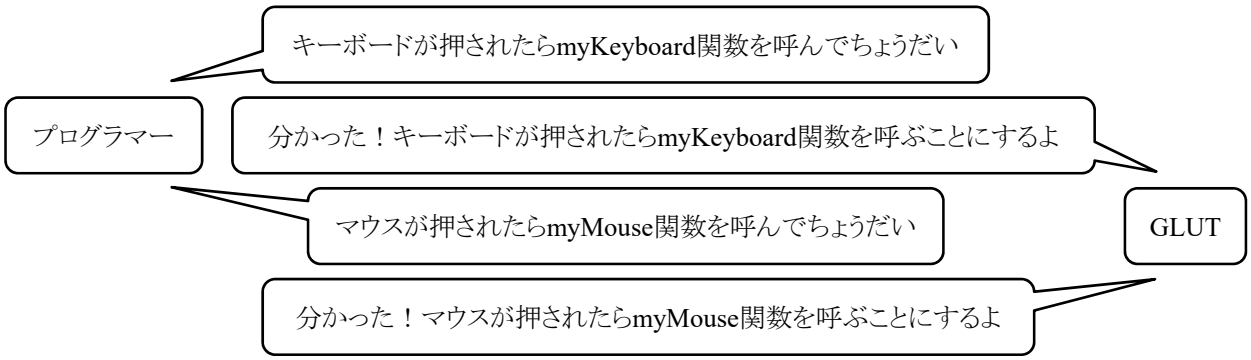
OpenGL

- typedef unsigned int GLenum
- typedef unsigned int GLbitfield
- typedef float GLclampf
- typedef double GLdouble
- void glutInit(int *argcp, char **argv)
  - GLUTライブラリを初期化します.
  - 「argc」と「argv」はmain関数の引数，すなわちコマンドライン引数を渡します．これらの引数は，コマンドラインのオプション指定時に用いられます.
- void glutInitDisplayMode(unsigned int mode)
  - ディスプレイの表示モードを設定します.
  - 「glutInitDisplayMode(GLUT_RGBA|GLUT_DOUBLE)」のように書くと，「RGBAカラーモデル」で「ダブルバッファ」を使うという指定になります.
- void glutInitWindowSize(int width, int height)
  - ウィンドウの初期サイズを設定します.
  - 「width」はウィンドウの幅，「height」はウィンドウの高さになります.
- void glutInitWindowPosition(int x, int y)
  - ウィンドウの左上の位置を指定する．引数は共にピクセル値.
- int glutCreateWindow(char *title)
  - ウィンドウを生成する．引数はそのウィンドウの名前となる.
- void glClearColor(GLclampf red, GLclampf green, GLclampf blue, GLclampf alpha)
  - 「glClear(GL_COLOR_BUFFER_BIT)」でウィンドウを塗りつぶす際の色を指定します.
  - 「red」「green」「blue」はそれぞれ「赤」「緑」「青色」の成分の強さを示すGLclampf型（float型と等価）の値で，0～1の間の値をもちます．1が最も明るく，この3つに(0,0,0)を指定すれば「黒色」になり，(1,1,1)を指定すれば「白色」になります.
  - 最後の「alpha」は「α値」と呼ばれ，OpenGLでは不透明度として扱われます（0で透明，1で不透明）．ここではとりあえず「1」にしておいてください.
- void glutMainLoop(void)
  - GLUTのイベントが発生するまで，待機状態になります.
- void glutSwapBuffers(void)
  - 描画の最後で記述する．この関数が実行されると，バックバッファの内容がフロントバッファに転送される.
- void glClear(GLbitfield mask)
  - 「mask」に指定したバッファのビットを初期化します.
  - 「glClear(GL_COLOR_BUFFER_BIT)」と指定すると「カラーバッファ」が初期化されます.
- void glutDisplayFunc(void (*)(void))
  - 引数は開いたウィンドウ内に描画する関数へのポインタです．ウィンドウが開かれたり，他のウィンドウによって隠されたウィンドウが再び現われたりして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ます．したがって，この関数内で図形表示を行います.
- void glutKeyboardFunc(void (*)(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．この関数の引数「key」には，タイプされたキーのASCIIコードが渡されます．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.
- void glBegin(GLenum mode)
  void glEnd(void)
  - 「glBegin」と「glEnd」の間に指定した頂点座標を使って，描画を行います.
  - 描画内容は「mode」に指定します．「mode」には「GL_POLYGON」などが指定できます.
- void glVertex2d(GLdouble x, GLdouble y)
  - 2次元の座標値を設定します.
  - 引数は，GLdouble型の(x, y) で指定します.
- void glColor3d(GLdouble red, GLdouble green, GLdouble blue)
  - これから描画するものの色を指定します.
  - 引数の型はGLdouble型で，「red」「green」「blue」にはそれぞれ「赤」「緑」「青」の強さを「0～1」の範囲で指定します.

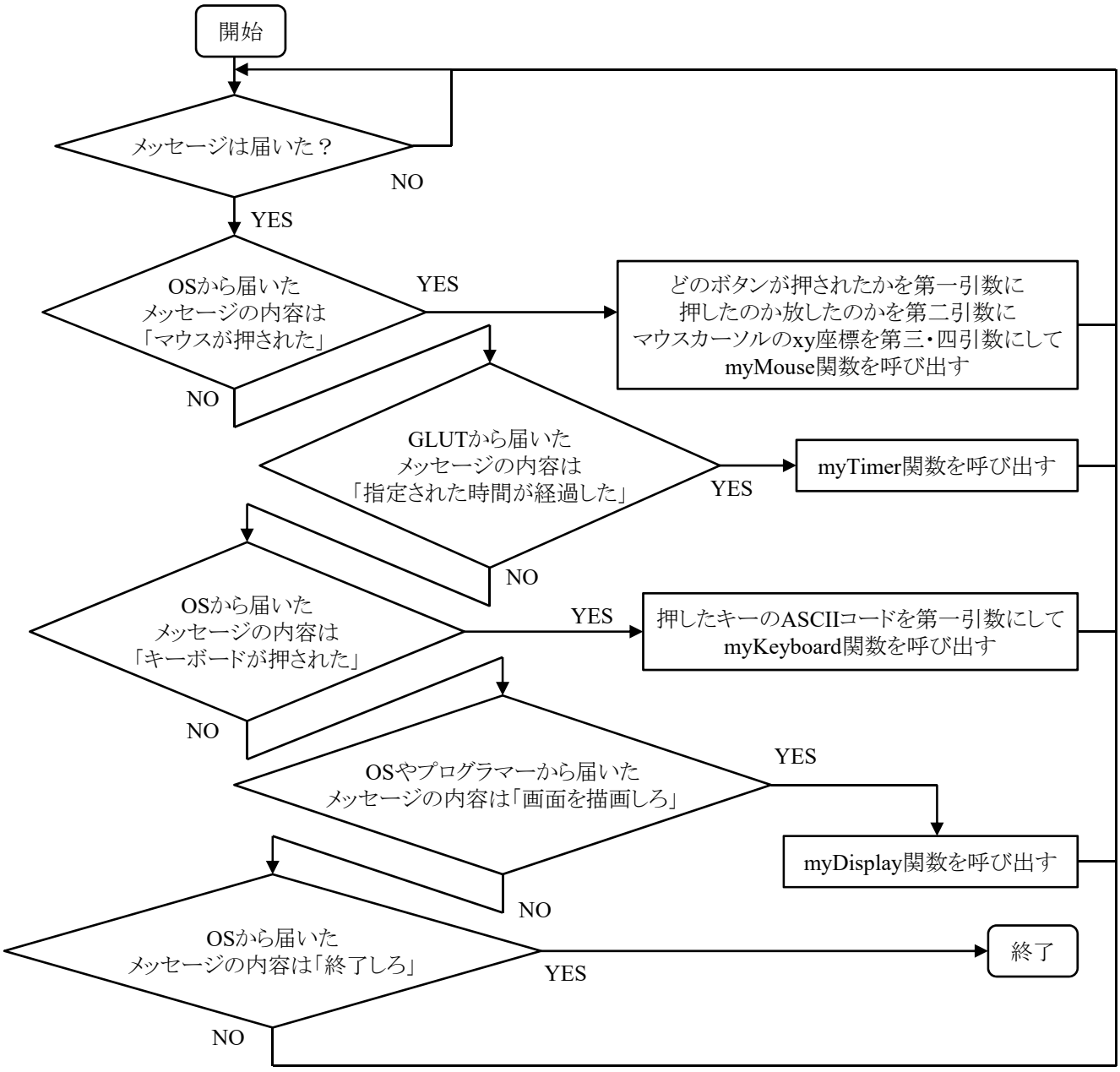
イベントドリブン型プログラミング

- glutMainLoopは無限ループしているだけで，メッセージが届くのをひたすら待ち続けます
- メッセージを受け取ったら，メッセージに応じた処理をして，また再び無限に待機します

コールバック関数の指定



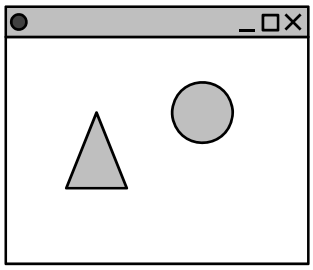
ループ中の動作の例



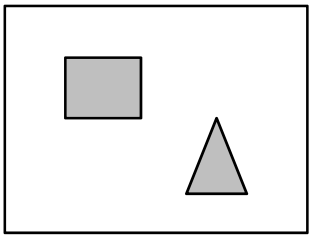
画面の描画

- glClearで画面を消去してglutSwapBuffersで描画します

描画の流れ

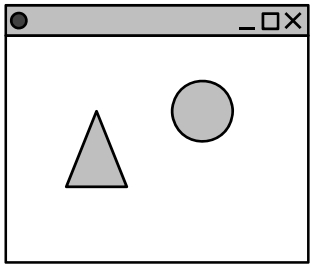


ウィンドウ表示用のバッファ



メモリ内にあるバッファ

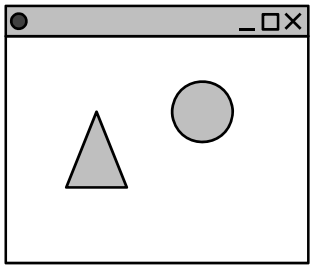
glClear(GL_COLOR_BUFFER_BIT);
画面消去



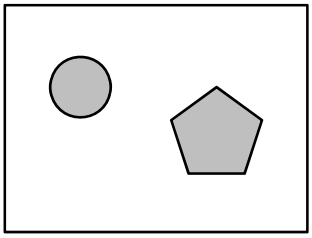
ウィンドウ表示用のバッファ

メモリ内にあるバッファ

glBegin～glEnd
図形の描画

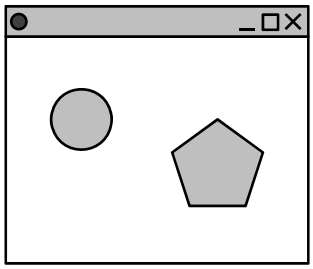


ウィンドウ表示用のバッファ

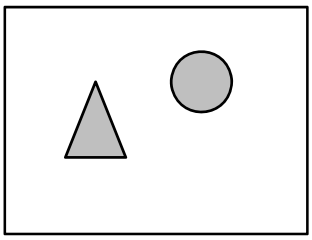


メモリ内にあるバッファ

glutSwapBuffers();
ウィンドウに表示



ウィンドウ表示用のバッファ



メモリ内にあるバッファ

```
void ディスプレイコールバック関数()
{
 // 画面消去
 glClear(GL_COLOR_BUFFER_BIT);

 // 図形の描画
 glBegin(GL_QUADS);
 glVertex3d(-1.0, -1.0, 0.0);
 glVertex3d(1.0, -1.0, 0.0);
 glVertex3d(1.0, 1.0, 0.0);
 glVertex3d(-1.0, 1.0, 0.0);
 glEnd();

 // ウィンドウに表示
 glutSwapBuffers();
}
```

点，線，ポリゴンの描画

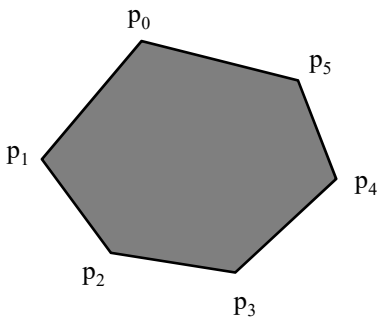
- 点，線，ポリゴンを描くのに，次のようにglBegin()とglEnd()および，glVertex*()を用いる．
- glBegin(mode);
  glVertex*(p0);
  glVertex*(p1);
  .....
  glVertex*(pn);
  glEnd();
- modeは描く図形の種類を指定し，p0，p1，…，pnは座標位置を意味する．

modeの種類

- GL_POLYGON　　単独の凸ポリゴンを描画する．

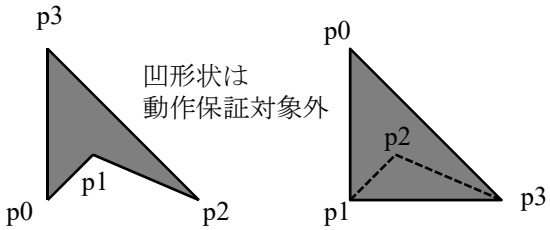
ポリゴン描写の注意点

- ポリゴン（polygon）とは多角形の意味



GL_POLYGON

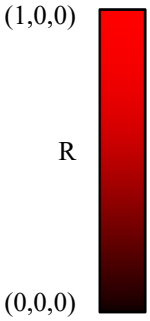
四角形の頂点の順番

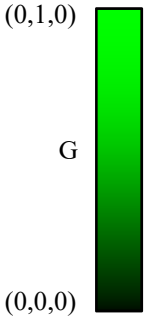


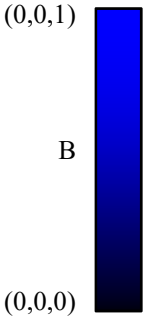
- 通常，凹多角形は三角形に分割して，凸多角形で表現するのが普通（検索キーワード：三角形分割）

色の表現

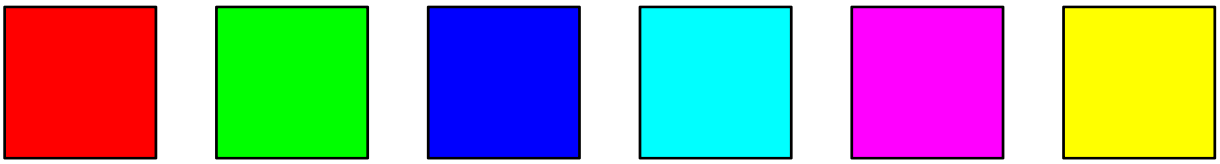
- 光の三原色で表す
- R: Red 赤, G: Green 緑, B: Blue 青
- glClearColor関数で背景色を指定，glColor3d関数で描画する図形の色を指定します
- OpenGLのこれらの関数の引数は，浮動小数点の0～1の値で表します
  - ちなみに，画面や画像ファイルなどの画素は通常，RGBそれぞれ8bitずつ，合計24bitで表します．整数で0～255の値で表します．OpenGLは0～1ですのでご注意ください．
- 0が暗くて，1が明るいです







(R,G,B)=(0,0,0)　　(R,G,B)=(0.5,0.5,0.5)　　(R,G,B)=(1,1,1)



(1,0,0)　(0,1,0)　(0,0,1)　(0,1,1)　(1,0,1)　(1,1,0)

(1,0,0)　(1,0.5,0.5)　(1,1,1)　　(0,1,0)　(0.5,1,0.5)　(1,1,1)　　(0,0,1)　(0.5,0.5,1)　(1,1,1)

(1,0,0)　(1,1,0)　(0,1,0)　(0,1,1)　(0,0,1)　(1,0,1)　(1,0,0)

図形の色

glColor3d関数

- 図形の色はglColor3d関数で指定します.

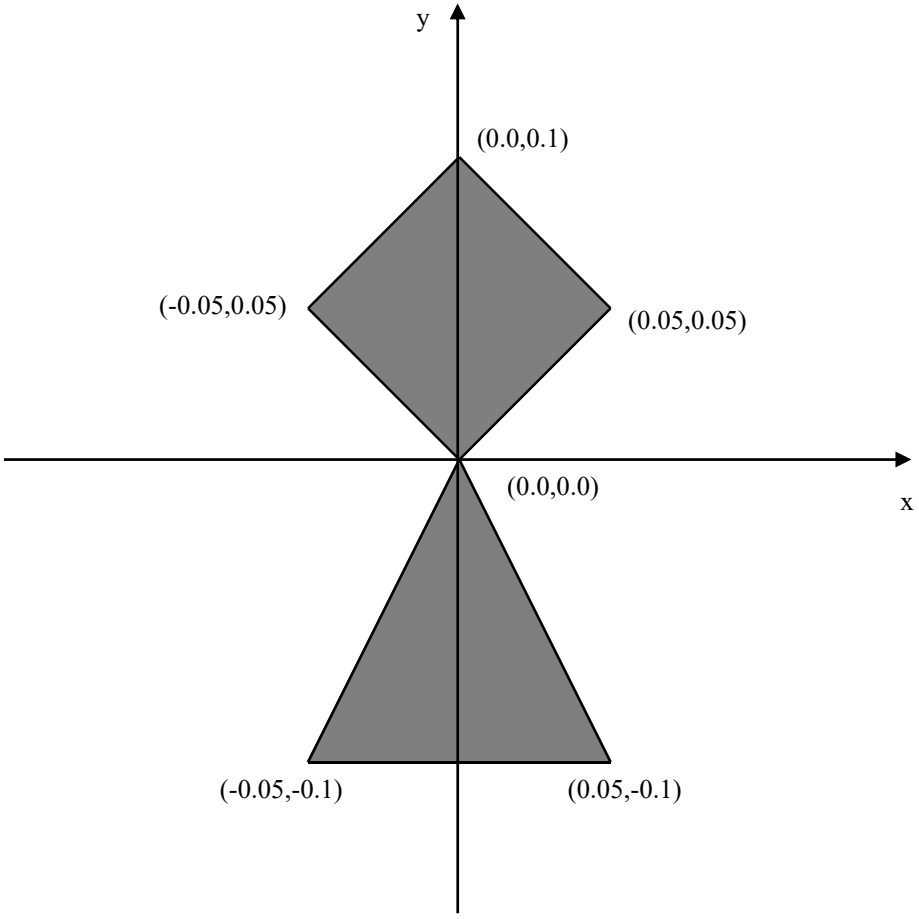
例
```
glColor3d(1.0, 0.0, 0.0);
glBegin(GL_QUADS);
glVertex3d(-1.0, -1.0, 0.0);
glVertex3d(1.0, -1.0, 0.0);
glVertex3d(1.0, 1.0, 0.0);
glVertex3d(-1.0, 1.0, 0.0);
glEnd();
```

お絵かき

- ポリゴンで好きな図形を描くプログラムを作る

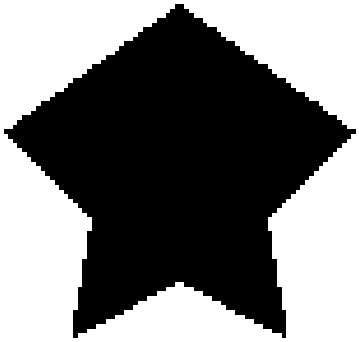
方眼紙

- 方眼紙に絵を描く
- -1～1の正方形を描く
- 真ん中が原点として，x軸は右向き，y軸は上向きとして，図形の頂点の座標を調べる
- その図形を描画するためのOpenGLのコマンドを入力する（凸多角形として描画する）



間違いの例

```
#include <stdlib.h>
#include <stdio.h>
#define _USE_MATH_DEFINES
#include <math.h>
#include <GL/glut.h>

void myObject()
{
    int i;
    double r;
    double a;
    double px, py;
    glBegin(GL_POLYGON);
    //  glBegin(GL_LINE_LOOP);
    for (i = 0; i < 5; i++) {
        r = 0.2;
        a = 2.0 * M_PI * (double)(i * 2) / 10.0 + M_PI / 2.0;
        px = r * cos(a);
        py = r * sin(a);
        glVertex2d(px, py);
        r = 0.1;
        a = 2.0 * M_PI * (double)(i * 2 + 1) / 10.0 + M_PI / 2.0;
        px = r * cos(a);
        py = r * sin(a);
        glVertex2d(px, py);
    }
    glEnd();
}

void myDisplay()
{
    glClearColor(1.0, 1.0, 1.0, 1.0);
    glClear(GL_COLOR_BUFFER_BIT);
    glColor3d(0.0, 0.0, 0.0);
    myObject();
    glutSwapBuffers();
}

void myKeyboard(unsigned char key, int x, int y)
{
    if (key == 0x1B) exit(0);
}

void myInit(char* progname)
{
    glutInitDisplayMode(GLUT_RGBA | GLUT_DOUBLE);
    glutInitWindowSize(400, 400);
    glutInitWindowPosition(0, 0);
    glutCreateWindow(progname);
}

int main(int argc, char* argv[])
{
    glutInit(&argc, argv);
    myInit(argv[0]);
    glutKeyboardFunc(myKeyboard);
    glutDisplayFunc(myDisplay);
    glutMainLoop();
    return 0;
}
```



正解の例

```
#include <stdlib.h>
#include <stdio.h>
#define _USE_MATH_DEFINES
#include <math.h>
#include <GL/glut.h>

void myStar()
{
    int i;
    double r;
    double a;
    double px, py;
    for (i = 0; i < 5; i++) {
        glBegin(GL_POLYGON);
        glVertex2d(0.0, 0.0);
        r = 0.2;
        a = 2.0 * M_PI * (double)(i * 2) / 10.0 + M_PI / 2.0;
        px = r * cos(a);
        py = r * sin(a);
        glVertex2d(px, py);
        r = 0.1;
        a = 2.0 * M_PI * (double)(i * 2 + 1) / 10.0 + M_PI / 2.0;
        px = r * cos(a);
        py = r * sin(a);
        glVertex2d(px, py);
        glEnd();
        glBegin(GL_POLYGON);
        glVertex2d(0.0, 0.0);
        r = 0.1;
        a = 2.0 * M_PI * (double)(i * 2 + 1) / 10.0 + M_PI / 2.0;
        px = r * cos(a);
        py = r * sin(a);
        glVertex2d(px, py);
        r = 0.2;
        a = 2.0 * M_PI * (double)(i * 2 + 2) / 10.0 + M_PI / 2.0;
        px = r * cos(a);
        py = r * sin(a);
        glVertex2d(px, py);
        glEnd();
    }
}

void myDisplay()
{
    glClearColor(1.0, 1.0, 1.0, 1.0);
    glClear(GL_COLOR_BUFFER_BIT);
    glColor3d(0.0, 0.0, 0.0);
    myStar();
    glutSwapBuffers();
}

void myKeyboard(unsigned char key, int x, int y)
{
    if (key == 0x1B) exit(0);
}

void myInit(char* progname)
{
    glutInitDisplayMode(GLUT_RGBA | GLUT_DOUBLE);
    glutInitWindowSize(400, 400);
    glutInitWindowPosition(0, 0);
    glutCreateWindow(progname);
}

int main(int argc, char* argv[])
{
    glutInit(&argc, argv);
    myInit(argv[0]);
    glutKeyboardFunc(myKeyboard);
    glutDisplayFunc(myDisplay);
    glutMainLoop();
    return 0;
}
```