
情報メディア工学特論

基本オブジェクト設計

2004/12/21

京都大学高等教育研究開発センター情報メディア教育部門

学術情報メディアセンター連携研究部門(兼任)

大学院工学研究科電気工学専攻(兼担)

小山田耕二

コース概要 (2/2)

□ 特異点ベース可視化技術

- 渦可視化(11/16)
- 等値面表示高速化、DT-MRI可視化(11/30)

□ システム開発技術/OpenGL基礎

- オブジェクト指向システム開発技術(12/7)
- 基本オブジェクト設計(12/14)

□ 可視化システム実装演習

- 等値面表示システム(12/21)
 - ボリュームレンダリング表示システム(1/11)
 - 流線表示システム(1/18)
-

OpenGL ~概要~

□ OpenGLとは？

Silicon Graphics社が中心となり開発された
“グラフィックス処理のためのソフトウェアインタフェース”

～簡単に言うと、プログラマがグラフィックスハードウェアにアクセスする
プログラムを書けるようにしてくれるプログラムインタフェース

多種の異なるハードウェア・プラットフォーム上で実行可能な
効率の高いハードウェア”非依存型”のインタフェースである。

OpenGL ~概要~

□ OpenGL関連ライブラリ

- OpenGL Utility Library (GLU)

視界の方向や射影、多角形のタイリング、表面のレンダリング用の行列の設定などのタスクを実行するために使用する低レベルのOpenGLコマンドを使用したルーチンの提供.
OpenGLの動作環境の一部として提供されている.

- OpenGL Utility Toolkit (GLUT)

ウィンドウを開いたりウィンドウマネージャとの通信にはそれぞれのOSに沿った流儀が必要.そのOSに依存した流儀を意識せずにユーザがプログラムできるようにしたライブラリ.

OpenGL ~導入~

□ Windows

Windows NT系のOSにはGL、GLUライブラリは標準搭載されている。Windows NT系・・・Windows 2000, XPなど

□ Unix系OSやMacintosh

<http://www.opengl.org/>などで配布されているMesaと呼ばれる無料ライブラリを導入する。

OpenGL ~導入 Window編~

GLUTのインストール

1. GLUTのWindows用バイナリを入手する.

<http://www.xmission.com/~nate/glut.html>などから入手できる.

2. バイナリ(glut-3.7.x-bin.zip)を展開する.

zip形式で圧縮されているために展開する.

3. 各ファイルを適切な場所に配置する.

- ・glut.h コンパイラのヘッダファイルのパス
(Visual C++ならC:¥ProgramFiles¥DevStudio¥Vc¥include¥GLなど)
 - ・*.lib コンパイラのライブラリのパス
(C:¥ProgramFiles¥DevStudio¥Vc¥libなど)
 - ・*.dll システムのディレクトリ
(C:¥WINDOWS¥systemやC:¥WINNT¥system32あたり)
-

OpenGL ~導入 UNIX系OS編~

1. ライブラリであるMesaを入手する.

- ・ <http://www.opengl.org/>のhpで左上メニュー”Documentation”中の“OS/Platform Implementations”を選択する.
 - ・ 表示されるPageの最下部にある”Mesa 3D”文章中にある”Mesa”リンクを選択する.
 - ・ “The Mesa 3D Graphics Library” Pageが開き,左メニューの“Download / Install” 中にある”Stable Release (6.2.1)”を選択する.
 - ・ <http://sourceforge.net/projects/mesa3d> のPageが表示されると中段付近に”Latest File Releases”とありFileをダウンロードできる.
 - ・ ”MesaDemos”と”MesaLib”をダウンロードする.
-

OpenGL ~導入 UNIX系OS編~

2. ダウンロードしたファイルを展開する.

分からない場合は<http://mesa3d.org/>を参考.

3. 新たに作成された”Mesa-6.2”ディレクトリに入り,コマンドラインにて”make linux-x86”と入力しコンパイルする.

4. Demoファイルを動作させてみる.

demoを動作させる前に”Mesa lib”ディレクトリ内で
“export LD_LIBRARY_PATH=\${PWD}”と入力することで利用可能に.
例としてMesa/progs/demos内の./gearsを実行してみよう.

5. Header fileとLibraryを適切なディレクトリに移す.

Header fileは/usr/include/GLに、Librariesは/usr/lib/が一般的な置き場所となる.(詳しくは<http://oss.sgi.com/projects/ogl-sample/ABI/>を参照)

OpenGL ~導入 コンパイルの仕方~

□ Unix系OS

cc(あるいはgcc)コマンドに以下のようなオプションを付ける.

```
% cc -I/usr/X11R6/include program.c -L/usr/lib -lglut -IGLU -IGL -IXmu  
-lXi -lXext -lX11 -lm -lpthread
```

□ Windows

コンパイラによって上記と同様の記述が必要.

(Visual C++の場合、GLUT3.7.2以降とVisual C++ 6.0の組み合わせなら自動的にライブラリを読み込んでくれる.)

OpenGL ~プログラミング概論~

□ 1 ウィンドウを開く

□ 1.1 空のウィンドウを開く

最も基本的なプログラムとしてウィンドウを開くプログラムを示す(GLtest1_1.c).
基本的にこのプログラムにコマンドを追加することで描画する.

用いられる関数

- void glutInit(int *argc, char **argv)
 - GLUTおよびOpenGL環境の初期化
 - int glutCreateWindow(char *name)
 - ウィンドウを開く. 引数はそのウィンドウの名前の文字列.
 - void glutDisplayFunc(void (*func) (void))
 - 描画関数. 引数funcは開いたウィンドウ内に描画する関数へのポインタ
 - void glutMainLoop(void)
 - 無限ループ. この関数によりプログラムはイベントの待ち受け状態になる.
-

OpenGL ~プログラミング概論~

□ 1 ウィンドウを開く

□ 1.2 ウィンドウを塗りつぶす

描画関数として、開いたウィンドウを青で塗りつぶす操作を示す(GLtest1_2.c).

用いられる関数

- void glutInitDisplayMode(unsigned int mode)
 - ディスプレイの表示モードを設定.引数がGLUT_RGBAだと色をRGBで指定.
 - int glClearColor(GLclampf R, GLclampf G, GLclampf B, GLclampf A)
 - glClearColor関数で塗りつぶされる色を指定する.Aは不透明度を示す.
 - void glClear(GLbitfield mask)
 - ウィンドウを塗りつぶす.maskには塗りつぶすバッファを指定する.
 - void glFlush(void)
 - まだ実行されていないOpenGL命令を全部実行する.OpenGLは命令を幾つか貯めこんでから実行するため.
-

OpenGL ~プログラミング概論~

□ 2 2次元図形を描く

□ 2.1 線分を引く

描画関数として、線分を表示させる関数を用意する(GLtest2_1.c).

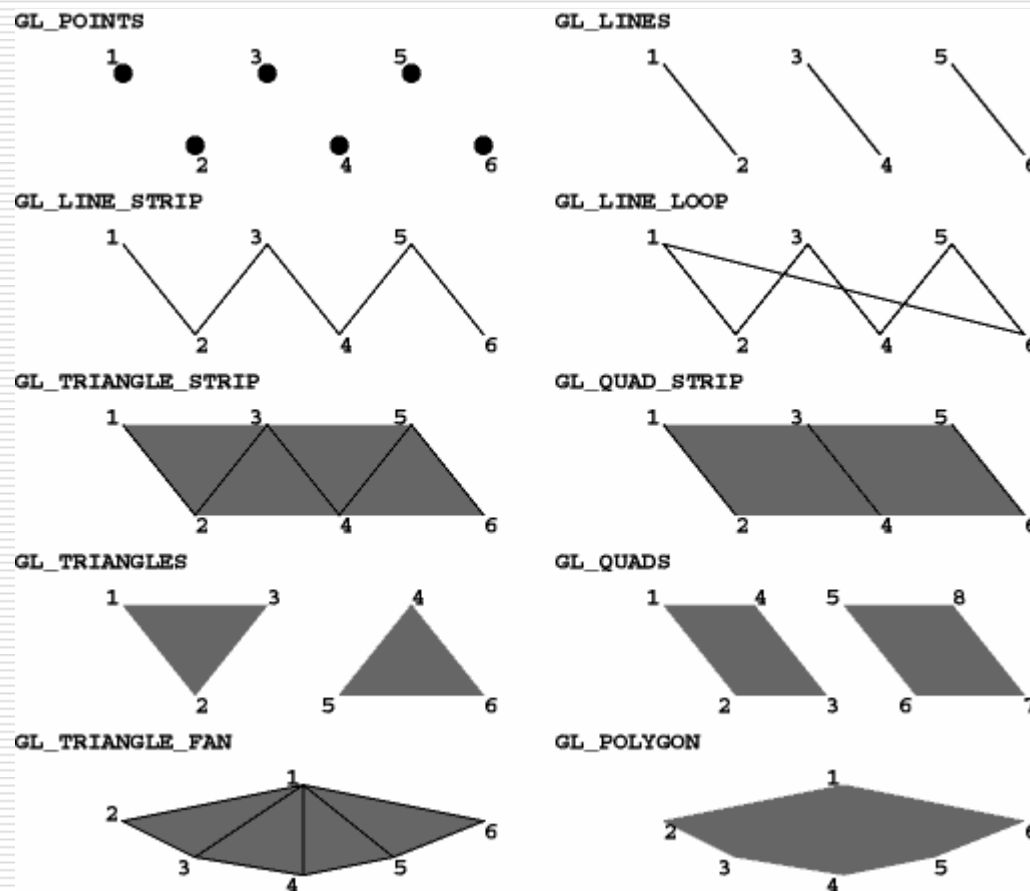
用いられる関数

- void glBegin(GLnum mode)
 - void glEnd(void)
 - 図形を描くには,glBegin~glEndの間に図形の各頂点の座標値を設定する関数を置く.
 - int glVertex2d(GLdouble x, GLdouble y)
 - 2次元の座標値を設定するのに用いられる.
-

OpenGL ~プログラミング概論~

- 2次元図形を描く
- 2.2 図形のタイプ

glBegin()の()内を右図に示すものに変わると座標の指定順によって表示される図形が変わる.



OpenGL ~プログラミング概論~

□ 2 2次元図形を描く

□ 2.3 線に色を付ける

描画関数として、線に色を付けてみる(GLtest2_3.c).

用いられる関数

- int glColor3d(GLdouble r, GLdouble g, GLdouble b)

- glColor3d()はこれから描画するものの色を指定する.r,g,bにはそれぞれ赤、緑、青の強さを0~1の範囲で指定する.

OpenGL ~プログラミング概論~

- 2次元図形を描く
 - 2.4 図形を塗りつぶす

描画関数として、三角形を描き、その面に色を付けてみる(GLtest2_4.c).

OpenGL ~プログラミング概論~

□ 3 座標軸の設定

□ 3.1 座標軸とビューポート

ウィンドウのサイズを変更しても表示内容の大きさが変化しないようにする (GLtest3_1.c).

用いられる関数

- void glViewport(GLint x, GLint y, GLsizei w, GLsizei h)
 - ビューポート(開いたウィンドウ中で実際に描画が行われる領域)を設定する.
- int glLoadIdentity(void)
 - 変換行列を初期化する.
- void glOrtho(GLdouble l, GLdouble r, GLdouble b, GLdouble t, GLdouble n, GLdouble f)
 - ワールド座標系を正規化デバイス座標系に平行投影する行列を変換行列に乘じる.(l = left, r = right, b = bottom, t = top, n = near, f = far)
- void glReshapeFunc(void (*func) (int w, int h))
 - 引数にはウィンドウがリサイズされたときに実行する関数のポインタを与える.

OpenGL ~プログラミング概論~

□ 3 座標軸の設定

□ 3.2 位置やサイズを指定してウィンドウを開く

開くウィンドウの位置やサイズを指定できるようにする(GLtest3_2.c).

用いられる関数

- void glutInitWindowSize(int w, int h)
 - 新たに開くウィンドウの幅と高さを指定する.指定しないときは300 × 300.
- int glutInitWindowPosition(int x, int y)
 - 新たに開くウィンドウの位置を指定する.指定しないときはウィンドウマネージャによってウィンドウを開く位置を決定する.

OpenGL ~プログラミング概論~

□ 4 3次元図形を描く

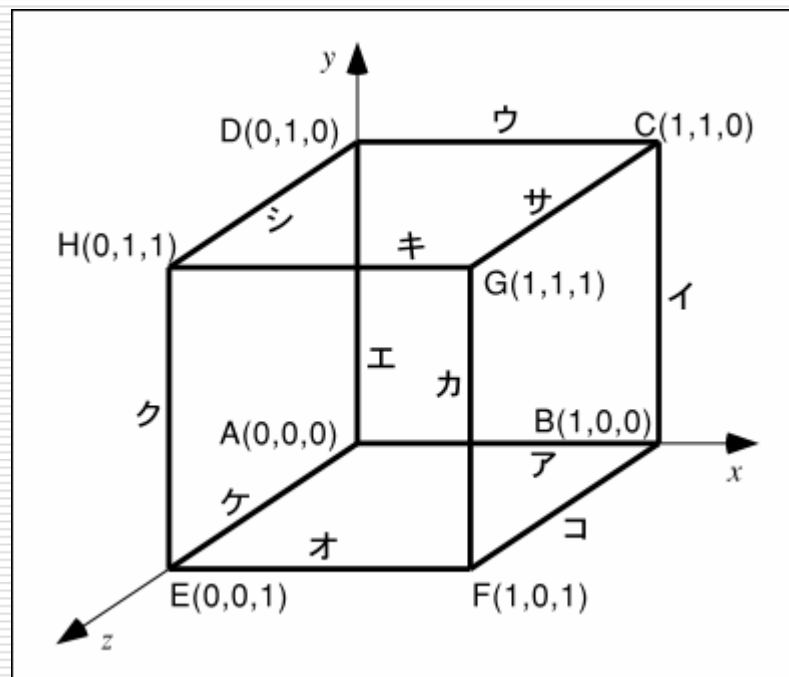
□ 4.1 線画を表示する

右図のような図形を描くために立方体
12辺全ての辺を線分で表示させる。

(GLtest4_1.c)

用いられる関数

- `glVertex3dv(const GLdouble *v)`
 - `glVertex3dv()`は3次元の座標値を指定するのに使う. 引数`v`は3個の要素を持つ配列.



OpenGL ~プログラミング概論~

□ 4 3次元図形を描く

□ 4.2 透視投影する

前のプログラムでは立方体が平行投影されているために正方形しか描かれていない。そこで現実のカメラのように透視投影をもちいる(GLtest4_2.c)。

用いられる関数

- void gluPerspective(GLdouble fovy, GLdouble aspect, GLdouble zNear, GLdouble zFar)
 - 変換行列に透視変換の行列を乗じる。(fovy = カメラの画角, aspect = 画面のアスペクト比, zNear = 手前側の座標, zfar = 奥側の座標)
 - int glTranslated(GLdouble x, GLdouble y, GLdouble z)
 - 変換行列に平行移動の行列を乗じる。
-

OpenGL ~プログラミング概論~

□ 4 3次元図形を描く

□ 4.2 視点位置を変更する

前のプログラムの方法(`glTranslated`や`glRotated`を用いて)で視点の位置を移動することもできるが,もっと簡単に視点位置を指定することもできる(`GLtest4_3.c`).

用いられる関数

- `void gluLookAt(GLdouble ex, GLdouble ey, GLdouble ez,
GLdouble cx, GLdouble cy, GLdouble cz,
GLdouble ux, GLdouble uy, GLdouble uz)`
 - 最初の3つの引数は視点の位置,次の3つの引数は目標の位置,
最後の3つの引数はウィンドウに表示される画像の上方向を示すベクトル
-

OpenGL ~プログラミング概論~

□ 5 シェーディング

前プログラムで描いた立方体の内部に3角形を4面描画する.面ごとに色を付ける代わりに,光を当ててみる.色の代わりに面ごとの法線ベクトルを与える.(GLtest5.c)

直接はシェーディングに関係しないが分かりやすくするためにマウスのクリックによって回転するようにプログラムを組んでいる.

用いられる関数

- void glNormal3dv()
 - 面の法線を設定する.
 - void glEnable(GL_LIGHT0)
 - void glDisable(GL_LIGHT0)
 - OpenGLが持つ0番目の光源の有効(点灯),無効(消灯)を設定する.
 - void glEnable(GL_LIGHTING)
 - void glDisable(GL_LIGHTING)
 - シェーディング処理を行うか、行わないかを設定する.
-

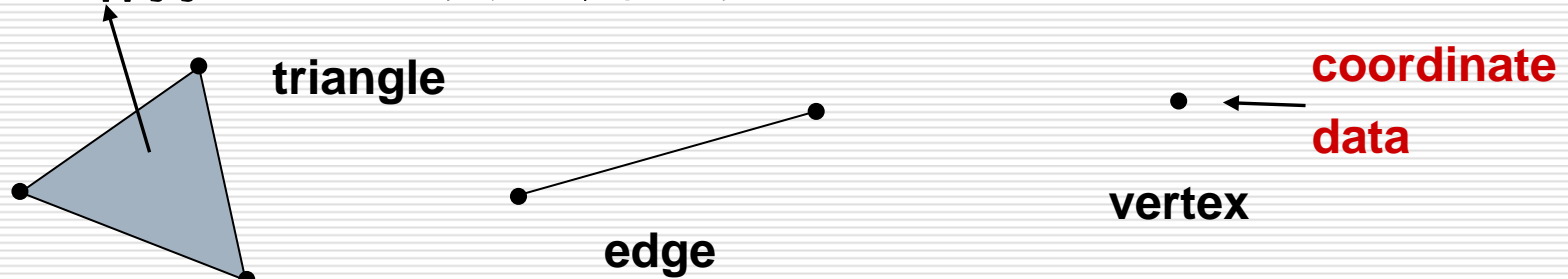
等値面生成システムの実装

関数 `create_isosurface_GEOMETRY`

入力	閾値(float)	CONST
	ボリュームオブジェクトポインタ	*V
出力	ジオメトリオブジェクトポインタ	

関数の出力であるジオメトリオブジェクトとは？

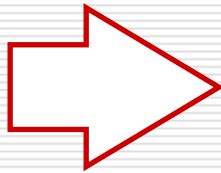
三角形・線分・点の図形を表すオブジェクトであり、それらの図形を構成する頂点に関する情報や面の法線等の情報を保持しているオブジェクトである。



等値面生成システムの実装

・OpenGLとの連動

等値面生成システムにおいて面を表示するには、配布されている `create_Isosurface_GEOMETRY` 関数によって作成された三角形ジオメトリを読み取り、その頂点座標、法線ベクトルをOpenGLに渡すことで等値面を表示する方法が考えられる。



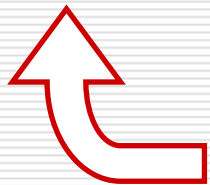
適切なプログラムを書くためにはOpenGLの処理の流れを理解しておく必要がある。

等値面生成システムの実装

・OpenGLの処理手順

最も基本的な処理の流れ

1. 環境の初期化(`glutInit`)
2. ウィンドウを開く(`glutCreateWindow`)
3. ウィンドウ内に絵を描く関数を決める(`glutDisplayFunc`)
4. 何かイベントが起こるのを待つ(`glutMainLoop`)



等値面生成システムで組み込みたい処理

- ファイルの読み込み, Volumeオブジェクトの生成, Geometryオブジェクトの生成
 - あるキーを押すと入力ファイルの変更ができる.
 - あるキーを押すと描画パラメータ(閾値)の変更ができる.
 - あるキーを押すと視線の変更ができる.
-

等値面生成システムの実装

・Volume, Geometryオブジェクト

Volumeオブジェクト、Geometryオブジェクトを利用する操作

1. データファイルからVolumeオブジェクトの生成
2. VolumeオブジェクトからGeometryオブジェクトの生成
3. GeometryオブジェクトからOpenGLによる三角形の描画

操作1はプログラムを動作させた始めと、途中でファイルを変更する場合に必要となる。(少なくともglutKeyboardFunc内で操作)


操作2は基本的にVolumeが変化した場合に操作する必要がある。(glutKeyboardFunc内での操作)

操作3は描画する際に操作(glutDisplayFunc内での操作)

様々な関数から呼び出されるためグローバル変数で宣言すればよい。

等値面生成システムの実装

・ファイルを変更,パラメータを変更,視点を変更



OpenGLにてあるデータを描画中に決められたキーボード上のキーを押すことによって,コマンドライン上からファイル名,パラメータ値,視点の位置等の入力を受け付け,すぐにそれをOpenGLによる描画に反映させる.

```
glutKeyboardFunc ( void (*func) ( unsigned char key, int x, int y ) )
```

引数にはキーがタイプされたときに実行する関数のポインタを与えます.またxとyにはキーがタイプされたときのマウスの位置が渡されます.

上記関数内に記述してしまえば,要求を満たすことができる.

等値面生成システムの実装

・状態遷移図

