModel, LPGLYPHMETRICSFLOAT[] gmf)
```

wglUseFontOutlines 関数は 8 個の引数をとり，戻り値はない．wglUseFontOutlines 関数の引数を次に示す．

引数名	型	引数の意味
HDC	HDC	デバイスコンテキスト
FontCode	DWORD	文字フォント
List	DWORD	ディスプレイリストの総数
baseID	DWORD	ディスプレイリストの番号
gap	float	なめらかさを示す値
push	float	奥行きの幅
FontModel	int	フォントモデル
gmf	LPGLYPHMETRICSFLOAT[]	グリフの寸法を格納する配列

本システムにおける文字要素の描画の流れを次に示す．



文字では，まず，文字の基点の位置とフォントを設定する．次に，文字の設定を行う．最後に，文字を描画する．文字の描画の実装例を次に示す．

```
// 1 文字ずつ文字を描画
for(int i = 0;i < barray->Length;i++){
    // 文字コードを取得
    unsigned char txt = static_cast<unsigned char>(barray[i]);
    long code = static_cast<long>(txt);
    // 文字のバイト数の判定
    if (IsDBCSLeadByte(barray[i]) ){
        // 2 バイト文字の場合
        // 文字コードの変換
        txt = static_cast<unsigned char>(barray[i + 1]);
        code = code * 256 + static_cast<long>(txt);
        // 変数 i の加算
        i++;
        // フォントサイズの設定
        iFontSize = 1;
    }else{
        // 1 バイト文字の場合
        // フォントサイズの設定
        iFontSize = 2;
    }

    // 行列スタックの取得
    glPushMatrix();
    // 文字の倍率の設定
    glScalef(dFontWidth*(10.0/8.0)*iFontSize,dFontHeight*(10.0/8.0),0);

    // 文字の描画
    wglUseFontOutlines(hDC,code,1,i,0.0f,0.1f,WGL_FONT_POLYGONS, &agmf);
    // i 番目のオブジェクトの呼出し
    glCallList(i);
    // 行列スタックの解放
    glPopMatrix();
```

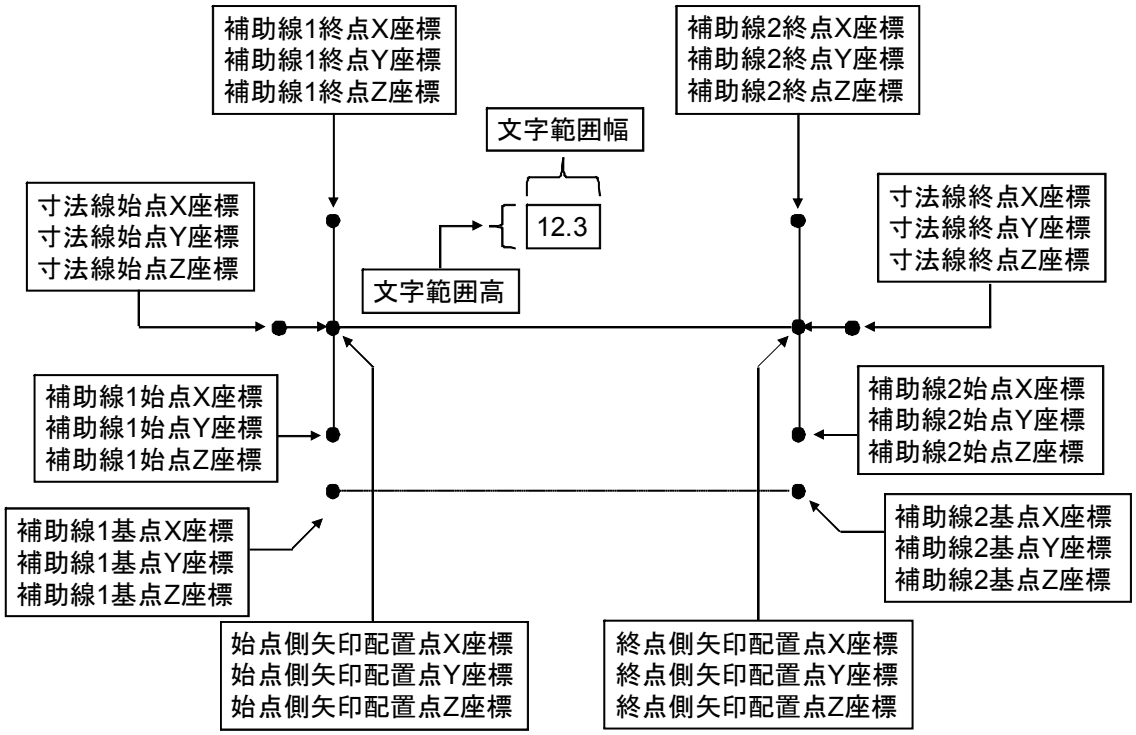
```
    // 文字の描画位置の設定
    if(m_iDirect == 1){
        // 横書きの場合
        glTranslatef((dFontWidth+m_dSpc),0,0);
    }else{
        // 縦書きの場合
        glTranslatef(0,-(dFontHeight+m_dSpc),0);
    }
}
```

4.8.4 直線寸法

本節では，直線寸法の挿入方法について説明する．

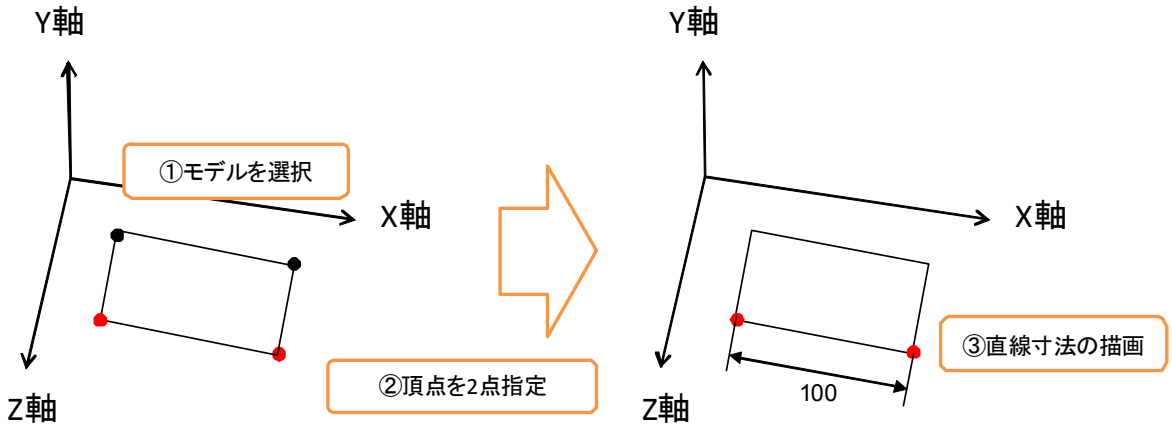
幾何情報

直線寸法の幾何情報を次に示す．



挿入方法

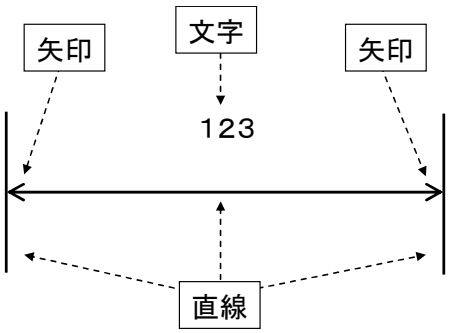
本システムでは，モデル空間上においてマウス操作で直線寸法を描画する．直線寸法の描画手順を次に示す．



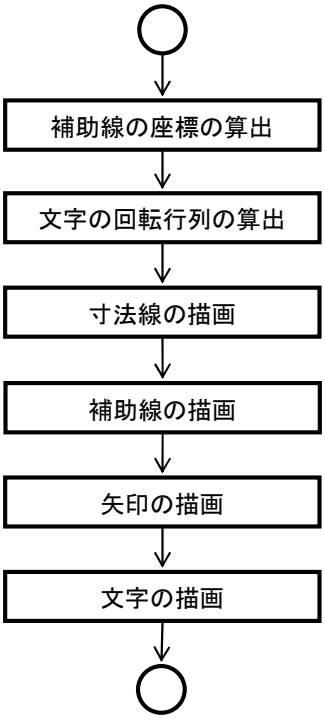
本システムでは，まず，直線寸法を挿入するモデルをマウス操作で指定する．次に，選択したモデルに頂点を表す球を表示し，直線寸法を挿入するために関連付ける頂点を 2 点選択する．最後に，選択した頂点の座標を基に直線寸法を描画する．ただし，寸法値は，2 点間の距離を表す．また，補助線の有無，位置および矢印の有無，形状については，オプションで設定・変更できるようにする．また，直線寸法を挿入する際，ラバーバンド表示を行う．直線寸法のラバーバンド表示では，関連付ける頂点の 1 点目を指定後，2 点を指定するまで，マウスの移動を検出し，マウスの 3 次元座標を算出する．そして，1 点目の頂点と算出した 3 次元座標を結び線分を描画する．

描画方法

直線寸法の描画方法について説明する．直線寸法は，直線，矢印，文字を描画することで実現する．直線寸法の描画方法を次に示す．

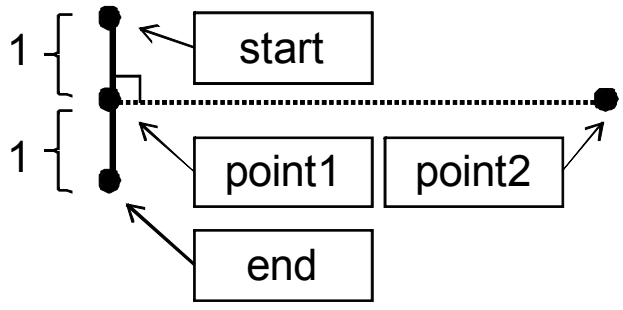


本システムにおける直線寸法の描画の流れを次に示す．



本システムでは，まず，寸法線の情報を基に補助線の描画位置の座標および文字の回転行列を算出する．次に，これらの情報を基に寸法線と補助線を描画する．そして，矢印を矢印コードに基づいて描画する．最後に，寸法値である文字を描画する．

補助線の座標の算出では，寸法線に直交する線分を考え，直交する線分の始点と終点を算出する．しかし，3 次元空間上では，直交する線分が一意に決定できないため，本システムでは，算出する線分(start，end)の z 座標を基準となる線分の始点(point1)の z 座標に固定する．また，算出する線分の距離を固定（オプションで設定）とする．本システムにおける寸法線と補助線の位置関係を次に示す.



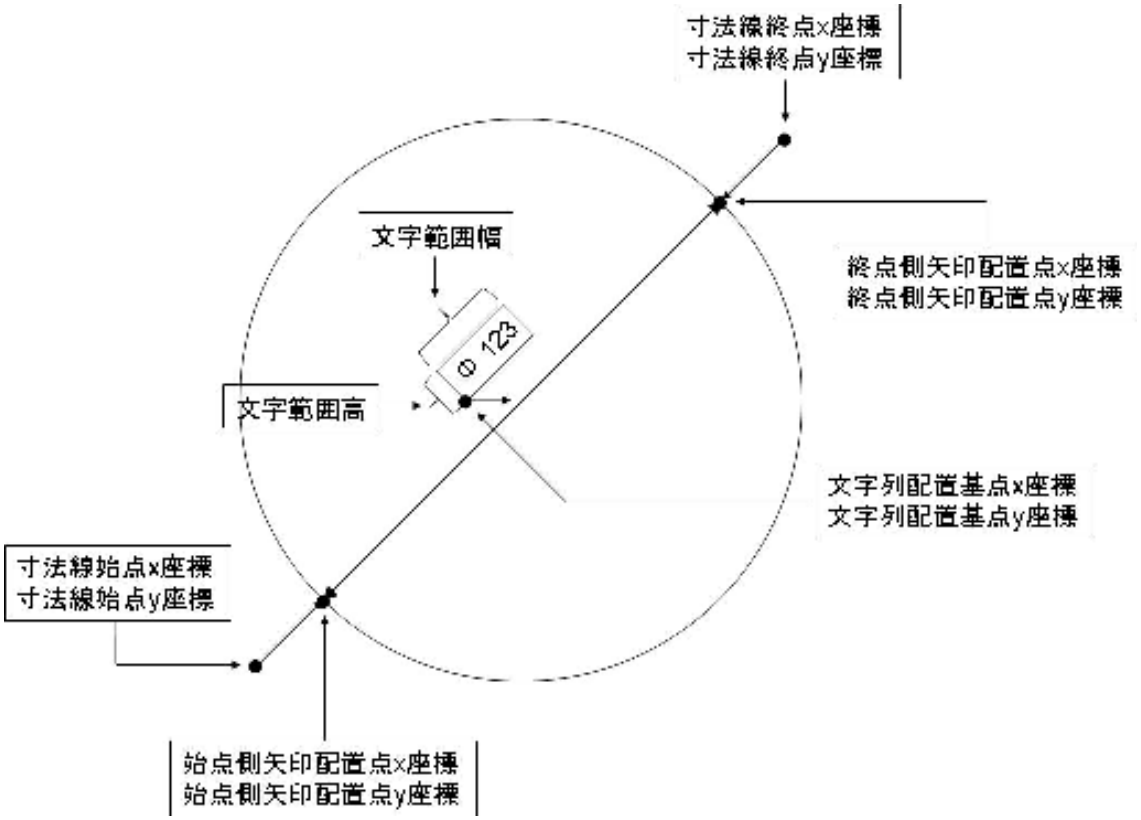
寸法線と補助線の描画では, 算出した座標に基に glVertex3d を使用して線分を描画することで実現する.

4.8.5　直径寸法

本節では，直径寸法の挿入方法について説明する．

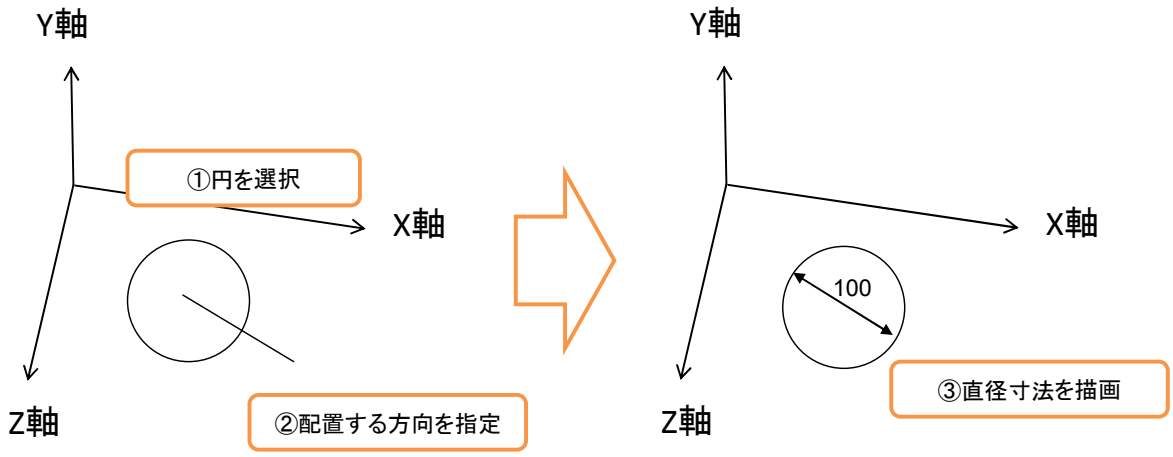
幾何情報

直径寸法の幾何情報を次に示す．



挿入方法

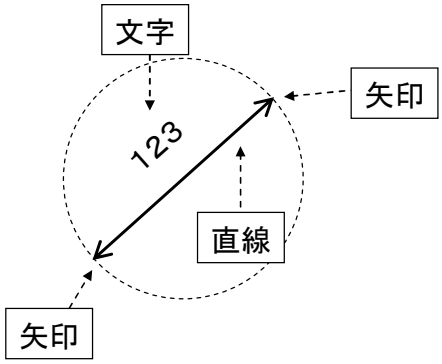
本システムでは，モデル空間上においてマウス操作で直径寸法を挿入する．直径寸法の挿入手順を次に示す．



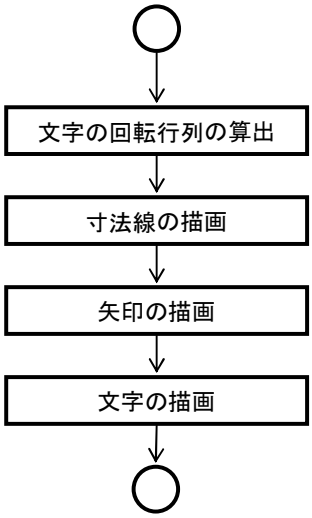
本システムでは，まず，直径寸法を挿入するモデル（円）をマウス操作で指定する．次に，直径寸法の方向をマウス操作で指定する．最後に，直径寸法を描画する．ただし，寸法値は，円の直径を表す．そして，矢印の有無，形状については，オプションで設定・変更できるようにする．また，直径寸法を挿入する際，ラバーバンド表示を行う．直径寸法のラバーバンド表示では，挿入するモデルを指定後，直径寸法の方向を指定するまで，マウスの移動を検出し，マウスの 3 次元座標を算出する．そして，挿入するモデルの中心を通り，モデルの直径の長さの線分を描画する．

描画方法

直径寸法の描画方法について説明する．直径寸法は，直線，矢印，文字を描画することで実現する．直径寸法の描画方法を次に示す．



本システムにおける直径寸法の描画の流れを次に示す．



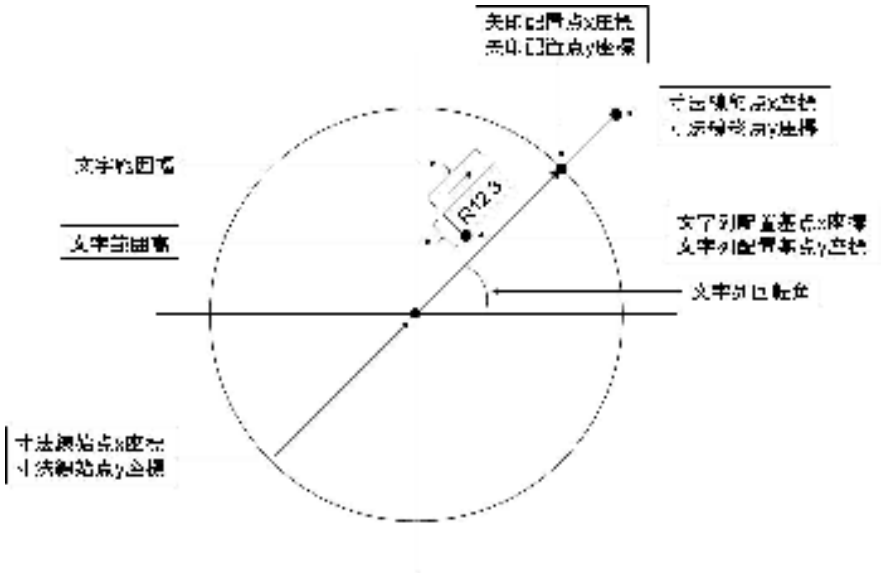
本システムでは，まず，文字の回転行列を算出する．次に，これらの情報を基に寸法線を描画する．そして，矢印コードに基づいて矢印を描画する．最後に，寸法値である文字を描画する．

4.8.6 半径寸法

本節では，半径寸法の挿入方法について説明する．

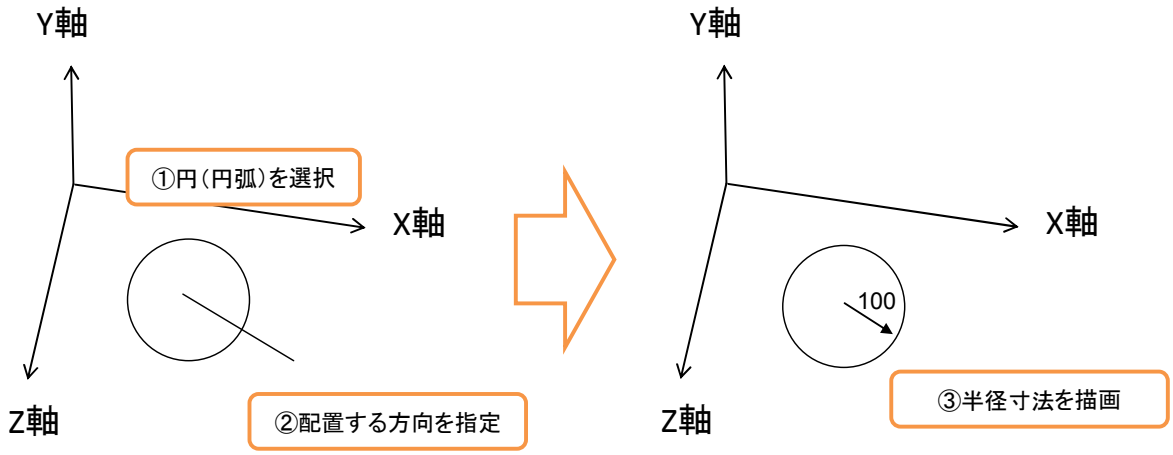
幾何情報

半径寸法の幾何情報を次に示す．



挿入方法

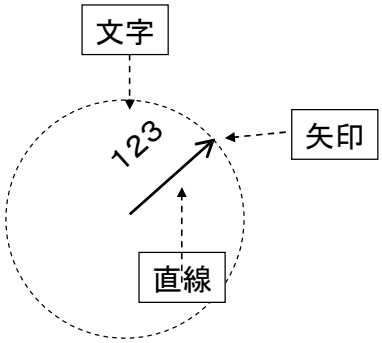
本システムでは，モデル空間上においてマウス操作で半径寸法を挿入する．半径寸法の挿入手順を次に示す．



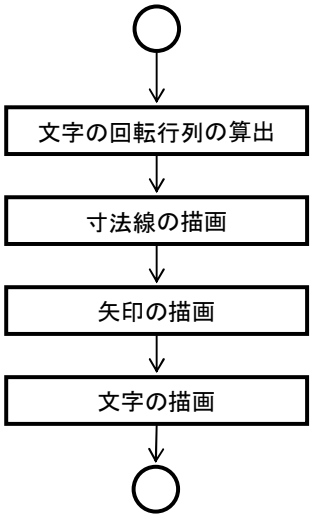
本システムでは，まず，半径寸法を挿入するモデル（円もしくは円弧）をマウス操作で指定する．次に，半径寸法の方向をマウス操作で指定する．最後に，半径寸法を描画する．ただし，寸法値は，円の半径を表す．そして，矢印の有無，形状については，オプションで設定・変更できるようにする．また，半径寸法を挿入する際，ラバーバンド表示を行う．半径寸法のラバーバンド表示では，挿入するモデルを指定後，半径寸法の方向をマウス操作で指定するまで，マウスの移動を検出し，マウスの 3 次元座標を算出する．そして，挿入するモデルの中心点と算出した 3 次元座標を結び線分を描画する．

描画方法

半径寸法の描画方法について説明する．半径寸法は，直線，矢印，文字を描画することで実現する．半径寸法の描画方法を次に示す．



本システムにおける半径寸法の描画の流れを次に示す.



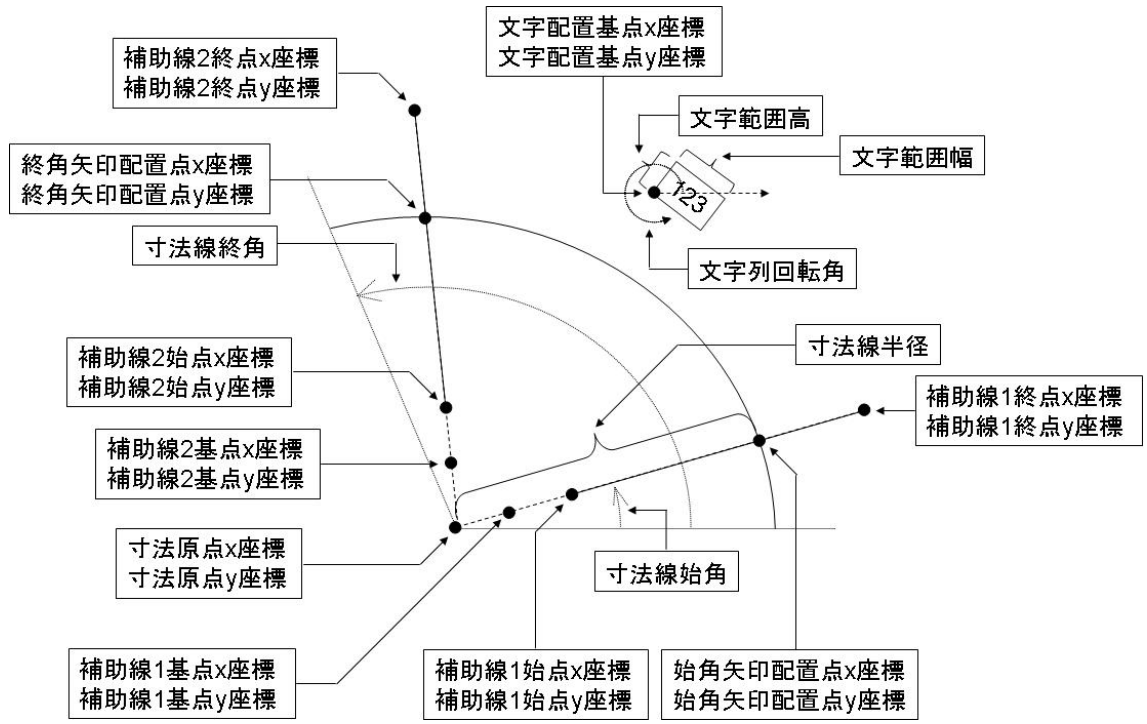
本システムでは，まず，文字の回転行列を算出する．次に，これらの情報を基に寸法線を描画する．そして，矢印コードに基づいて矢印を描画する．最後に，寸法値である文字を描画する.

4.8.7　角度寸法（弧長）

本節では，角度（弧長）寸法の挿入方法について説明する.

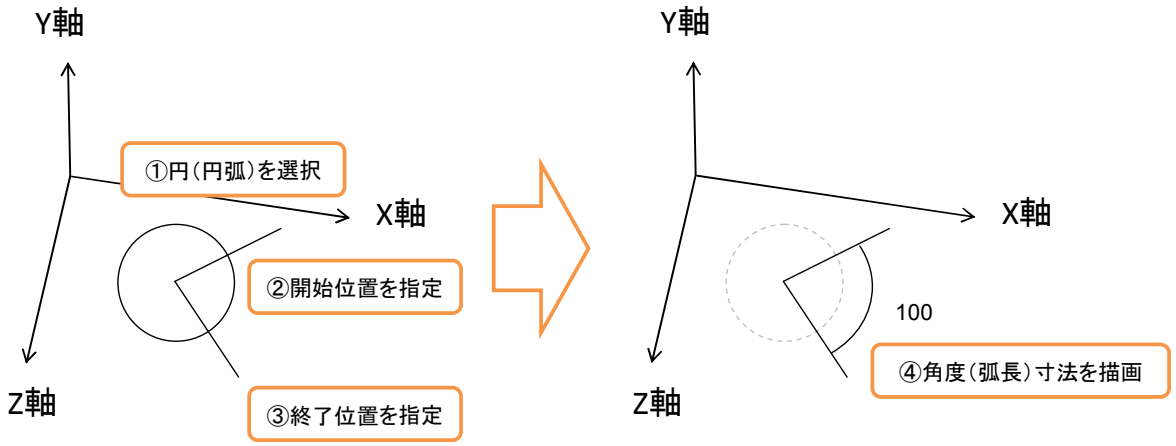
幾何情報

角度（弧長）寸法の幾何情報を次に示す.



挿入方法

本システムでは，モデル空間上においてマウス操作で角度（弧長）寸法を挿入する．角度（弧長）寸法の挿入手順を次に示す．

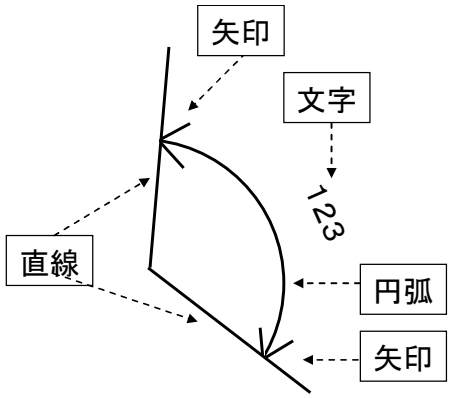


本システムでは，まず，角度（弧長）寸法を挿入するモデル（円もしくは円弧）をマウス操作で指定する．次に，角度（弧長）寸法の開始位置（始角）と終了位置（終角）をマウス操作で指定する．最後に，角度（弧長）寸法を描画する．ただし，寸法値は，角度寸法の場合，開始位置と終了位置間の角度を表し，弧長寸法の場合，開始位置と終了位置間の距離を表す．そして，補助線の有無および矢印の有無，形状については，オプションで

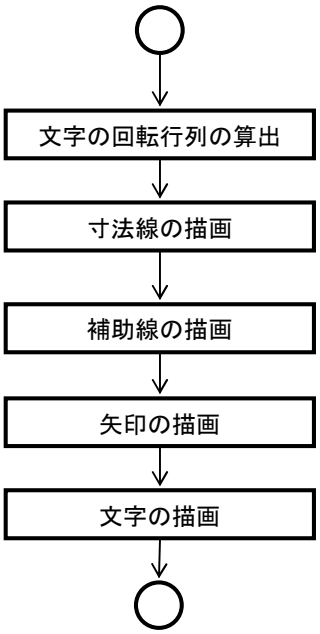
設定・変更できるようにする．また，角度（弧長）寸法を挿入する際，角度（弧長）寸法のラバーバンド表示を行う．角度（弧長）寸法のラバーバンド表示では，まず，挿入するモデルを指定後，開始位置を指定するまで，マウスの移動を検出し，マウスの 3 次元座標を算出する．そして，挿入するモデルの中心点とマウスの 3 次元座標を結ぶ線分を描画する．次に，開始位置を指定後，終了位置を指定するまで，同様に，マウスの移動を検出し，マウスの 3 次元座標を算出する．そして，挿入するモデルの中心点と開始位置を結び線分および挿入するモデルの中心位置とマウスの 3 次元座標を結ぶ線分を描画し，開始位置とマウスの 3 次元座標を結び線分を描画する．

描画方法

角度（弧長）寸法の描画方法について説明する．角度（弧長）寸法は，円弧，直線，矢印，文字を描画することで実現する．角度（弧長）寸法の描画方法を次に示す．



本システムにおける角度（弧長）寸法の描画の流れを次に示す．



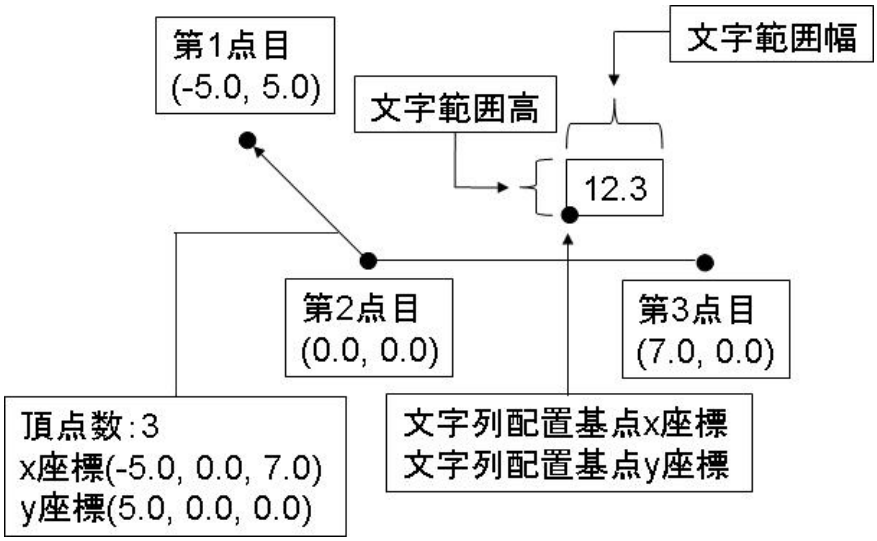
本システムでは，まず，文字の回転行列を算出する．次に，これらの情報を基に寸法線と補助線を描画する．そして，矢印コードに基づいて矢印を描画する．最後に，寸法値である文字を描画する．

4.8.8 引出線

本節では，引出線の挿入方法について説明する．

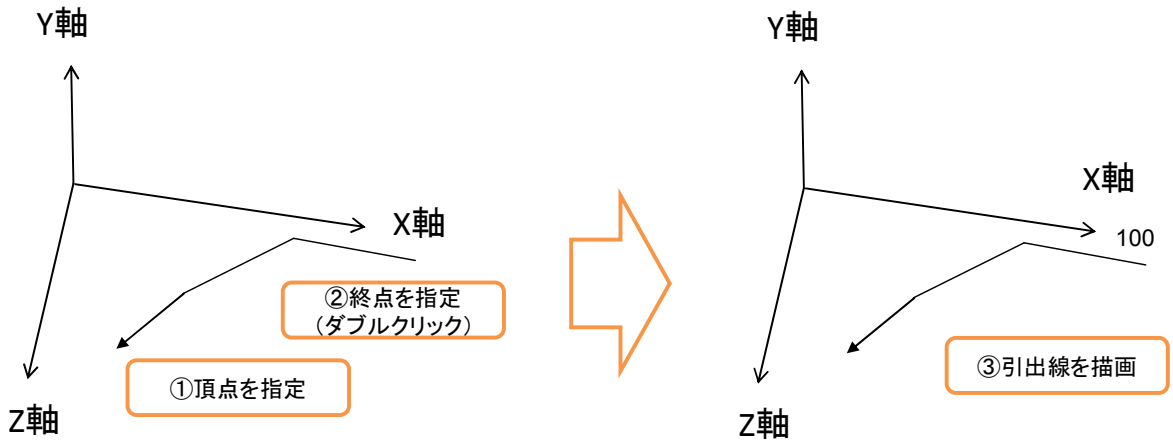
幾何情報

引出線の幾何情報を次に示す．



挿入方法

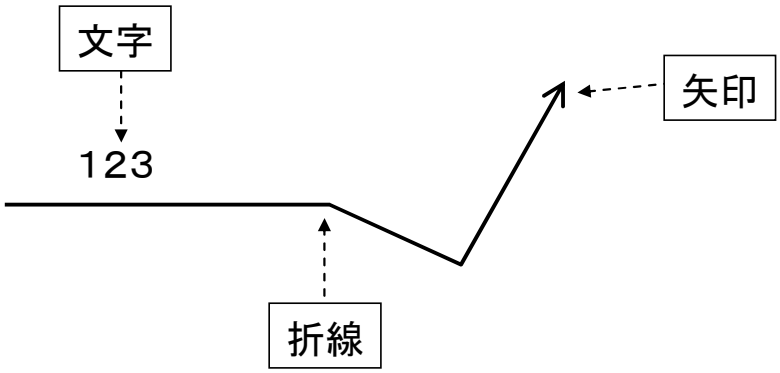
本システムでは，モデル空間上においてマウス操作で引出線を挿入する．引出線の挿入手順を次に示す．



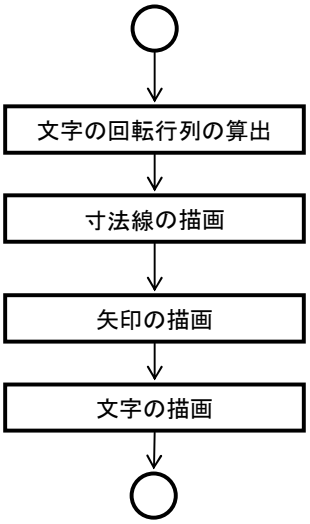
本システムでは，まず，引出線の始点となる頂点を指定する．次に，引出線の終点を指定する（ダブルクリックのイベントを使用する）．そして，ダイアログ上で文字列を入力する．最後に，頂点座標を基に引出線を描画する．ただし，引出線の始点は，3 次元空間上の任意の位置とするが，これ以外の座標については，折線と同様に，１つ前の頂点の座標を通り，各座標軸から平面と平行な平面に投影する．そして，矢印の有無，形状については，オプションで設定・変更できるようにする．また，引出線を挿入する際，ラバーバンド表示を行う．引出線のラバーバンド表示では，折線と同様のラバーバンド表示を行う．

描画方法

引出線の描画方法について説明する．引出線は，折線，矢印，文字を描画することで実現する．引出線の描画方法を次に示す．



本システムにおける引出線の描画の流れを次に示す．



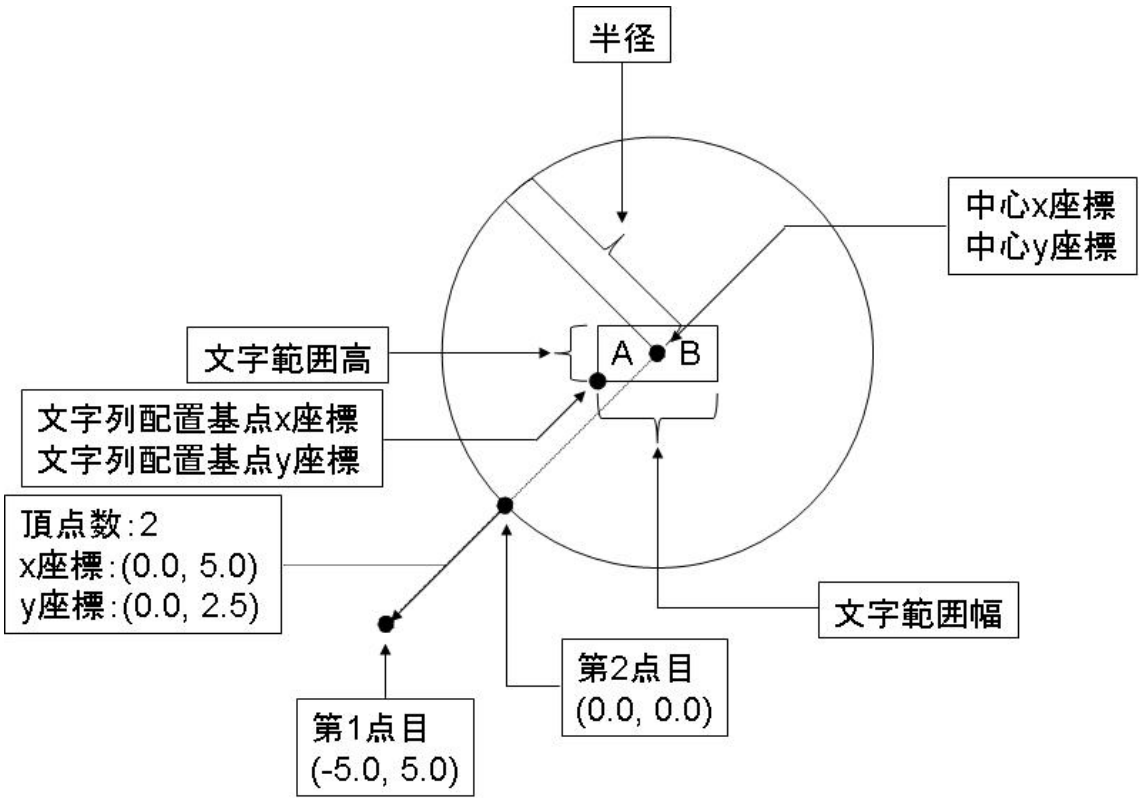
本システムでは，まず，文字の回転行列を算出する．次に，これらの情報を基に寸法線を描画する．そして，矢印コードに基づいて矢印を描画する．最後に，寸法値である文字を描画する．

4.8.9　バルーン

本節では，バルーンの挿入方法について説明する．

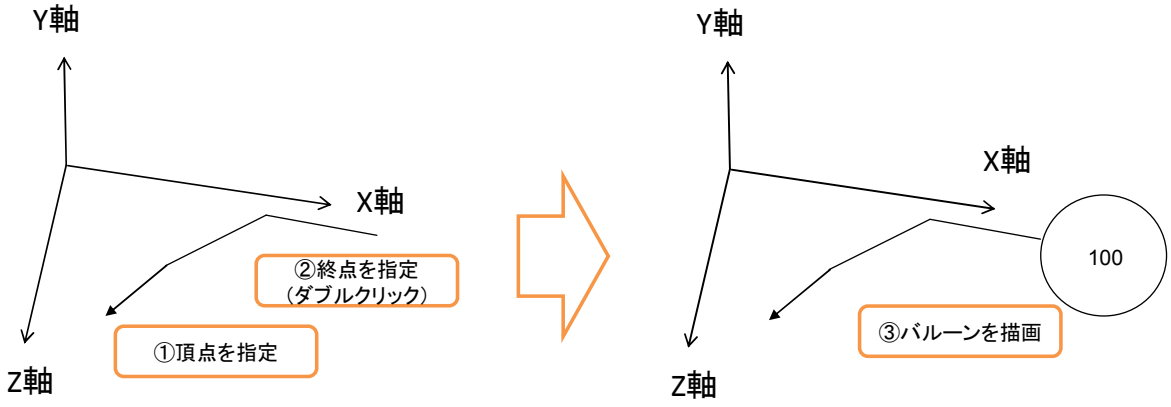
幾何情報

バルーンの幾何情報を次に示す．



挿入方法

本システムでは，モデル空間上においてマウス操作でバルーンを挿入する．バルーンの挿入手順を次に示す．

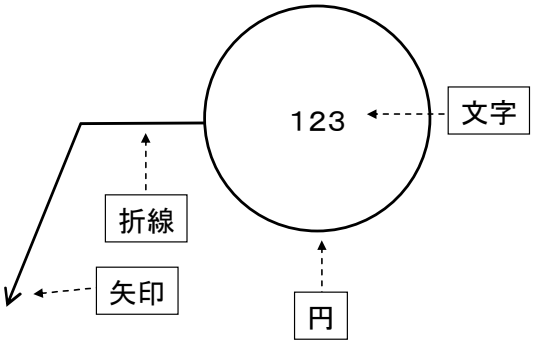


本システムでは，まず，バルーンの始点となる頂点を指定する．次に，バルーンの終点を指定する（ダブルクリックのイベントを使用する）．そして，ダイアログ上で文字列を入力する．最後に，頂点座標を基にバルーンを描画する．ただし，バルーンの始点は，3 次元空間上の任意の位置とするが，これ以外の座標については，折線と同様に，１つ前の頂点

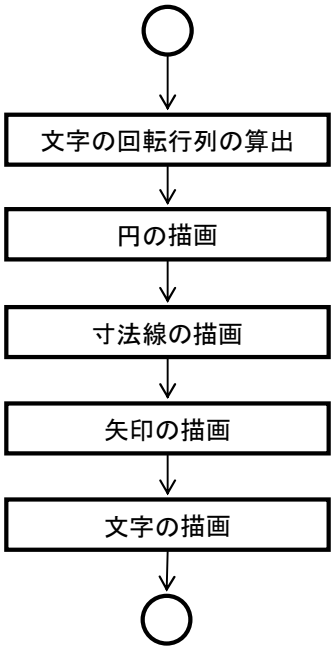
の座標を通り，各座標軸から平面と平行な平面に投影する．そして，矢印の有無，形状については，オプションで設定・変更できるようにする．また，バルーンを挿入する際，ラバーバンド表示を行う．バルーンのラバーバンド表示では，折線と同様のラバーバンド表示を行う．

描画方法

バルーンの描画方法について説明する．バルーンは，折線，円，矢印，文字を描画することで実現する．バルーンの描画方法を次に示す．



本システムにおけるバルーンの描画の流れを次に示す．



本システムでは，まず，文字の回転行列を算出する．次に，これらの情報を基にバルー

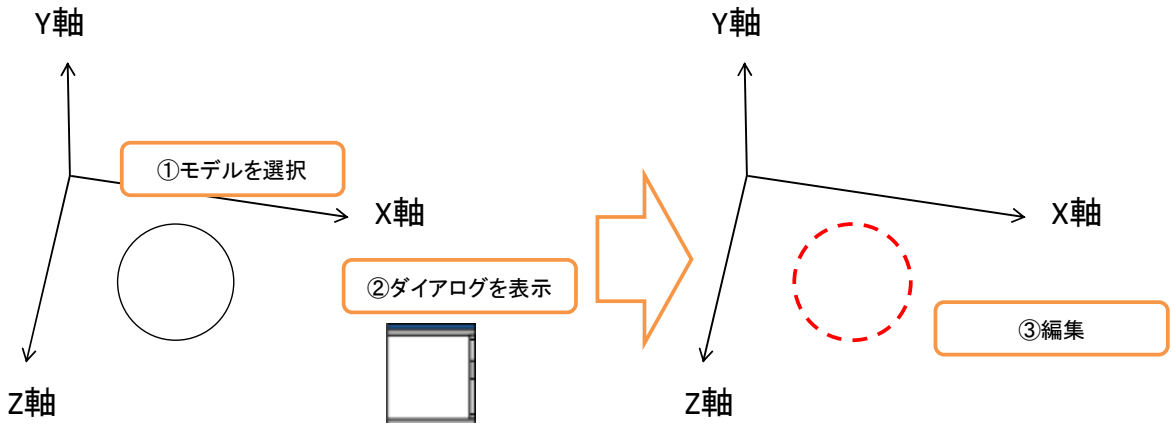
ンの円と寸法線を描画する．そして，矢印コードに基づいて矢印を描画する．最後に，寸法値である文字を描画する．

4.9　編集

本章では，モデルの編集について説明する．本システムでは，テーブル要素の編集，元に戻す，やり直す，削除，移動（複写）と各モデルの編集について実装する．

4.9.1　テーブル要素の編集

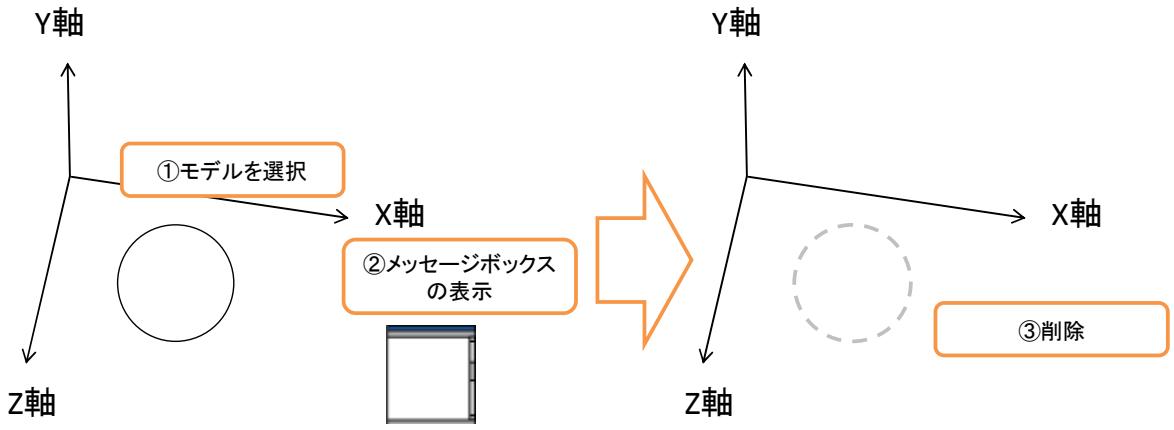
本節では，テーブル要素の編集について説明する．テーブル要素の編集では，モデルの色，線種，線幅を変更する．本システムにおけるテーブル要素の編集の手順を次に示す．



本システムでは，まず，編集するモデルを選択する．モデルを選択すると，テーブル要素を指定するためのダイアログが表示される．次に，ダイアログ上でテーブル要素を指定する．最後に，指定したテーブル要素が反映されたモデルを描画する．

4.9.2　削除

本節では，削除について説明する．本システムでは，指定したモデルを削除する．本システムにおける削除の手順を次に示す．



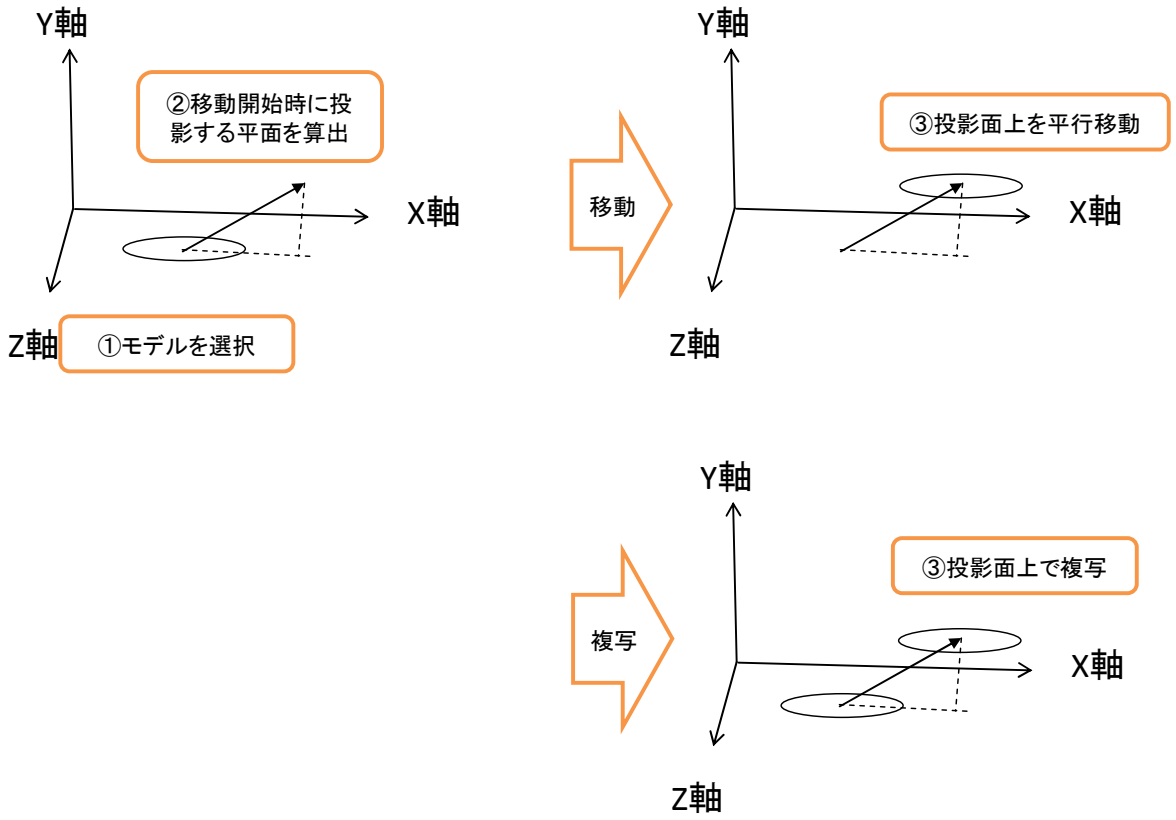
本システムでは，まず，削除するモデルをマウス操作で選択する．モデルを選択すると，「選択したモデルを削除しますか」というメッセージボックスを表示する．最後に，「はい」を選択した場合，モデルを削除する．

4.9.3　移動（複写）

本節では，移動と複写について説明する．移動（複写）では，モデルの平行移動，回転，尺度変更について実装する．

平行移動

本項では，平行移動について説明する．平行移動では，選択したモデルの位置を変更する．本システムにおける平行移動の手順を次に示す．



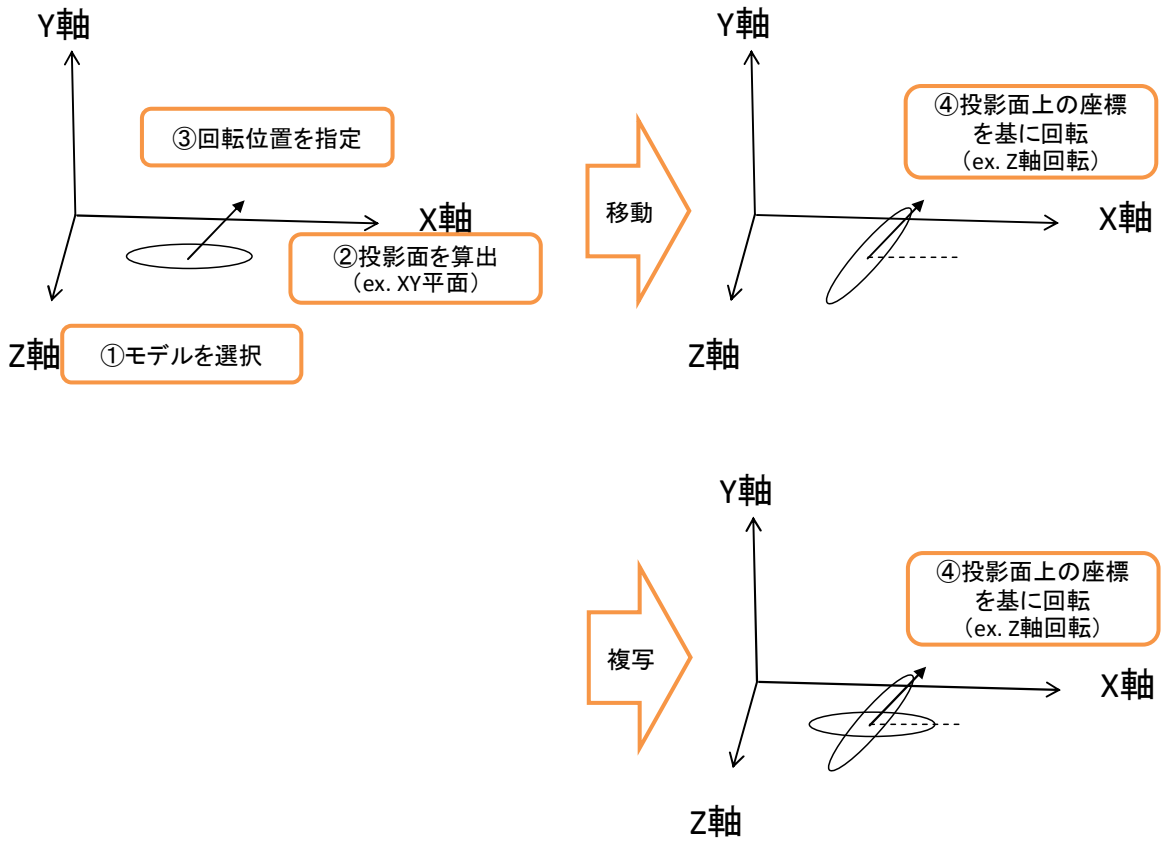
本システムにおけるモデルの移動は，次の手順で行う．

1. モデルをマウス操作で選択する．※マウスのダウンイベント
2. オブジェクト座標を算出（平面投影なし）※マウスのアップイベント
3. マウスの移動でモデルを移動（オブジェクト座標を算出（平面投影なし））※マウスのムーブイベント
4. 2 と 3 で算出したオブジェクト座標の XYZ の差を算出
5. ローカル座標の原点に XYZ の差を加算し，モデルを描画（ラバーバンド処理）
6. マウス操作で移動先を決定（オブジェクト座標を算出（平面投影なし））※マウスのダウンイベント
7. 移動の場合は，モデルを移動させ，複写の場合は，選択したモデルを残した状態で移動先のモデルを描画

※移動の場合は，移動中のモデルのみ描画し，複写の場合は，選択したモデルと移動中のモデルを描画する．

回転

本項では，回転について説明する．回転では，選択したモデルに対して，基準点を中心にモデルを回転させる．本システムにおける回転の手順を次に示す．



本システムにおけるモデルの回転は，次の手順で行う．

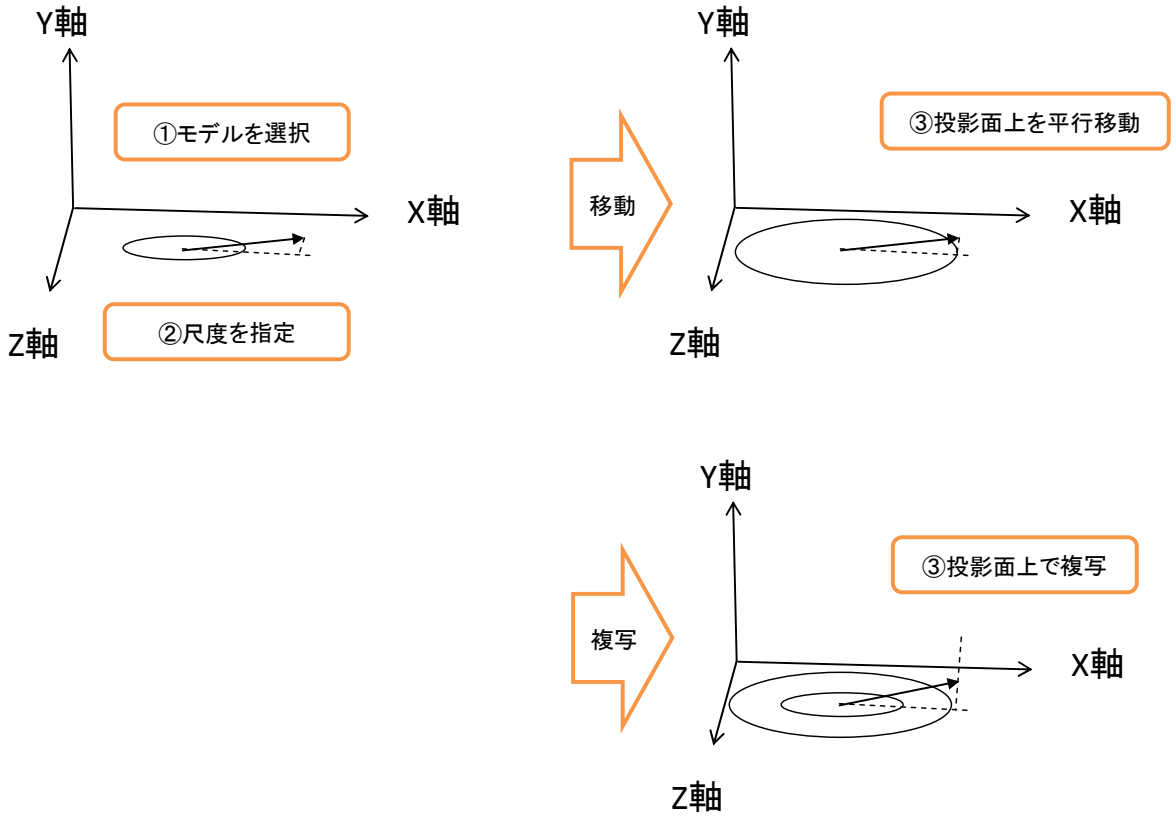
1. モデルをマウス操作で選択する．
2. オブジェクト座標を算出（投影面については，モデルの配置点（中心点）を通り，各座標軸からなる平面に平行になる平面とする．）※マウスのダウンイベント
3. マウスの移動でモデルを回転（回転の角度を算出するため，3 次元座標を算出する際の投影面を決定する．そして，配置点（中心点）と移動後の座標からモデルの回転角度を算出する．）※マウスのムーブイベント
   ※XY 平面：X 軸，XZ 平面：X 軸，YZ 平面：Y 軸
4. マウス操作で移動先を決定（オブジェクト座標を算出）※マウスのダウンイベント
5. 移動の場合は，モデルを回転させ，複写の場合は，選択したモデルを残した状態で回転後のモデルを描画

※移動の場合は，回転中のモデルのみ描画し，複写の場合は，選択したモデルと回転中の

モデルを描画する．

尺度変更

本項では，尺度変更について説明する．尺度変更では，選択したモデルに対して，基準点を中心にモデルの大きさを変更する．本システムにおける尺度変更の手順を次に示す．



本システムにおけるモデルの尺度変更は，次の手順で行う．

1. モデルをマウス操作で選択する．※マウスのダウンイベント
2. オブジェクト座標を算出（投影面については，モデルの配置点（中心点）を通り，各座標軸からなる平面に平行になる平面とする．）※マウスのムーブイベント
3. 配置点（中心点）と移動後の座標からモデルの尺度を算出
4. 各モデルの基準となる長さとの比率を算出（各モデルの基準については次で説明する．）
5. 比率に基にモデルの尺度を変更
6. マウス操作で移動先を決定（オブジェクト座標を算出）※マウスのダウンイベント

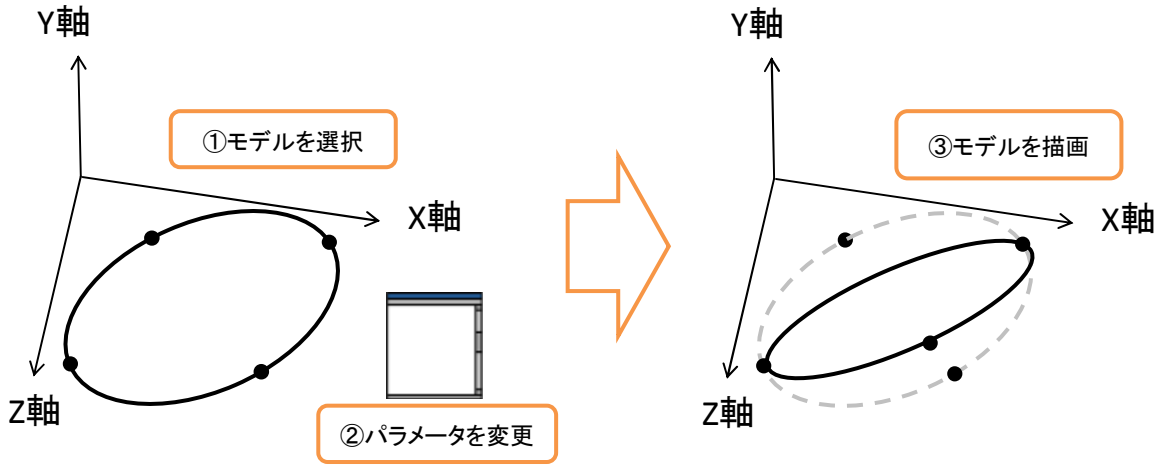
※移動の場合は，尺度変更中のモデルのみ描画し，複写の場合は，選択したモデルと尺度変更中のモデルを描画する．

各モデルの基準は次の通りである．

モデル名	基準	モデル	基準
点マーカ	マーカの大きさ（2.5）	直方体	X 軸方向長さ
線分	2 点間の距離	ウェッジ	X 軸方向長さ
折線	1 点目と最終点との距離	球	半径
円	半径	円錐	底面の半径
楕円	X 軸方向半径	円柱	底面の半径
円弧	半径	トーラス	準線の半径
楕円弧	X 軸方向半径	ベジェ曲面	1 点目と最終点との距離
クロソイド	開始長と終了長の差	NURBS 曲面	1 点目と最終点との距離
ベジェ曲線	1 点目と最終点との距離	複合曲線	各点に外接する直方体の最大長
NURBS 曲線	1 点目と最終点との距離	押出面	各モデルの基準
掃引面	各モデルの基準	回転面	各モデルの基準

4.9.4　モデルのパラメータの編集

本節では，モデルのパラメータの編集について説明する．モデルのパラメータの編集では，各モデルのパラメータの一部をダイアログ上で編集する．パラメータの編集の手順を次に示す．



パラメータの編集では，モデルをマウス操作で選択して編集する． 編集可能なモデルのパラメータを次に示す.

(1) 点マーカ

本項では，点マーカの編集可能なパラメータについて説明する．点マーカのパラメータを次に示す.

パラメータ	型
モデル名	String
配置点 X 座標	double
配置点 Y 座標	double
配置点 Z 座標	double
マーカコード	int
尺度	double

(2) 線分

本項では，線分の編集可能なパラメータについて説明する．線分のパラメータを次に示す.

パラメータ	型
モデル名	String
始点 X 座標	double
始点 Y 座標	double
始点 Z 座標	double
終点 X 座標	double
終点 Y 座標	double
終点 Z 座標	double

(3) 折線

本項では，折線の編集可能なパラメータについて説明する．折線のパラメータを次に示す.

パラメータ	型
モデル名	String
頂点番号	int
X 座標	double
Y 座標	double
Z 座標	double

折線では，各頂点に対して編集を行うため，各座標は頂点番号に対応した値を表示する．

(4) 円

本項では，円の編集可能なパラメータについて説明する．円のパラメータを次に示す．

パラメータ	型
モデル名	String
配置点 X 座標	double
配置点 Y 座標	double
配置点 Z 座標	double
半径	double

(5) 楕円

本項では，楕円の編集可能なパラメータについて説明する．楕円のパラメータを次に示す．

パラメータ	型
モデル名	String
配置点 X 座標	double
配置点 Y 座標	double
配置点 Z 座標	double
X 方向半径	double
Y 方向半径	double

(6) 円弧

本項では，円弧の編集可能なパラメータについて説明する．円弧のパラメータを次に示す．

パラメータ	型
モデル名	String
配置点 X 座標	double
配置点 Y 座標	double
配置点 Z 座標	double
半径	double
向きフラグ	int
始角	double
終角	double

(7) 楕円弧

本項では，楕円弧の編集可能なパラメータについて説明する．楕円弧のパラメータを次に示す.

パラメータ	型
モデル名	String
配置点 X 座標	double
配置点 Y 座標	double
配置点 Z 座標	double
X 方向半径	double
Y 方向半径	double
向きフラグ	int
始角	double
終角	double

(8) ベジェ曲線

本項では，ベジェ曲線の編集可能なパラメータについて説明する．ベジェ曲線のパラメータを次に示す.

パラメータ	型
モデル名	String
頂点番号	int
X 座標	double
Y 座標	double
Z 座標	double
開閉区分	int

ベジェ曲線では，各頂点に対して編集を行うため，各座標は頂点番号に対応した値を表示する.

(9) NURBS 曲線

本項では，NURBS 曲線の編集可能なパラメータについて説明する．NURBS 曲線のパラメータを次に示す.

パラメータ	型
モデル名	String
頂点番号	int
X 座標	double
Y 座標	double

Z 座標	double
重み	double
開閉区分	int

NURBS 曲線では，各頂点に対して編集を行うため，各座標は頂点番号に対応した値を表示する．

(10) クロソイド

本項では，クロソイドの編集可能なパラメータについて説明する．クロソイドのパラメータを次に示す．

パラメータ	型
モデル名	String
配置点 X 座標	double
配置点 Y 座標	double
配置点 Z 座標	double
パラメータ	int
向き	int
回転角	double
開始曲線長	double
終了曲線長	double

(11) 直方体

本項では，直方体の編集可能なパラメータについて説明する．直方体のパラメータを次に示す．

パラメータ	型
モデル名	String
配置点 X 座標	double
配置点 Y 座標	double
配置点 Z 座標	double
X 軸方向長さ	double
Y 軸方向長さ	double
Z 軸方向長さ	double

(12) ウェッジ

本項では，ウェッジの編集可能なパラメータについて説明する．ウェッジのパラメータを次に示す．

パラメータ	型
モデル名	String
配置点 X 座標	double
配置点 Y 座標	double
配置点 Z 座標	double
X 軸方向長さ	double
Y 軸方向長さ	double
Z 軸方向長さ	double
上辺の長さ	double

(13) 球

本項では，球の編集可能なパラメータについて説明する．球のパラメータを次に示す．

パラメータ	型
モデル名	String
配置点 X 座標	double
配置点 Y 座標	double
配置点 Z 座標	double
半径	double

(14) 円錐

本項では，円錐の編集可能なパラメータについて説明する．円錐のパラメータを次に示す．

パラメータ	型
モデル名	String
配置点 X 座標	double
配置点 Y 座標	double
配置点 Z 座標	double
半径	double
高さ	double
角度	double

(15) 円柱

本項では，円柱の編集可能なパラメータについて説明する．円柱のパラメータを次に示す．

パラメータ	型
モデル名	String

配置点 X 座標	double
配置点 Y 座標	double
配置点 Z 座標	double
半径	double
高さ	double

(16) トーラス

本項では，トーラスの編集可能なパラメータについて説明する．トーラスのパラメータを次に示す.

パラメータ	型
モデル名	String
配置点 X 座標	double
配置点 Y 座標	double
配置点 Z 座標	double
準線半径	double
母線半径	double

(17) ベジェ曲面

本項では，ベジェ曲面の編集可能なパラメータについて説明する．ベジェ曲面のパラメータを次に示す.

パラメータ	型
モデル名	String
頂点番号	int
配置点 X 座標	double
配置点 Y 座標	double
配置点 Z 座標	double

ベジェ曲面では，各頂点に対して編集を行うため，各座標は頂点番号に対応した値を表示する.

(18) NURBS 曲面

本項では，NURBS 曲面の編集可能なパラメータについて説明する．NURBS 曲面のパラメータを次に示す.

パラメータ	型
モデル名	String

頂点番号	int
配置点 X 座標	double
配置点 Y 座標	double
配置点 Z 座標	double
重み	double

NURBS 曲面では，各頂点に対して編集を行うため，各座標は頂点番号に対応した値を表示する．

(19) 引出線

本項では，引出線の編集可能なパラメータについて説明する．引出線のパラメータを次に示す．

パラメータ	型
モデル名	String
頂点番号	int
配置点 X 座標	double
配置点 Y 座標	double
配置点 Z 座標	double
文字列	String

引出線では，各頂点に対して編集を行うため，各座標は頂点番号に対応した値を表示する．

(20) バルーン

本項では，バルーンの編集可能なパラメータについて説明する．バルーンのパラメータを次に示す．

パラメータ	型
モデル名	String
頂点番号	int
配置点 X 座標	double
配置点 Y 座標	double
配置点 Z 座標	double
文字列	String

バルーンでは，各頂点に対して編集を行うため，各座標は頂点番号に対応した値を表示する．

4.10　部品の作成

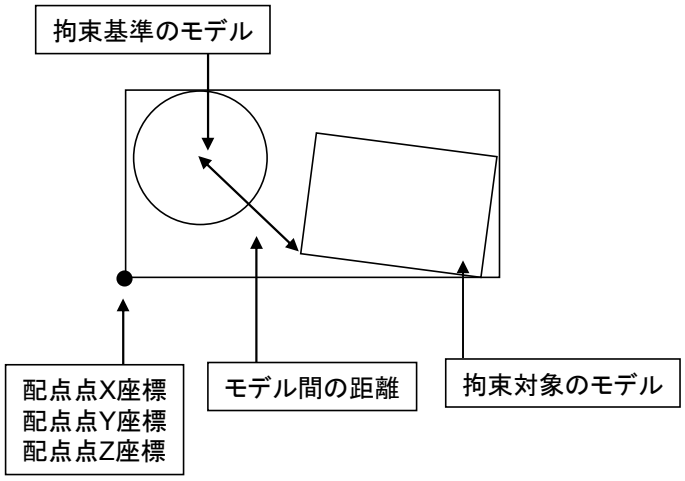
本章では，部品の作成について説明する．本章では，まず，モデルの拘束によって生成されるモデルを管理するため，複合図形を定義する．次に，モデル拘束方法について説明する．最後に，作成した部品の登録方法，登録した部品の挿入方法について説明する．

4.10.1　複合図形

本節では，複合図形について説明する．

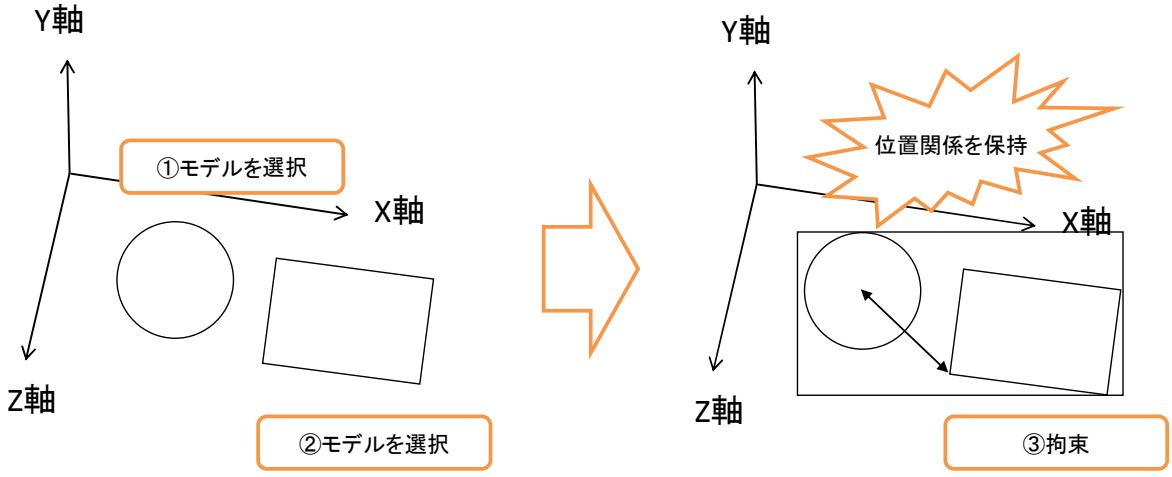
(1) 幾何情報

複合図形の幾何情報を次に示す．



4.10.2　モデルの拘束

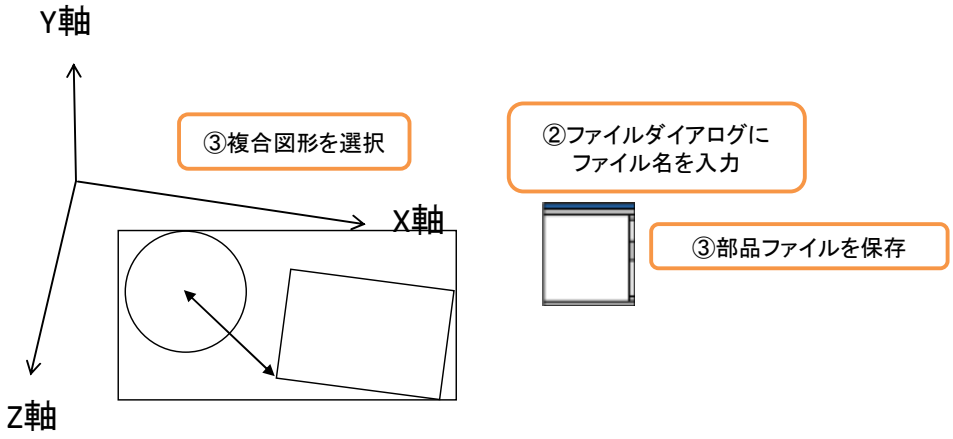
本節では，モデルの拘束について説明する．モデルを拘束方法は，拘束するモデルをマウス操作で選択する．モデルの拘束方法を次に示す．



モデルの拘束では，まず，拘束基準となるモデルをマウス操作で選択する．次に，拘束対象となるモデルを同じくマウス操作で選択する．最後に，選択した２つのモデルの位置関係を算出し，各パラメータを保持した複合図形を生成する．

4.10.3　部品の登録

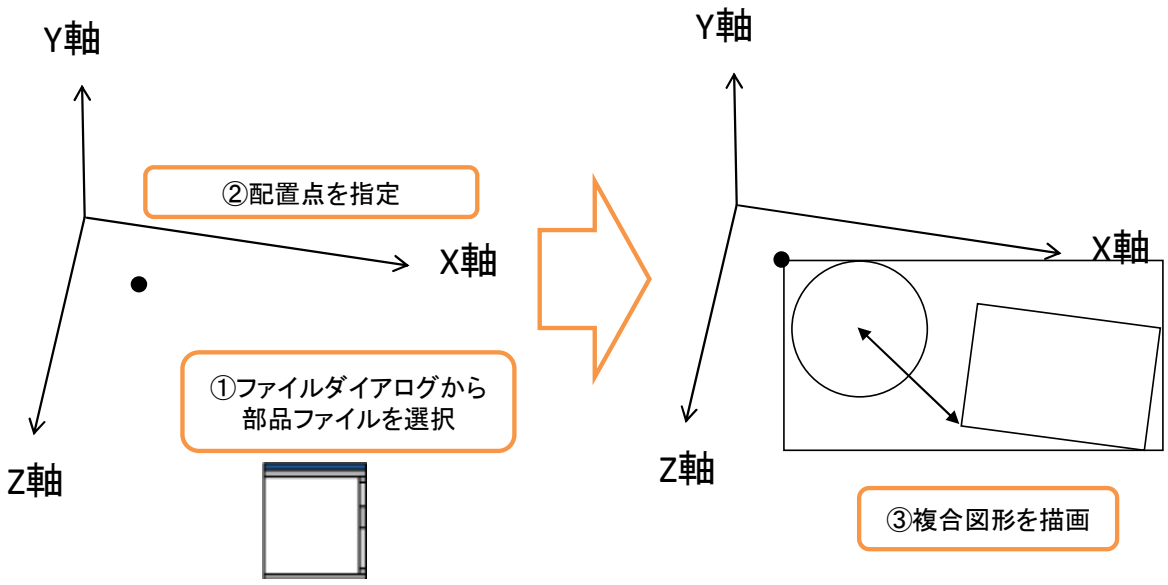
本節では，部品の登録について説明する．部品の登録方法を次に示す．



部品の登録方法は，まず，複合図形をマウス操作で選択する．次に，ファイルダイアログで保存場所とファイル名を入力する．部品の情報を保持したファイルが作成される．

4.10.4　部品の挿入

本節では，部品の挿入について説明する．部品の挿入方法を次に示す．



部品の挿入方法として，まず，ファイルダイアログから部品ファイルを選択する．次に，モデル空間上にマウス操作で部品の挿入位置を指定する．最後に，挿入位置を基点に部品（複合図形）を描画する．ただし，挿入位置の指定の際の座標は，座標軸で構成される平面に投影することとする．

4.11　ファイルの入出力

本章では，本システムにおける入出力ファイルフォーマットについて説明する．本ファイルフォーマットは，わが国の CAD データ変換標準フォーマットである SXF(scadec data exchange format)レベル 2　ver.3.1 の SFC 形式を参考に作成する．

4.11.1　ファイルフォーマット

本節では，本システムにおける入出力ファイルフォーマットについて説明する．

(1) ファイル形式

本項では，ファイル形式として，要素の定義と文字コードについて解説する．

要素の定義

本ファイルフォーマットでは，１つの要素を「#xx = 要素名」の形式で記述する．ここで，xx は任意の整数とする．ファイルフォーマットの例を次に示す．

```
#700 = line_feature(“線分 1”,1,1,1,1,100.0,0.0,0.0,200.0,0.0,0.0)
```

文字コード

本ファイルフォーマットにおける文字コードとして，1 バイト文字には ASCII を使用し，2 バイト文字には SHIFT-JIS を使用する．

(2) データ形式

本項では，データ形式として，実数，文字列，要素名と集合型データについて説明する．

実数

実数は，先頭に正と負の符号をもった 10 進数の実数値で表現する．符号が省略された場合は正の値とする．指数表現や桁区切りの「,」(カンマ)は使用しない．

文字列

文字列を表現する場合は，対象となる文字列を「”」(ダブルコーテーション)で囲む．文字列の最大長は，C 言語で使用される NULL 文字を含まず 256 バイトとする．文字列は，入力されたコードのまま記述する．

要素名

本ファイルフォーマットで使用する要素名と日本語名を次に示す．

要素名	日本語名	要素名	日本語名
layer_feature	レイヤ	bezie_surface_feature	ベジェ曲面
text_font_feature	文字フォント	nurba_surface_feature	NURBS 曲面
point_marker_feature	点マーカ	composite_curve_feature	複合曲線定義
line_feature	線分	text_string_feature	文字列
polyline_feature	折線	linear_dim_feature	直線寸法
circle_feature	円	curve_dim_feature	弧長寸法
ellipse_feature	円弧	angular_dim_feature	角度寸法
arc_feature	楕円	radius_dim_feature	半径寸法
ellipse_arc_feature	楕円弧	diameter_dim_feature	直径寸法
clothoid_feature	クロソイド曲線	label_feature	引出線
bezier_spline_feature	ベジェ曲線	balloon_feature	バルーン

nurbs_spline_feature	NURBS 曲線	sfig_org_feature	複合図形定義
block_feature	直方体	surface_linear_extrusion_feature	押出面
wedge_feature	ウェッジ	surface_sweep_feature	掃引面
sphere_feature	球	surface_revolution_feature	回転面
cone_feature	円錐		
cylinder_feature	円柱		
torus_feature	トーラス		

集合型データ

集合型データを表現する場合は，データを「(」と「)」で囲む．データとデータの区切りには「,」を使用する．集合型データを次に示す．

集合型データタイプ	表現
集合	(1,2,3,4,5)
文字列集合	("hello","world")
集合の集合	((1,2,3,4,5),("hello","world"))

(3) テーブル要素のファイル仕様

本項では，テーブル要素のファイル仕様として，レイヤと文字フォントについて説明する．

レイヤ

レイヤは，2 個のパラメータで構成される．レイヤの定義を次に示す．

```
#xx = layer_feature(レイヤ名,表示/非表示フラグ)
```

文字フォント

文字フォントは，1 個のパラメータで構成される．線幅の定義を次に示す．

```
#xx = text_font_feature(文字フォント)
```

(4) 点・曲線要素のファイル仕様

本項では，点・曲線要素である，点マーカ，線分，折線，円，円弧，楕円，楕円弧，クロソイド，ベジェ曲線と NURBS 曲線のファイル仕様について説明する．

点マーカ

点マーカは，18 個のパラメータで構成される．点マーカの定義を次に示す．

```
#xx = point_marker_feature(図形 ID, 図形名, レイヤコード, 色コード,
                           x 座標, y 座標, z 座標,
                           マーカの種類, 尺度,
                           x 軸ベクトル x 要素, x 軸ベクトル y 要素, x 軸ベクトル z 要素,
                           y 軸ベクトル x 要素, y 軸ベクトル y 要素, y 軸ベクトル z 要素,
                           z 軸ベクトル x 要素, z 軸ベクトル y 要素, z 軸ベクトル z 要素)
```

線分

線分は，21 個のパラメータで構成される．線分の定義を次に示す．

```
#xx =line_feature(図形 ID, 図形名, レイヤコード, 色コード, 線種コード, 線幅コード,
                  始点 x 座標, 始点 y 座標, 始点 z 座標,
                  終点 x 座標, 終点 y 座標, 終点 z 座標,
                  x 軸ベクトル x 要素, x 軸ベクトル y 要素, x 軸ベクトル z 要素,
                  y 軸ベクトル x 要素, y 軸ベクトル y 要素, y 軸ベクトル z 要素,
                  z 軸ベクトル x 要素, z 軸ベクトル y 要素, z 軸ベクトル z 要素)
```

折線

折線は，19 個のパラメータで構成される．折線の定義を次に示す．

```
#xx = polyline_feature(図形 ID, 図形名, レイヤコード, 色コード, 線種コード, 線幅コード,
                       頂点数,
                       x 座標　配列, y 座標　配列, z 座標　配列,
                       x 軸ベクトル x 要素, x 軸ベクトル y 要素, x 軸ベクトル z 要素,
                       y 軸ベクトル x 要素, y 軸ベクトル y 要素, y 軸ベクトル z 要素,
                       z 軸ベクトル x 要素, z 軸ベクトル y 要素, z 軸ベクトル z 要素)
```

円

円は，19 個のパラメータで構成される．円の定義を次に示す．

```
#xx = circle_feature(図形 ID, 図形名, レイヤコード, 色コード, 線種コード, 線幅コード,
                     配置点 x 座標, 配置点 y 座標, 配置点 z 座標,
                     半径,
                     x 軸ベクトル x 要素, x 軸ベクトル y 要素, x 軸ベクトル z 要素,
                     y 軸ベクトル x 要素, y 軸ベクトル y 要素, y 軸ベクトル z 要素,
                     z 軸ベクトル x 要素, z 軸ベクトル y 要素, z 軸ベクトル z 要素)
```

円弧

円弧は，22 個のパラメータで構成される．円弧の定義を次に示す．

```
#xx = arc_feature(図形 ID, 図形名, レイヤコード, 色コード, 線種コード, 線幅コード,
                  配置点 x 座標, 配置点 y 座標, 配置点 z 座標,
                  半径,
                  向き,
                  始角, 終角,
                  x 軸ベクトル x 要素, x 軸ベクトル y 要素, x 軸ベクトル z 要素,
                  y 軸ベクトル x 要素, y 軸ベクトル y 要素, y 軸ベクトル z 要素,
                  z 軸ベクトル x 要素, z 軸ベクトル y 要素, z 軸ベクトル z 要素)
```

楕円

楕円は，20 個のパラメータで構成される．楕円の定義を次に示す.

```
#xx = ellipse_feature(図形 ID, 図形名, レイヤコード, 色コード, 線種コード, 線幅コード,
                      配置点 x 座標, 配置点 y 座標, 配置点 z 座標,
                      x 方向半径, y 方向半径,
                      x 軸ベクトル x 要素, x 軸ベクトル y 要素, x 軸ベクトル z 要素,
                      y 軸ベクトル x 要素, y 軸ベクトル y 要素, y 軸ベクトル z 要素,
                      z 軸ベクトル x 要素, z 軸ベクトル y 要素, z 軸ベクトル z 要素)
```

楕円弧

楕円弧は，23 個のパラメータで構成される．楕円弧の定義を次に示す.

```
#xx = ellipse_arc_feature(図形 ID, 図形名, レイヤコード, 色コード, 線種コード, 線幅コード,
                          配置点 x 座標, 配置点 y 座標, 配置点 z 座標,
                          x 方向半径, y 方向半径,
                          向き,
                          始角, 終角,
                          x 軸ベクトル x 要素, x 軸ベクトル y 要素, x 軸ベクトル z 要素,
                          y 軸ベクトル x 要素, y 軸ベクトル y 要素, y 軸ベクトル z 要素,
                          z 軸ベクトル x 要素, z 軸ベクトル y 要素, z 軸ベクトル z 要素)
```

クロソイド

クロソイドは，22 個のパラメータで構成される．クロソイドの定義を次に示す.

```
#xx = clothod_feature(図形 ID, 図形名, レイヤコード, 色コード, 線種コード, 線幅コード,
                      始点 x 座標, 始点 y 座標, 始点 z 座標,
                      パラメータ, 向き,
                      開始曲線長, 終了曲線長,
                      x 軸ベクトル x 要素, x 軸ベクトル y 要素, x 軸ベクトル z 要素,
                      y 軸ベクトル x 要素, y 軸ベクトル y 要素, y 軸ベクトル z 要素,
                      z 軸ベクトル x 要素, z 軸ベクトル y 要素, z 軸ベクトル z 要素)
```

ベジェ曲線

ベジェ曲線は，21 個のパラメータで構成される．ベジェ曲線の定義を次に示す．

```
#xx = bezier_spline_feature(図形 ID，図形名，レイヤコード，色コード，線種コード，線幅コード，
                            開閉区分，
                            制御点数，
                            階数，
                            x 座標 配列，y 座標 配列，z 座標 配列，
                            x 軸ベクトル x 要素，x 軸ベクトル y 要素，x 軸ベクトル z 要素，
                            y 軸ベクトル x 要素，y 軸ベクトル y 要素，y 軸ベクトル z 要素，
                            z 軸ベクトル x 要素，z 軸ベクトル y 要素，z 軸ベクトル z 要素)
```

NURBS 曲線

NURBS 曲線は，23 個のパラメータで構成される．NURBS 曲線の定義を次に示す．

```
#xx = nurbs_spline_feature(図形 ID，図形名，レイヤコード，色コード，線種コード，線幅コード，
                           開閉区分，
                           制御点数，
                           階数，
                           knot ベクトル 配列，
                           x 座標 配列，y 座標 配列，z 座標 配列，
                           制御点の重み 配列，
                           x 軸ベクトル x 要素，x 軸ベクトル y 要素，x 軸ベクトル z 要素，
                           y 軸ベクトル x 要素，y 軸ベクトル y 要素，y 軸ベクトル z 要素，
                           z 軸ベクトル x 要素，z 軸ベクトル y 要素，z 軸ベクトル z 要素)
```

(5) 面要素のファイル仕様

本項では，面要素である直方体，ウェッジ，球，円錐，円柱，トーラス，ベジェ曲面，NURBS 曲面，複合曲線定義，押出面，掃引面と回転面のファイル仕様について説明する．

直方体

直方体は，21 個のパラメータで構成される．直方体の定義を次に示す．

```
#xx = block_feature(図形 ID，図形名，レイヤコード，色コード，線種コード，線幅コード，
                    配置点 x 座標，配置点 y 座標，配置点 z 座標，
                    x 軸方向の長さ，y 軸方向の長さ，z 軸方向の長さ，
                    x 軸ベクトル x 要素，x 軸ベクトル y 要素，x 軸ベクトル z 要素，
                    y 軸ベクトル x 要素，y 軸ベクトル y 要素，y 軸ベクトル z 要素，
                    z 軸ベクトル x 要素，z 軸ベクトル y 要素，z 軸ベクトル z 要素)
```

ウェッジ

ウェッジは，22 個のパラメータで構成される．ウェッジの定義を次に示す．

```
#xx = wedge_feature(図形 ID，図形名，レイヤコード，色コード，線種コード，線幅コード，
                    配置点 x 座標，配置点 y 座標，配置点 z 座標，
                    x 軸方向の長さ，y 軸方向の長さ，z 軸方向の長さ，
                    x 軸方向の短い方の長さ，
                    x 軸ベクトル x 要素，x 軸ベクトル y 要素，x 軸ベクトル z 要素，
                    y 軸ベクトル x 要素，y 軸ベクトル y 要素，y 軸ベクトル z 要素，
                    z 軸ベクトル x 要素，z 軸ベクトル y 要素，z 軸ベクトル z 要素)
```

球

球は，20 個のパラメータで構成される．球の定義を次に示す．

```
#xx = sphere_feature(図形 ID，図形名，レイヤコード，色コード，線種コード，線幅コード，
                     配置点 x 座標，配置点 y 座標，配置点 z 座標，
                     半径，
                     x 軸ベクトル x 要素，x 軸ベクトル y 要素，x 軸ベクトル z 要素，
                     y 軸ベクトル x 要素，y 軸ベクトル y 要素，y 軸ベクトル z 要素，
                     z 軸ベクトル x 要素，z 軸ベクトル y 要素，z 軸ベクトル z 要素)
```

円錐

円錐は，21 個のパラメータで構成される．円錐の定義を次に示す．

```
#xx = cone_feature(図形 ID，図形名，イヤコード，色コード，線種コード，線幅コード，
                   配置点 x 座標，配置点 y 座標，配置点 z 座標，
                   高さ，
                   半径，
                   角度，
                   x 軸ベクトル x 要素，x 軸ベクトル y 要素，x 軸ベクトル z 要素，
                   y 軸ベクトル x 要素，y 軸ベクトル y 要素，y 軸ベクトル z 要素，
                   z 軸ベクトル x 要素，z 軸ベクトル y 要素，z 軸ベクトル z 要素)
```

円柱

円柱は，20 個のパラメータで構成される．円柱の定義を次に示す．

```
#xx = cylinder_feature(図形 ID，図形名，レイヤコード，色コード，線種コード，線幅コード，
                       配置点 x 座標，配置点 y 座標，配置点 z 座標，
                       高さ，
                       半径，
                       x 軸ベクトル x 要素，x 軸ベクトル y 要素，x 軸ベクトル z 要素，
                       y 軸ベクトル x 要素，y 軸ベクトル y 要素，y 軸ベクトル z 要素，
```

```
                    z 軸ベクトル x 要素，z 軸ベクトル y 要素，z 軸ベクトル z 要素）
```

トーラス

トーラスは，20 個のパラメータで構成される．トーラスの定義を次に示す.

```
#xx = torus_feature(図形 ID，図形名，レイヤコード，色コード，線種コード，線幅コード，
                    配置点 x 座標，配置点 y 座標，中心点 z 座標，
                    準線の半径，母線の半径，
                    x 軸ベクトル x 要素，x 軸ベクトル y 要素，x 軸ベクトル z 要素，
                    y 軸ベクトル x 要素，y 軸ベクトル y 要素，y 軸ベクトル z 要素，
                    z 軸ベクトル x 要素，z 軸ベクトル y 要素，z 軸ベクトル z 要素）
```

ベジェ曲面

ベジェ曲面は，21 個のパラメータで構成される．ベジェ曲面の定義を次に示す.

```
#xx = bezie_surface_feature(図形 ID，図形名，レイヤコード，色コード，線種コード，線幅コード，
                            制御点数，
                            平面 階数，奥行 次数，
                            x 座標 配列，y 座標 配列，z 座標 配列，
                            x 軸ベクトル x 要素，x 軸ベクトル y 要素，x 軸ベクトル z 要素，
                            y 軸ベクトル x 要素，y 軸ベクトル y 要素，y 軸ベクトル z 要素，
                            z 軸ベクトル x 要素，z 軸ベクトル y 要素，z 軸ベクトル z 要素）
```

NURBS 曲面

NURBS 曲面は，25 個のパラメータで構成される．NURBS 曲面の定義を次に示す.

```
#xx = nurbs_surface_feature(図形 ID，図形名，レイヤコード，色コード，線種コード，線幅コード，
                            開閉区分，
                            制御数，
                            平面 次数，奥行 次数，
                            平面 knot ベクトル 配列，奥行 knot ベクトル 配列，
                            制御点の重み 配列，
                            x 座標 配列，y 座標 配列，z 座標 配列，
                            x 軸ベクトル x 要素，x 軸ベクトル y 要素，x 軸ベクトル z 要素，
                            y 軸ベクトル x 要素，y 軸ベクトル y 要素，y 軸ベクトル z 要素，
                            z 軸ベクトル x 要素，z 軸ベクトル y 要素，z 軸ベクトル z 要素）
```

複合曲線定義

複合曲線定義は，6 個のパラメータで構成される．また，複合曲線定義は，他のフィーチャの出力形と異なり，折線，円弧，楕円弧とベジェ曲線を要素として保持する．そのため, composite_curve_feature で複合曲線定義の要素を囲む. 複合曲線定義の定義を次に示す.

```
#xx = composite_curve_feature(図形 ID, 図形名, レイヤコード, 色コード, 線種コード, 線幅コード)
（折線, 円弧, 楕円弧, ベジェ曲線のフィーチャを出力）
#xx = composite_curve_ feature(図形名)
```

押出面

押出面は，9 個のパラメータで構成される．また，押出面は，他のフィーチャの出力形式と異なり，押出面を構成する要素を保持する．押出面の定義を次に示す.

```
#xx = surface_linear_extrusion_feature(図形 ID, 図形名, レイヤコード, 色コード, 線種コード, 線
                                       幅コード,
                                       x 軸ベクトル, y 軸ベクトル, z 軸ベクトル)
（押出面を構成するフィーチャの出力）
```

掃引面

掃引面は，6 個のパラメータで構成される．また，押出し面は，他のフィーチャの出力形式と異なり，掃引面の要素と掃引線の要素を保持する．掃引面の定義を次に示す.

```
#xx = surface_sweep_feature(図形 ID, 図形名, レイヤコード, 色コード, 線種コード, 線幅コード)
（掃引面の要素となるフィーチャの出力）
（掃引線の要素となるフィーチャの出力）
```

回転面

回転面は，9 個のパラメータで構成される．また，回転面は，他のフィーチャの出力形式と異なり，回転面を構成する要素を保持する．回転面の定義を次に示す.

```
#xx = surface_revolution_feature(図形 ID, 図形名, レイヤコード, 色コード, 線種コード, 線幅コ
                                 ード,
                                 回転軸の x 座標, 回転軸の y 座標, 回転軸の z 座標)
（回転面を構成するフィーチャの出力）
```

(6) 注釈要素のファイル仕様

本項では，注釈要素である文字，直線寸法，弧長寸法，角度寸法，半径寸法，直径寸法，引出線とバルーンのファイル仕様について説明する．

文字要素

文字は，16 個のパラメータで構成される．文字の定義を次に示す．

```
#xx = text_string_feature(図形 ID，図形名，レイヤコード，色コード，フォントコード，
                          文字列，
                          文字配置 x 座標，文字配置 y 座標，文字配置 z 座標，
                          文字列範囲高，文字列範囲幅，文字間隔，
                          回転行列，
                          スラント角度，文字列配置位置，文字方向)
```

直線寸法

直線寸法は，54 個のパラメータで構成される．直線寸法の定義を次に示す．

```
#xx = linear_dim_feature(図形 ID，図形名，レイヤコート，色コード，線種コード，線幅コード，
                         関連付けたモデル名，
                         関連付けたモデルの頂点番号（始点），関連付けたモデルの頂点番号（終
                         点），補助線 1 の有無，
                         補助線 1 の基点 x 座標，補助線 1 の基点 y 座標，補助線 1 の基点 z 座標，
                         補助線 1 の始点 x 座標，補助線 1 の始点 y 座標，補助線 1 の始点 z 座標，
                         補助線 1 の終点 x 座標，補助線 1 の終点 y 座標，補助線 1 の終点 z 座標，
                         補助線 2 有無，
                         補助線 2 基点 x 座標，補助線 2 基点 y 座標，補助線 2 基点 z 座標，
                         補助線 2 始点 x 座標，補助線 2 始点 y 座標，補助線 2 始点 z 座標，
                         補助線 2 終点 x 座標，補助線 2 終点 y 座標，補助線 2 終点 z 座標，
                         始点の矢印コード，始点の矢印の向き，
                         始点の矢印の x 座標，始点の矢印の y 座標，始点の矢印の y 座標，
                         始点の矢印の尺度，
                         終点の矢印コード，終点の矢印の向き，
                         終点の矢印の x 座標，終点の矢印の y 座標，終点の矢印の y 座標，
                         終点の矢印の尺度，
                         寸法値の有無，
                         フォントコード，
                         文字列，
                         文字配置 x 座標，文字配置 y 座標，文字配置 z 座標，
                         文字列範囲高，文字列範囲幅，文字間隔，
                         回転行列，
                         スラント角度，文字列配置位置，文字方向)
```

半径寸法

半径寸法は，27 個のパラメータで構成される．半径寸法の定義を次に示す．

```
#xx = radius_dim_feature(図形 ID, 図形名, レイヤコート, 色コード, 線種コード, 線幅コード,
                         関連付けたモデル名, 寸法線の方向
                         矢印コード, 矢印の向き,
                         矢印の x 座標, 矢印の y 座標, 矢印の z 座標,
                         矢印の尺度,
                         寸法値の有無,
                         フォントコード,
                         文字列,
                         文字配置 x 座標, 文字配置 y 座標, 文字配置 z 座標,
                         文字列範囲高, 文字列範囲幅, 文字間隔,
                         回転行列,
                         スラント角度, 文字列配置位置, 文字方向)
```

直径寸法

直径寸法は，33 個のパラメータで構成される．直径寸法の定義を次に示す．

```
#xx = diameter_dim_feature(図形 ID, 図形名, レイヤコート, 色コード, 線種コード, 線幅コード,
                           関連付けたモデル名, 寸法線の方向
                           始点の矢印コード, 始点の矢印の向き,
                           始点の矢印の x 座標, 始点の矢印の y 座標, 始点の矢印の z 座標,
                           始点の矢印の尺度,
                           終点の矢印コード, 終点の矢印の向き,
                           終点の矢印の x 座標, 終点の矢印の y 座標, 終点の矢印の z 座標,
                           終点の矢印の尺度,
                           寸法値の有無,
                           フォントコード,
                           文字列,
                           文字配置 x 座標, 文字配置 y 座標, 文字配置 z 座標,
                           文字列範囲高, 文字列範囲幅, 文字間隔,
                           回転行列,
                           スラント角度, 文字列配置位置, 文字方向)
```

角度寸法

角度寸法は，55 個のパラメータで構成される．角度寸法の定義を次に示す．

```
#xx = angular_dim_feature(図形 ID, 図形名, レイヤコート, 色コード, 線種コード, 線幅コード,
                          関連付けたモデル名,
                          寸法線の半径, 寸法線の始角, 寸法線の終角,
                          補助線 1 の有無,
                          補助線 1 の基点 x 座標, 補助線 1 の基点 y 座標, 補助線 1 の基点 z 座標,
                          補助線 1 の始点 x 座標, 補助線 1 の始点 y 座標, 補助線 1 の始点 z 座標,
                          補助線 1 の終点 x 座標, 補助線 1 の終点 y 座標, 補助線 1 の終点 z 座標,
                          補助線 2 有無,
                          補助線 2 基点 x 座標, 補助線 2 基点 y 座標, 補助線 2 基点 z 座標,
                          補助線 2 始点 x 座標, 補助線 2 始点 y 座標, 補助線 2 始点 z 座標,
```

```
                        補助線 2 終点 x 座標，補助線 2 終点 y 座標，補助線 2 終点 z 座標,
                        始点の矢印コード，始点の矢印の向き,
                        始点の矢印の x 座標，始点の矢印の y 座標，始点の矢印の y 座標,
                        始点の矢印の尺度,
                        終点の矢印コード，終点の矢印の向き,
                        終点の矢印の x 座標，終点の矢印の y 座標，終点の矢印の y 座標,
                        終点の矢印の尺度,
                        寸法値の有無,
                        フォントコード,
                        文字列,
                        文字配置 x 座標，文字配置 y 座標，文字配置 z 座標,
                        文字列範囲高，文字列範囲幅，文字間隔,
                        回転行列,
                        スラント角度，文字列配置位置，文字方向)
```

弧長寸法

弧長寸法は，55 個のパラメータで構成される．弧長寸法の定義を次に示す.

```
#xx = curve_dim_feature(図形 ID，図形名,レイヤコート,色コード,線種コード,線幅コード,
                        関連付けたモデル名,
                        寸法線の半径，寸法線の始角，寸法線の終角,
                        補助線 1 の有無,
                        補助線 1 の基点 x 座標，補助線 1 の基点 y 座標，補助線 1 の基点 z 座標,
                        補助線 1 の始点 x 座標，補助線 1 の始点 y 座標，補助線 1 の始点 z 座標,
                        補助線 1 の終点 x 座標，補助線 1 の終点 y 座標，補助線 1 の終点 z 座標,
                        補助線 2 有無,
                        補助線 2 基点 x 座標，補助線 2 基点 y 座標，補助線 2 基点 z 座標,
                        補助線 2 始点 x 座標，補助線 2 始点 y 座標，補助線 2 始点 z 座標,
                        補助線 2 終点 x 座標，補助線 2 終点 y 座標，補助線 2 終点 z 座標,
                        始点の矢印コード，始点の矢印の向き,
                        始点の矢印の x 座標，始点の矢印の y 座標，始点の矢印の y 座標,
                        始点の矢印の尺度,
                        終点の矢印コード，終点の矢印の向き,
                        終点の矢印の x 座標，終点の矢印の y 座標，終点の矢印の y 座標,
                        終点の矢印の尺度,
                        寸法値の有無,
                        フォントコード,
                        文字列,
                        文字配置 x 座標，文字配置 y 座標，文字配置 z 座標,
                        文字列範囲高，文字列範囲幅，文字間隔,
                        回転行列,
                        スラント角度，文字列配置位置，文字方向)
```

引出線

引出線は，25 個のパラメータで構成される．引出線の定義を次に示す.

```
#xx = label_feature(図形 ID，図形名，レイヤコード，色コード，線種コード，線幅コード,
```

```
                    頂点数,
                    x 座標 配列, y 座標 配列, z 座標 配列,
                    矢印コード, 矢印倍率,
                    寸法値の有無,
                    フォントコード,
                    文字列,
                    文字配置 x 座標, 文字配置 y 座標, 文字配置 z 座標,
                    文字列範囲高, 文字列範囲幅, 文字間隔,
                    回転行列,
                    スラント角度, 文字列配置位置, 文字方向)
```

バルーン

バルーンは，29 個のパラメータで構成される．バルーンの定義を次に示す．

```
#xx = balloon_feature(図形 ID, 図形名, レイヤコード, 色コード, 線種コード, 線幅コード,
                      頂点数,
                      x 座標 配列, y 座標 配列, z 座標 配列,
                      中心 x 座標, 中心 y 座標, 中心 z 座標,
                      半径,
                      矢印コード, 矢印倍率,
                      寸法値の有無,
                      フォントコード,
                      文字列,
                      文字配置 x 座標, 文字配置 y 座標, 文字配置 z 座標,
                      文字列範囲高, 文字列範囲幅, 文字間隔,
                      回転行列,
                      スラント角度, 文字列配置位置, 文字方向)
```

(7) 複合図形のファイル仕様

本項では，複合図形のファイル仕様について説明する．

複合図形

複合図形は，14 個のパラメータにより構成される．複合図形の定義を次に示す．

```
#xx = sfig_org_feature（図形 ID, 図形名, レイヤコード, 色コード, 線種コード, 線幅コード）

（複合図形となるフィーチャの出力）

#xx = sfig_locate_feature（図形名, 配置点の x 座標, 配置点の y 座標, 配置点の z 座標,
                           x 軸ベクトル, y 軸ベクトル, z 軸ベクトル, フィーチャ間の距離）
```

5　結論

本研究では，まず，OpenGL を用いた 3 次元アプリケーションを開発するために必要な画面仕様，クラス構造，実装仕様の 3 つの仕様を定義した．まず，画面仕様として，3 次元グラフィックスアプリケーションで提供する機能を実現するために必要な仕様を示した．画面仕様の定義においては，3 次元 CAD の開発における参考となるために，平成 20 年度の研究で実施した既存の 3 次元 CAD の調査結果を参考とした．つぎに，クラス構造として，3 次元グラフィックスアプリケーションで扱う情報を管理するためデータ構造を定義した．データ構造の定義においては，他のシステムとの親和性の高いデータ交換を想定して，ISO10303 で定義される Entity や SXF で定義されるフィーチャ仕様を参考とした．そして，実装仕様として，3 次元グラフィックスアプリケーションの各機能について，OpenGL を使用した実装方法について定義した．実装仕様の定義において，平成 20 年度の研究で定義した内容に加え, 3 次元 CAD として必要とされる独自の機能を新たに調査･検討することで，システム開発に必要な全般的な仕様を示した．これらの各仕様については，OpenGL を用いて 3 次元 CAD エンジンを開発する際において，コスト面，機能面，開発面を評価するための有効な参考資料として活用されることが期待される．

本研究では，前述した各仕様をもとに，3 次元グラフィックスアプリケーションの開発を行うとともに，その速度面に関して評価を行った．その結果として，3 次元グラフィックスアプリケーションに実装した各操作においては，ユーザに負担を与えることなく，スムーズなレスポンスを提供できることが分かった．3 次元 CAD において，膨大なデータを描画する場合等は速度の低下が問題となることが予想されるが，OpenGL ではなく，他のグラフィックスアプリケーションを使用した場合も同様の問題が起こる．そのため，グラフィックスではなく，メモリ管理や効率的な内部処理の高速化といった対処が必要である．

結論として，前年度の調査結果や本研究の成果により，OpenGL は 3 次元 CAD を開発する上で，コスト面，機能面，開発面，速度面の全ての面で優れたグラフィックスライブラリであることがわかった．今後は，本研究での成果をもとに，3 次元 CAD の開発が活発化し，建設業界における 3 次元データの流通が促進されることが期待される．

様式－３－３

RESEARCH AND DEVELOPMENT OF 3D-GRAPHICS APPLICATIONS USING “OPENGL“

Tanaka, S. [1] Shibasaki, R. [2] Kitagawa, E. [3] Kubota, S. [4] Monobe, K. [5] Nakamura, K. [6]
[1] Faculty of Informatics, Kansai University [2]Center for Spatial Information Science, The University of Tokyo
[3]Faculty of Management Information, Hannan University [4]Faculty of Software and Information Science, Iwate Prefectural University [5]School of Project Design, Miyagi University [6]Faculty of Information Science and Engineering, Ritsumeikan University

CALS/EC Action Program 2005 under Ministry of Land, Infrastructure and Transport proposes a vision about utilizing three-dimensional (3D) CAD. Under present circumstances, however, 3D CAD data are not utilized in the design and construction phases of civil engineering and construction projects because there is no inexpensive 3D CAD. Therefore, it is strongly desired to develop a domestic 3D CAD that is inexpensive as soon as possible. With this as a backdrop, the authors performed research for developing graphics applications using ″OpenGL″ and prepared a research report.

In this research, we performed research and development of 3D-Graphics applications using ″OpenGL″ as the test research towards practical use based on the research report prepared in the aforementioned preceding research. The investigation content is as follows in the present research.

・Definitions of the user interface specification

In the preceding research, we collected and organized the functions that should be implemented in a 3D CAD engine by investigating commercial 3D CAD engines. Therefore, we defined the user interface specifications that are required to realize these functions.

・Definitions of the class specifications

We defined the class structure required to implement a 3D-Graphics application. Regarding the class structure, we designed it with Entity definitions in the ISO10303 standards and the feature specifications of SXF as guides.

・Definitions of the implementation specifications

We defined implementation specifications for a 3D-Graphics application. Regarding implementation specifications, in addition to the contents collected and organized in the preceding research, we newly defined 2D-3D functions (for creating rotary and sweep surfaces) that will be essential for 3D CAD, functions of editing models (such as parallel translation, rotation, and duplication), and CAD functions (such as grouping and switching layers).

・Systemization

In this research, we performed systemization of 3D-Graphics applications based on the contents collected and organized in the preceding research, the above-stated user interface specifications, the class specifications, and the implementation specifications.

The result of this research proves the feasibility of development of 3D CAD using ″OpenGL″, and serves as a reference work when Japanese vendors develop their own 3D CAD engines. In the future, following the outcome of this research, it is expected that development of 3D CAD engine will be promoted with CAD vendors’ own ideas, and that 3D data will be utilized more widely in the civil engineering and building industries.

KEYWORDS: *3D CAD, OpenGL, graphics library, Model representation*

様式－３－２

研　究　成　果　の　要　約

助成番号	助　成　研　究　名	研 究 者・所 属
第2009-2号	OpenGLを用いた３次元グラフィックスアプリケーションの研究開発	関西大学総合情報学部 教授　田中成典

１．はじめに

国土交通省CALS/ECアクションプログラム2005では，３次元CADの利活用に関する構想が提案されている．しかし，現状では，安価な３次元CADが存在しないため，土木・建設事業の設計・施工フェーズで３次元CADデータが利活用されていない．そのため，国産の安価な３次元CADの早急な開発が切望されている．この背景を受けて，筆者らは，「平成20年度(財)日本建設情報総合センター研究助成」において，OpenGLによるグラフィックスアプリケーションの開発のための調査を行い，調査報告書の作成した．

本研究では，前述の先行研究で作成した調査報告書を基に，実用化に向けての試験研究として，OpenGLを用いた３次元グラフィックスアプリケーションの研究開発を行った．

２．研究内容

本研究では，次の手順をもとに３次元グラフィックスアプリケーションの研究開発を行った．

２．１　画面仕様の定義

先行研究では，市販される３次元CADを調査することで，３次元CADに実装すべき機能を取りまとめた．そこで，本研究では，これらの機能を実現するために必要な画面仕様を定義した．

２．２　クラス仕様の定義

本研究では，３次元グラフィックスアプリケーションの実装に必要なクラス構造を定義した．クラス構造については，システム間での円滑なデータ交換を実現するために，ISO10303規格のEntity定義やSXFのフィーチャ仕様を参考に設計した．本研究で定義したクラス構造の主要な構成要素は，次の通りである．

・モデル情報を保持するクラス群

点，線や円等の幾何要素，押出しや回転操作で作成される３次元モデルの情報を保持するためのクラス構造を定義した．

・注記情報を保持するクラス群

３次元モデルに付加する寸法や引出し線等の注釈要素を保持するためのクラス構造を定義した．

・視覚的情報を保持するクラス群

色，線種や線幅等の表示に関する情報を保持するためのクラス構造を定義した．

２．３　実装仕様の定義

本研究では，３次元CADグラフィックスアプリケーションの実装仕様を定義した．実装仕様については，先行研究で取りまとめた内容に加え，３次元CADとして必須となる２D-３D機能（回転面やスイープ面等の作成），モデル編集機能（平行移動・回転や複写等）やCAD機能（グループ化やレイヤ切り替え等）の実装仕様を新たに定義した．

２．４　システム化

本研究では，先行研究で取りまとめた内容と前述した画面仕様，クラス仕様，実装仕様を基に，３次元グラフィックアプリケーションのシステム化を行った．

３．おわりに

本研究の成果は，OpenGLを用いた３次元CAD開発の実現可能性を明らかとするものであり，わが国のCADベンダが独自の３次元CADを開発する場合の参考資料となる．

今後，本研究の成果を受け，CADベンダでの独自のアイデアを活かした３次元CADの開発が促進されるとともに，土木・建築業界における３次元データの活用場面の拡大が期待される．