GLUTの動作と座標系

山本昌志*

2007年11月21日

概 要

イベント駆動型プログラミングの概要とGLUTの座標系の取り扱い方を説明する．

1 本日の学習内容

先週に引き続き，コンピューターグラフィックスの学習を行う．本日のゴールは，以下の通りである．

- イベント駆動型プログラムの動作方法の概要が分かる．
- コンピューターグラフィックスの座標系の扱い方が分かる．

2 イベント駆動型プログラム

2.1 従来のプログラムとの違い

これまで学習してきたプログラムは，プログラムに記述されたとおりに実行された．通常はプログラムは上から下へ実行されれるが，`if`や繰り返し文があるとそれに従う．プログラムは，フロー (flow:流れ) を記述することになるので，フロー駆動型プログラムと呼ばれる．

それに対して，イベント (event:出来事) に対して，実行される内容が変化するプログラムをイベント駆動型プログラムと呼ぶ．例えば，マウスを動かすと，それに反応するゲームプログラムなどである．このようなプログラムでは，フローの代わりにイベントに応じた動作を記述する．プログラムに必要なことは，イベントの監視とどの動作内容である．このようなイベントの監視と動作内容をあわせて，イベントハンドラと呼ぶことがある．

ここで，学習しているGLUTを使ったプログラムは，イベント駆動型のプログラムである．作成したウインドウをクリックして他のウインドウの上にしたり，または移動させたりすると自動的に再描画してくれる．クリックとか移動とかのイベントに対応して，プログラムが実行されている．

*独立行政法人　秋田工業高等専門学校　電気情報工学科

2.2 簡単な仕組み

イベント駆動型プログラムは，さまざまなイベント[1]に対するプログラムの反応を記述する．このようなプログラムでは，次々に生じるイベントをどのようにして処理するのか?—というような疑問が生じるであろう．

イベントはイベントキュー (event queue) に入れられて，古いイベントから順に取り出される．取り出されたイベントは，それに対応する処理を行う．プログラム中では，イベントに対応する処理は，コールバック関数 (callback function) を使って記述する．

プログラムは，イベントに対する処理内容コールバック関数で記述した後，イベントループと呼ばれる無限ループに動作が移る．この無限ループで，イベントを監視の監視と，それに対応する処理を永遠行うことになる．先週学習したリスト 1 の例では，

- 16 行目により `draw` をコールバック関数に指定している．この場合，クリックにより，ウィンドウを前面に移動させて再描画が必要なときに，`draw` 関数が呼び出される．
- 18 行目の `glutMainLoop()` がイベントループである．これは無限ループでイベントの監視とコールバック関数の実行を司る．

である．

リスト 1: 赤で塗りつぶしたウィンドウを作成するプログラム．16 行目の `draw` がコールバック関数，18 行目の `glutMainLoop()` がイベントループである．

```
1  #include <stdio.h>
2  #include <GL/glut.h>
3
4  void draw(void);                                  // プロトタイプ宣言
5  void set_color(void);
6
7  //=============================================================
8  // main関数
9  //=============================================================
10 int main(int argc, char *argv[])
11 {
12
13   glutInit(&argc, argv);                          // GLUT 初期化
14   glutInitDisplayMode(GLUT_RGBA);                 // 表示モードの指定
15   glutCreateWindow("Yamamoto's window");          // windowをタイトルを付けてを開く
16   glutDisplayFunc(draw);                          // イベントにより呼び出し
17   set_color();                                    // 塗りつぶす色指定
18   glutMainLoop();                                 // GLUTの無限ループ
19
20   return 0;
21 }
22
23 //=============================================================
24 // ウィンドウを塗りつぶす
25 //=============================================================
26 void draw(void)
27 {
28   glClear(GL_COLOR_BUFFER_BIT);
```

[1]マウスの動作に関するマウスイベント，キーボードの動作に関するキーボードイベント，ウィンドウに関するウインドウイベントなど．

```
29    glFlush();                              // 描画
30  }
31
32  //=============================================================
33  // 色の指定
34  //=============================================================
35  void set_color(void)
36  {
37    glClearColor(1.0, 0.0, 0.0, 1.0);   //赤緑青と透明度
38  }
```

3 座標系

3.1 座標系について

コンピューターを使って，図を描く場合，以下に示す 2 つの座標系を区別しなくてはならない．

ワールド 座標系 (world coordinate)
: ユーザーが勝手に決めることができ，プログラムを記述するときの座標系である．プログラマーの好きな単位; [mm] でも [km] でも図形を描くことができる．オブジェクト座標系 (object coodinates) と呼ばれることもある．

ウィンドウ座標系 (window coordinate)
: ディスプレイ上の座標系で，実際に表示するときに使う．この座標系の単位は，ピクセル (pixel)[2]となる．

先週使ったプログラムでは，`glVertex2d(GLdouble x, GLdouble y);` を使って頂点の座標を指定した．このときの座標系がワールド座標系である．`GLdouble` は OpenGL の型で，通常の `double` と同じと考えて良い．

図をディスプレイ上に描く場合，ワールド座標系の作図範囲をしてしなくてはならない．その指定には，次の命令を使う．

void gluOrtho2D(GLdouble *left*, **GLdouble** *right*, **GLdouble** *buttom*, **GLdouble** *top***)**

: *left* と *right*，*buttom*，*top* は，ディスプレイに描くワールド座標系の四角形領域を指定している (図 1 参照)．

つぎに，ワールド座標系とディスプレイに表示されているウインドウとの位置関係を指定する必要がある．次の命令を使う．

void glViewport(GLint x, **GLint** y, **GLint** w, **GLint** h**)**
: x と y はウィンドウに描くワールド座標系の左下の位置，w と h はサイズを表している (図 1 参照)．いずれも，単位はピクセルである．

[2]ディスプレイ上の表示は小さい点の集まりである．その一つの点をピクセルと言う．

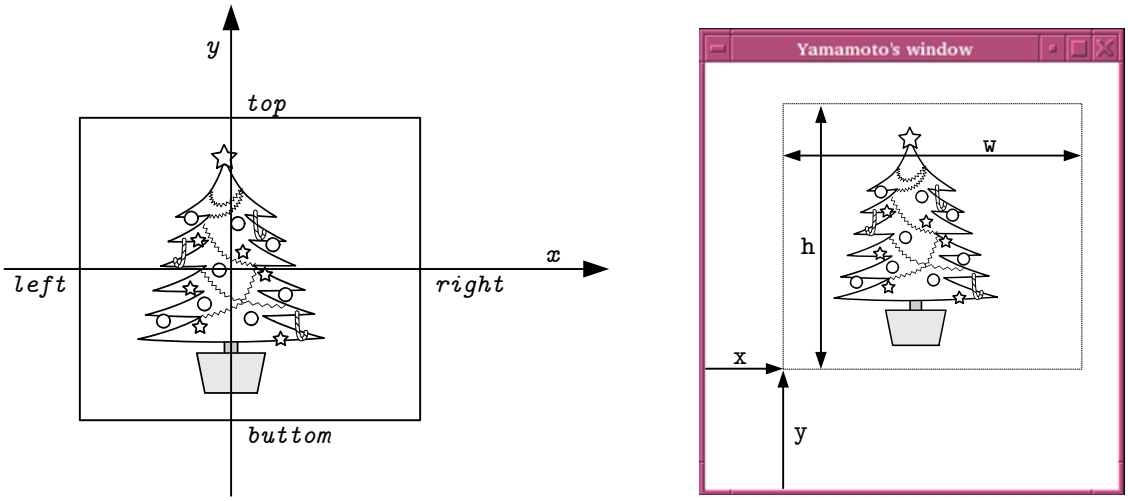

図 1: ワールド座標系とウィンドウ座標系の関係.

3.2 ウィンドウの初期値

ウィンドウを作成するときに，位置とサイズを指定することができる．そのためには，以下の命令を使う．

void glutInitWindowPosition(int x**, int** y**)**
`glCreatWindow()` で作成したウィンドウの初期位置を指定する．ウィンドウの左上の座標を指定する．この場合，ディスプレイの座標は左上が (0,0) で，右に行くに従い x 座標は大きくなり，下に行くに従い y 座標は大きくなる．単位は，ピクセル．

void glutInitWindowSize(int *width*, **int** *height*)
`glCreatWindow()` で作成したウィンドウの初期サイズを指定する．*width* が幅で，*height* が高さである．単位は，ピクセル．

4 座標系を指定したプログラムの例

4.1 先週のプログラムの問題点

先週のプログラムでは，最初は図 2 が描かれた．ウィンドウを変形すると，それに応じて図 3 のように描いた図形が変形した．これは，ワールド座標系とウィンドウ座標系の対応が悪いためである．

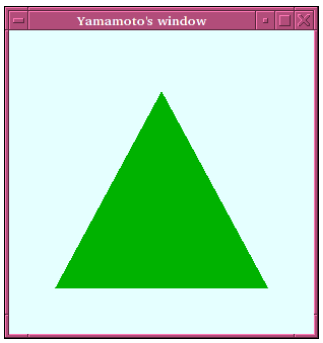

図 2: 元の図

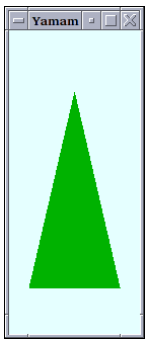

図 3: ウィンドウを縮める．図形の形が変化する．

4.2 サイズを変更しても図形が変化しない

ワールド座標系とウィンドウ座標系をきちんと指定し，ウィンドウのサイズを変化させても，形が変わらないプログラムを書く．

4.2.1 プログラム例

リスト 2 のようにすると，ウィンドウのサイズを変えても図形の形は変化しない．ウィンドウのサイズが変化するというイベントが発生すると，コールバック関数 `resize()` を呼び出すようにしている．イベントとコールバック関数の関係は，19 行目の `glutReshapeFunc(resize);` で決めている．このコールバック関数で，ワールド座標系とウィンドウ座標系を次のように決めている．

54 行目　　void resize(int w, int h)
コールバック関数が呼び出された時に，仮引数 `w` と `h` は，ウィンドウの幅と高さがピクセル単位で格納される．

60 行目　　glLoadIdentity()
変換行列の話．ちょっと難しいので，説明しない．最初のうちは，とりあえず，このように書く．

61 行目　　gluOrtho2D(-w/200.0, w/200.0, -h/200.0, h/200.0)
ワールド座標を決めている．次のウィンドウ座標系の指定とあわせて，ウィンドウ座標系の 1 ピクセルが，ワールド座標系の 0.01 に対応している．

62 行目　　glViewport(0, 0, w, h)
作成したウィンドウの全領域を使って作図するように指示している．

このプログラムでは，イベントが発生したときにコールバック関数が呼び出されたことが分かるようにしている．イベントに応じて，`draw()` 関数と `resize()` 関数が呼び出される．これらの関数が呼び出されると，ターミナルにメッセージが書かれる．このメッセージを見ると，コールバック関数が呼び出されるタイミングが分かる．

リスト 2: ウインドウサイズが変更しても，図形は変化しないプログラム例

```
1  #include <stdio.h>
2  #include <GL/glut.h>
3
4  void draw(void);                                  // プロトタイプ宣言
5  void resize(int w, int h);
6  void set_color(void);
7
8  //=============================================================
9  // main関数
10 //=============================================================
11 int main(int argc, char *argv[])
12 {
13   glutInitWindowPosition(100,200);                // 初期位置(x,y)指定
14   glutInitWindowSize(300,500);                    // 初期サイズ(幅, 高さ)指定
15   glutInit(&argc, argv);                          // GLUT 初期化
16   glutInitDisplayMode(GLUT_RGBA);                 // 表示モードの指定
17   glutCreateWindow("Yamamoto's window");          // windowをタイトルを付けてを開く
18   glutDisplayFunc(draw);                          // イベントにより呼び出し
19   glutReshapeFunc(resize);                        // サイズ変更のときに呼び出す関数指定
20   set_color();                                    // 塗りつぶす色指定
21   glutMainLoop();                                 // GLUTの無限ループ
22
23   return 0;
24 }
25
26 //=============================================================
27 // 図形を描く
28 //=============================================================
29 void draw(void)
30 {
31
32   printf("draw has been called.\n");    // イベント確認用(通常は不要)
33
34   glClear(GL_COLOR_BUFFER_BIT);
35
36   // ----- 三角形 ----------
37   glColor3d(0.0, 0.7, 0.0);                       // 線の色指定(RGB) 赤
38   glBegin(GL_TRIANGLES);                          // 開始 三角形
39   glVertex2d(-0.7, -0.7);                         // 頂点の指定
40   glVertex2d( 0.0,  0.6);
41   glVertex2d( 0.7, -0.7);
42   glEnd();                                        // 終了
43
44   glFlush();                                      // 描画
45 }
46
47 //=============================================================
48 // リサイズ
49 //      この関数は window のサイズが変化したら呼び出される
50 //      引数
51 //              w:ウィンドウの幅
52 //              h:ウィンドウの高さ
53 //=============================================================
54 void resize(int w, int h)
55 {
56
57   // イベント確認用(通常は不要)
58   printf("resize has been called. w=%d\th=%d\n", w, h);
59
60   glLoadIdentity();              // 変換行列を単位行列に
```

```
61    gluOrtho2D(-w/200.0, w/200.0, -h/200.0, h/200.0); // world座標系の範囲
62    glViewport(0, 0, w, h);          // ウィンドウ座標系を指定
63  }
64
65  //=============================================================
66  // 色の指定
67  //=============================================================
68  void set_color(void)
69  {
70    glClearColor(0.9, 1.0, 1.0, 1.0);   //赤緑青と透明度
71  }
```

実行結果

端末には以下のように表示され，

```
resize has been called. w=739   h=317
draw has been called.
resize has been called. w=729   h=317
draw has been called.
draw has been called.
draw has been called.
```

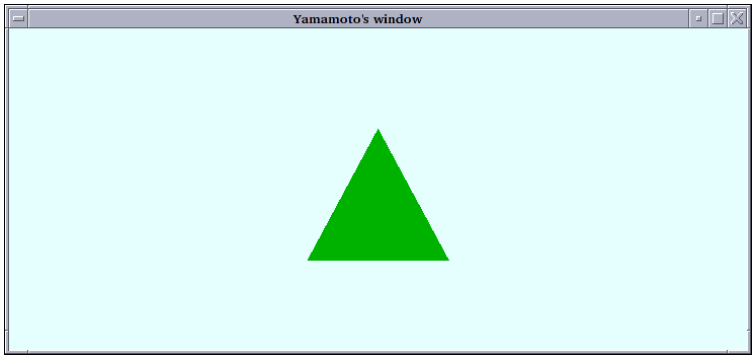

図 4: ウィンドウの枠を変化させても，中の図形は変わらない．

5 プログラム作成の練習

[**練習 1**] リスト 2 のプログラムを実行させて，イベント駆動型プログラムの動作を実感せよ．特に，コールバック関数とイベントループの動作を理解しなくてはならない．

[**練習 2**] リスト 2 のプログラムを変えて，以下を理解せよ．

- ワールド座標系とウィンドウ座標系の関係
- ウィンドウの初期値

[**練習 3**] 床井さんのページ図形のタイプ[3]を参考にして，いろいろな図形を作図せよ．

[3] http://www.wakayama-u.ac.jp/~tokoi/opengl/libglut.html#5.2

6　課題

以下の課題を実施し，レポートとして提出すること．

[**問 1**] (復) 美しいと思う図形を作図するプログラムを作成せよ．レポートには，プログラムのソースと図を提出すること．

提出要領はいつものとおり．ただし，期限は 11 月 28 日 (水)AM 8:45，課題名は「課題　GLUT の動作と座標系」とすること．