平成 25 年度 CG 基礎(GC2 3301)　　平成 25 年 10 月

I. 科目概要

福井幸男・三谷 純・金森由博

秋学期 AB　月曜５・６時限（15:15 – 16:30, 16:45-18:00）

5 時限目　講義、6 時限目　演習（いずれも創成実習室 7C202）

担当教員

１～４回目　福井幸男　fukui@cs.tsukuba.ac.jp　853-5524　総 B 棟 907

５～７回目　三谷　純　mitani@cs.tsukuba.ac.jp　853-2333　総 B 棟 906

８～10 回目　金森由博　kanamori@cs.tsukuba.ac.jp 853-5724　3F828(総 B925)

TA　山田裕貴　yamada@npal.cs.tsukuba.ac.jp

講義に関連した演習課題を行う。レポート課題として提出するものを含む。

参考書　Computer Graphics (技術編 CG 標準テキストブック)　CG-ARTS 協会，¥3,000

購入希望の履修者は次週までに申し込む

II. 日程予定

10 月 7 日(月)　オリエンテーションと受講者確認、授業概要、CG 導入（福井）

OpenGL グラフィックライブラリ演習、演習環境(OpenGL+GLUT)、練習プログラム

10 月 17 日(木)　（曜日振替）直線、円弧の描画アルゴリズムとその演習（福井）

10 月 21 日(月)　2 次元座標変換（同次座標系、変換行列表現、合成変換等）とその演習（福井）

10 月 28 日(月)　3 次元幾何学的変換と投影変換とその演習（福井）

11 月 6 日(水)（曜日振替）曲線のモデリング（ベジエ曲線、B スプライン曲線等）と演習（三谷）

11 月 11 日(月)　3 次元曲面形状モデリングとその演習（三谷）

11 月 18 日(月)　3 次元多面体形状モデリングとその演習（三谷）

11 月 25 日(月)　レンダリングとその演習（金森）

12 月 2 日(月)　レンダリングとその演習（金森）

12 月 9 日(木)（曜日振替）自由トピックス（金森）

12 月 16 日(月)　期末試験（5 限のみ）

サンプルプログラムをコピーして、OpenGL＋GLUT グラフィックスライブラリを使った演習用プログラムをコンパイル、実行して働きをよく理解すること。

サンプルプログラムは下記に置いてある。

http://www.coins.tsukuba.ac.jp/~fukui/CG/ex1.htm

Windows Vista、CentOS（Linux）のいずれの OS システムを使ってもコンパイルできるようなグラフィックプログラムを作成する。Windows を使っても、Linux を使っても良い。両方できるとなお良い。

演習環境

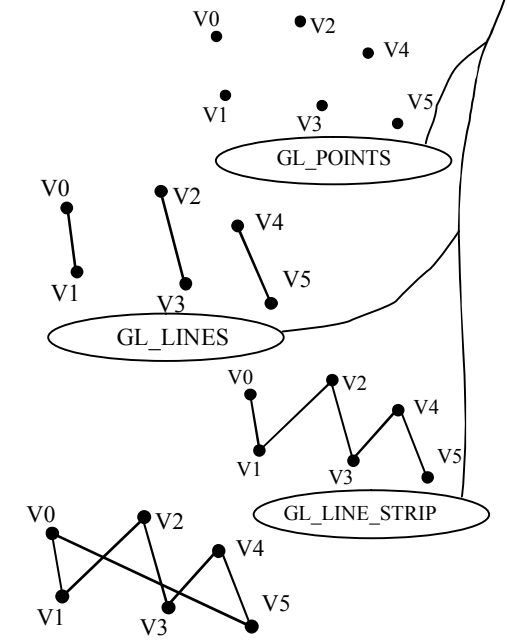
・OpenGL を使うプログラムは、通常と違い、割り込み（対話処理）待ちの無限ループに入る。

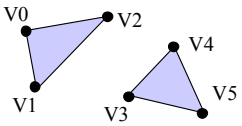
・右のプログラムでは、display( ) 関数内に描画する。

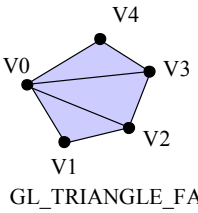
・glBegin( )と glEnd( )の間に描画対象の点情報を指定する。

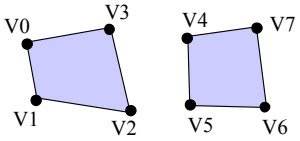
・glBegin( )の引数で幾何描画要素を指定する。

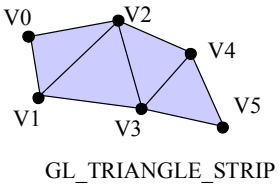
```
// ２次元図形を描く　t1.c
// GL_LINE_STRIP の代わりに他の描画要素を指定してみよう
#include <GL/glut.h> // GLUT ライブラリを使用

void display(void) // 表示関数
{
   glClear(GL_COLOR_BUFFER_BIT); //画面クリア(背景色)
   glColor3d(1.0, 0.0, 0.0); // 描画色指定(red, green, blue)
   glBegin(GL_LINE_STRIP);      // 表示対象(多角形)作成開始
      glVertex2d(-0.8, 0.3); // ２次元の点を指定
      glVertex2d(-0.7, -0.1);
      glVertex2d(-0.1, 0.4);
      glVertex2d( 0.1, -0.2);
      glVertex2d( 0.5, 0.1);
      glVertex2d( 0.8, -0.3);
   glEnd();                          // 表示対象作成終了
   glFlush();                        // 画面に出力
}

int main(int argc, char *argv[])
{
   glutInit(&argc, argv);              // グラフィック初期化
   glutInitDisplayMode(GLUT_RGBA); // 描画モード()
   glutCreateWindow(argv[0]);          // 描画ウィンドウ作成
   glutDisplayFunc(display);           // 描画関数指定

   glClearColor(1.0, 1.0, 1.0, 0.0); // 背景色指定
   glutMainLoop( );                          // 割り込み待ち
   return 0;
}
```

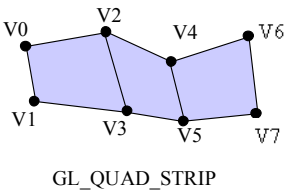


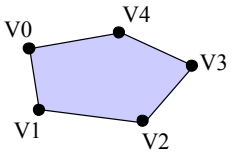












```
// ２次元図形を描く　t2.c
// t1.c では、ウィンドウ枠の縦横比をマウスで変えると、それに
//　応じて図形が歪む。これを歪まないようにするために、
//　ウィンドウ(glViewport で指定)の縦横比に合わせて
//　データ表示領域を変える(glOrtho で指定)

#include <GL/glut.h> // GLUT ライブラリを使用

void display(void) //  表示関数
{
   glClear(GL_COLOR_BUFFER_BIT); //画面クリア(背景色)
   glColor3d(1.0, 0.0, 0.0); //  描画色指定(red, green, blue)
   glBegin(GL_TRIANGLES);      //  表示対象(多角形)作成開始
      glVertex2d(-0.8, 0.3); //  ２次元の点を指定
      glVertex2d(-0.7, -0.1);
      glVertex2d(-0.1, 0.4);
      glVertex2d( 0.1, -0.2);
      glVertex2d( 0.5, 0.1);
      glVertex2d( 0.8, -0.3);
   glEnd( );                          //  表示対象作成終了
   glFlush( );                        //  画面に出力
}

void resize(int w, int h)
{
   glViewport(0, 0, w, h);   //  ウィンドウ全体を表示領域に指定
   glLoadIdentity( );   //  変換行列の初期化
   glOrtho(-w/300.0, w/300.0, -h/300.0, h/300.0, -1.0, 1.0);
}

int main(int argc, char *argv[ ])
{
   glutInitWindowPosition(100, 100); //  ウィンドウ左上の位置
   glutInitWindowSize(320, 240);   //   ウィンドウの大きさ

   glutInit(&argc, argv);               //  グラフィック初期化
   glutInitDisplayMode(GLUT_RGBA); //  描画モード
   glutCreateWindow(argv[0]);          //  描画ウィンドウ作成
   glutDisplayFunc(display);           //  描画関数指定
   glutReshapeFunc(resize);            //  再描画関数指定

   glClearColor(1.0, 1.0, 1.0, 0.0); //  背景色指定
   glutMainLoop( );                          //  割り込み待ち
   return 0;
}
```

t2.c では、ウィンドウの大きさを変更しても、図形は歪まない。大きさも変わらない。ウィンドウを大きくしたときに、表示される図形も歪まずに大きくするためには、どのような工夫が必要か。

t3.c を利用して、マウスで線分を描くプログラムを作成してみよう

```
// マウス操作情報の出力　t3.c
#include <stdio.h>
#include <GL/glut.h>

void display(void)
{ glClear(GL_COLOR_BUFFER_BIT);
   glFlush( );
}

void resize(int w, int h)
{ glViewport(0, 0, w, h);
   glLoadIdentity( );
}

void mouse(int button, int state,
                  int x, int y)
{
   switch (button) {
   case GLUT_LEFT_BUTTON:
      printf("left");       break;
   case GLUT_MIDDLE_BUTTON:
      printf("middle");    break;
   case GLUT_RIGHT_BUTTON:
      printf("right");     break;
   default:                    break;
   }

   printf(" button is ");
   switch (state) {
   case GLUT_UP:
      printf("up");          break;
   case GLUT_DOWN:
      printf("down");       break;
   default:                    break;
   }
   printf(" at (%d, %d)¥n", x, y);
}

void init(void)
{
   glClearColor(1.0, 1.0, 1.0, 0.0);
}

int main(int argc, char *argv[ ] )
{
   glutInitWindowPosition(100, 100);
   glutInitWindowSize(320, 240);
   glutInit(&argc, argv);
   glutInitDisplayMode(GLUT_RGBA);
   glutCreateWindow(argv[0]);
   glutDisplayFunc(display);
   glutReshapeFunc(resize);
   glutMouseFunc(mouse);
   init( );
   glutMainLoop( );
   return 0;
}
```

```
//  プログラム名：t4.c
#include <GL/glut.h>   //  グラフィック用

void display(void)        //  表示関数
{
   int      i;
   double a=1.995128, y0=0., y2, x1=0.069756;
   double b=1.989043, x0=0., x2, y1=0.104528;

   glClear(GL_COLOR_BUFFER_BIT); //   注 5
   glColor3d(1.0, 0.0, 0.0);                  //   注 6
   glBegin(GL_LINE_LOOP);                 //   注 7
      glVertex2d( x0,y0 );                  //   注 8
      glVertex2d( x1,y1 );
      for( i=0; i<=180; i++ ){
         x2 = a*x1-x0;     y2 = b*y1-y0;
         x0 = x1;          y0 = y1;
         x1 = x2;          y1 = y2;
         glVertex2d( x2,y2 );
      }
   glEnd();                                  //   注 7
   glFlush();                                //   注 9
}

void init(void)                           //   初期化設定
{
   glClearColor(1.0, 1.0, 1.0, 0.0); //   注 10
}

int main(int argc, char *argv[ ])
{
   glutInit(&argc, argv);                    //   注 0
   glutInitDisplayMode(GLUT_RGBA); //   注 1
   glutCreateWindow(argv[0]);             //   注 2
   glutDisplayFunc(display);              //   注 3
   init( );
   glutMainLoop( );                          //   注 4
   return 0;
}
```

http://www.center.wakayama-u.ac.jp/~tokoi/opengl/libglut.html から解説引用

注 0 void glutInit(int *argcp, char **argv)

GLUT(Graphic Library Utility Toolkit) および OpenGL 環境を初期化. 引数には main の引数をそのまま渡す.
X Window で使われるオプション -display などはここで処理される.

注 1　void glutInitDisplayMode(unsigned int mode)

ディスプレイの表示モードを設定.
mode に GLUT_RGBA を指定した場合は, 色の指定を RGB (赤緑青, 光の３原色) で行えるようにする.

注 2　int glutCreateWindow(char *name)

ウィンドウを開く. 引数 name はそのウィンドウの名前の文字列で, タイトルバーなどに表示される.
以降の OpenGL による図形の描画等は, 開いたウィンドウに対して行われる. 返戻値は開いたウィンドウの識別子.

注 3　void glutDisplayFunc(void (*func)(void))

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. ウィンドウを再描画する必要があるときに, この関数が実行される.この関数内で図形表示を行う.

注 4　void glutMainLoop(void)

これは無限ループ. この関数を呼び出すことで, プログラムはイベントの待ち受け状態になる. このプログラムは終了しないので、マウスでウィンドウを閉じて強制終了させる

注 5　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. OpenGL が管理する画面上のバッファ（メモリ）には, 色を格納するカラーバッファの他, 隠面消去処理に使うデプスバッファ, 他がある.
mask に GL_COLOR_BUFFER_BIT を指定したときは, カラーバッファだけが塗りつぶされる.

注 6　void glColor3d(double r, double g, double b)

描画する対象の色を RGB(Red, Green, Blue 成分)で指定する．引数は double 型で、0.0 が最も暗く、1.0 が最も明るい．　同じ機能で、glColor3f(float r, float g, float b)を使うと、引数は float 型の変数または定数を使う.

注 7　void glBegin(GLnum mode), void glEnd(void)

glBegin の引数には図形のタイプを指定する．前々ページ参照

注 8　void glVertex2d(double x, double y)

glVertex2d() は２次元の座標値を設定するのに使う. 引数の型は double (GLdouble). 引数が float 型のときは glVertex2f(), int 型のときは glVertex2i() を使う.

注 9　glFlush(void)

まだ実行されていない OpenGL の命令を全部実行する.
OpenGL は関数呼び出しによって生成される OpenGL の命令をその都度実行するのではなく, いくつか溜め込んでおいてまとめて実行する.

注 10　void glClearColor(float R, float G, float B, float A)

glClear(GL_COLOR_BUFFER_BIT) でウィンドウを塗りつぶす際の色を指定. R, G, B はそれぞれ赤, 緑, 青色の成分の強さを示す float 型(GLclampf 型)の値で, 0.0～1.0 の間の値を持つ. 1.0 が最も明るく, この３つに (0., 0., 0.) を指定すれば黒色, (1., 1., 1.) を指定すれば白色になる. 最後の A はα値と呼ばれ, OpenGL では不透明度として扱われる (0. で透明, 1. で不透明).

```
// t5.c        ビューポートの設定
#define   R1 200.0
#define   R2 173.2
#define   R3 100.
#include <GL/glut.h>

void display(void)
{   glClear(GL_COLOR_BUFFER_BIT);
   glBegin(GL_TRIANGLES);
      glColor3d(1.0, 0.0, 0.0);   glVertex2d(0., R1);
      glColor3d(1.0, 1.0, 0.0);   glVertex2d(-R2, R3);
      glColor3d(1.0, 1.0, 1.0);   glVertex2d( 0., 0.);

      glColor3d(1.0, 1.0, 0.0);   glVertex2d(-R2, R3);
      glColor3d(0.0, 1.0, 0.0);   glVertex2d(-R2,-R3);
      glColor3d(1.0, 1.0, 1.0);   glVertex2d( 0., 0.);

      glColor3d(0.0, 1.0, 0.0);   glVertex2d(-R2,-R3);
      glColor3d(0.0, 1.0, 1.0);   glVertex2d(0.,-R1);
      glColor3d(1.0, 1.0, 1.0);   glVertex2d( 0., 0.);

      glColor3d(0.0, 1.0, 1.0);   glVertex2d(0.,-R1);
      glColor3d(0.0, 0.0, 1.0);   glVertex2d(R2,-R3);
      glColor3d(1.0, 1.0, 1.0);   glVertex2d( 0., 0.);

      glColor3d(0.0, 0.0, 1.0);   glVertex2d(R2,-R3);
      glColor3d(1.0, 0.0, 1.0);   glVertex2d(R2, R3);
      glColor3d(1.0, 1.0, 1.0);   glVertex2d( 0., 0.);

      glColor3d(1.0, 0.0, 1.0);   glVertex2d(R2, R3);
      glColor3d(1.0, 0.0, 0.0);   glVertex2d(0., R1);
      glColor3d(1.0, 1.0, 1.0);   glVertex2d( 0., 0.);
   glEnd( );
   glFlush( );
}

void reshape(int w, int h)    // 再描画時に実行される
{   glViewport(0, 0, w, h);              // 注 4
   glLoadIdentity( );                    // 注 5
   glOrtho(-w, w, -h, h, -1.0, 1.0);     // 注 6
}

void init(void)
{   glClearColor(1.0, 1.0, 1.0, 0.0);   }

int main(int argc, char *argv[])
{   glutInitWindowPosition(10, 10);      // 注 1
   glutInitWindowSize(320, 240);         // 注 2
   glutInit(&argc, argv);
   glutInitDisplayMode(GLUT_RGBA);
   glutCreateWindow(argv[0]);
   glutDisplayFunc(display);
   glutReshapeFunc(reshape);             // 注 3
   init( );
   glutMainLoop( );
   return 0;
}
```

http://www.center.wakayama-u.ac.jp/~tokoi/opengl/libglut.html から解説引用

注 1　void glutInitWindowPosition(int x, int y)

新たに開くウィンドウの位置を指定. 指定しないときは, ウィンドウマネージャによって位置を決定. X Window の場合, -geometry オプションによってコマンドラインからウィンドウを開く位置やサイズを指定

注 2　void glutInitWindowSize(int w, int h)

新たに開くウィンドウの幅と高さを指定.指定しないときは, 300×300 のウィンドウを開く.

注 3　glutReshapeFunc(void (*func)(int w, int h))

引数 func には, ウィンドウが再描画されたときに実行する関数のポインタを与える. この関数の引数には再描画後のウィンドウの幅と高さが、システムによって渡される. reshape() の処理によって, プログラムは glViewport() で指定した領域に glOrtho() で指定した領域内の図形を表示するようになる.

ここで glOrtho() で指定するの領域の大きさをビューポートの大きさに比例するように設定すれば, 表示内容の大きさをビューポートの大きさにかかわらず一定に保つことができる. 逆に、この引数を定数にすると（例えば、glOrtho(-320., 320., -240., 240., -1., 1.);)、ウィンドウの大きさを変えたときに、内部の図形も対応して伸縮する。

注 4　void glViewport(int x, int y, int w, int h)

ビューポートを設定. ビューポートとは, 開いたウィンドウの中で, 実際に描画が行われる領域のこと. 正規化デバイス座標系の 2 点 (-1, -1), (1, 1) を結ぶ線分を対角線とする矩形領域がここに表示される. 最初の 2 つの引数 x, y にはその領域の左下隅の位置, w には幅, h には高さをデバイス座標系, すなわちディスプレイ上の画素数で指定する. 関数 reshape() の引数 w, h にはそれぞれウィンドウの幅と高さが入るから, glViewport(0, 0, w, h) は再描画後のウィンドウの全面を表示領域に使うことになる.

注 5　void glLoadIdentity(void)

これは変換行列を初期化する. 座標変換の合成は行列の積で表されるから, 変換行列には初期値として単位行列を設定しておく.

注 6　void glOrtho(double l, double r, double b, double t, double n, double f)

glOrtho() はワールド座標系を正規化デバイス座標系に平行投影 (orthographic projection : 正射影) する行列を変換行列に乗じる. 引数には左から, l に表示領域の左端 (left) の位置, r に右端 (right) の位置, b に下端 (bottom) の位置, t に上端 (top) の位置, n に前方面 (near) の位置, f に後方面 (far) の位置を指定する. この関数は, ビューポートに表示される空間の座標軸を設定する.