はじめてのGLUT

山本昌志*

2007年11月14日

概 要

GLUTを使った二次元グラフィックスについて学習する．

1 本日の学習内容

本日は，コンピューターグラフィックスの学習を行う．これについては教科書に記述はないので，プリントおよび関連する内容のWEBサイトを使って，プログラムの書き方を学ぶ．また，少しだけ，この内容と関わりのある，コマンドライン引数についても学習する．コマンドライン引数については，教科書 [1] のpp.136–137に記述がある．くわえて，makeの簡単な使い方も学習する．本日のゴールは，以下の通りである．

- コマンドライン引数の使い方が分かる．
- makeの簡単な使い方がわかる．
- GLUTをつかって，簡単な2次元グラフィックスが描ける．

2 コマンドライン引数

メイン関数に引数としてデータを与えるコマンドライン引数について説明する．

2.1 使い方

メイン関数に引数を与えたい場合がある．例えば，ファイル名を指定して，そのファイルに書かれているデータの処理をしたい場合などである．プログラム中にファイル名を書くと，処理するデータが変わる毎にコンパイルの作業が必要となり，実用的なプログラムではない．

メイン関数に引数としてデータを与えたい場合，コマンドライン引数を使う．その例のプログラムを，リスト1に示す．このプログラムは，実行ファイル名と引数をタイプして，実行させる．すると，メイン関数に引数という形でデータを渡すことができる．この引数をコマンドライン引数と呼ぶ．

本当は，先に示したようにファイル名を引数にとって，その処理をするプログラムを例にすべきであるが，ソースが長くなる．そのため，ここでは，引数の値を表示するプログラムにとどめている．

*独立行政法人　秋田工業高等専門学校　電気情報工学科

リスト 1: コマンドライン引数を使った例．実行時に main 関数に渡された引数を表示する

```
1  #include <stdio.h>
2
3  int main(int argc, char *argv[])
4  {
5
6    int i;
7
8    printf("argc = %d\n", argc);
9
10   for(i=0; i<argc; i++){
11     printf("argv[%d] = %s\n", i, argv[i]);
12   }
13
14   return 0;
15 }
```

実行例

```
./cl hoge fugafuga
```

実行結果

```
argc = 3
argv[0] = ./cl
argv[1] = hoge
argv[2] = fugafuga
```

このプログラムを見て分かるとおり，main() 関数の引数は，次のようになっている．

- 整数型の変数 argc には，実行時のコマンドと引数の個数が格納される．
- 文字型のポインターの配列 argv[] には，コマンドとその引数の各々が格納される．

2.2 書き方

これは，ワンパターンで次のように書く．

```
int main(int argc, char *argv[])
```

すると，実行時に main() に引数として，コマンドと引数の総数が argc に，コマンドと引数の文字列の先頭アドレスが配列 argv[] に格納される．先のリスト 1 の通りである．

ここで，仮引数の変数名 argc は *ARGument Count*，argv は *ARGument Vector* の略である．他の多くの言語でも，コマンドライン引数の変数名には argc と argv を使う．C 言語の場合，慣例として argc と argv を使うが，他の変数名でも問題はない．ただし，引数の型は変えてはならない．

ときどき，char *argv[] の代わりに，**argv と記述しているプログラム見る．これでも同じように動作する．なぜ，同じなの?—と言う疑問もあるだろう．説明すると，結構長くなるし，退屈する者も出てくるだろう．興味のある者は，各自考えよ．

2.3 メモリーの内容

コマンドライン引数の char *argv[] が分かりにくいであろう．これは，文字型のポインターの配列という意味である．argv[] は配列になっており，そこには文字型のポインター，すなわち文字列の先頭アドレスが格納される．先ほどの実行例では，図 1 のようにメモリーに引数の値が格納される．

ようするに，文字列を表す場合，その先頭アドレスが必要と言うことである．

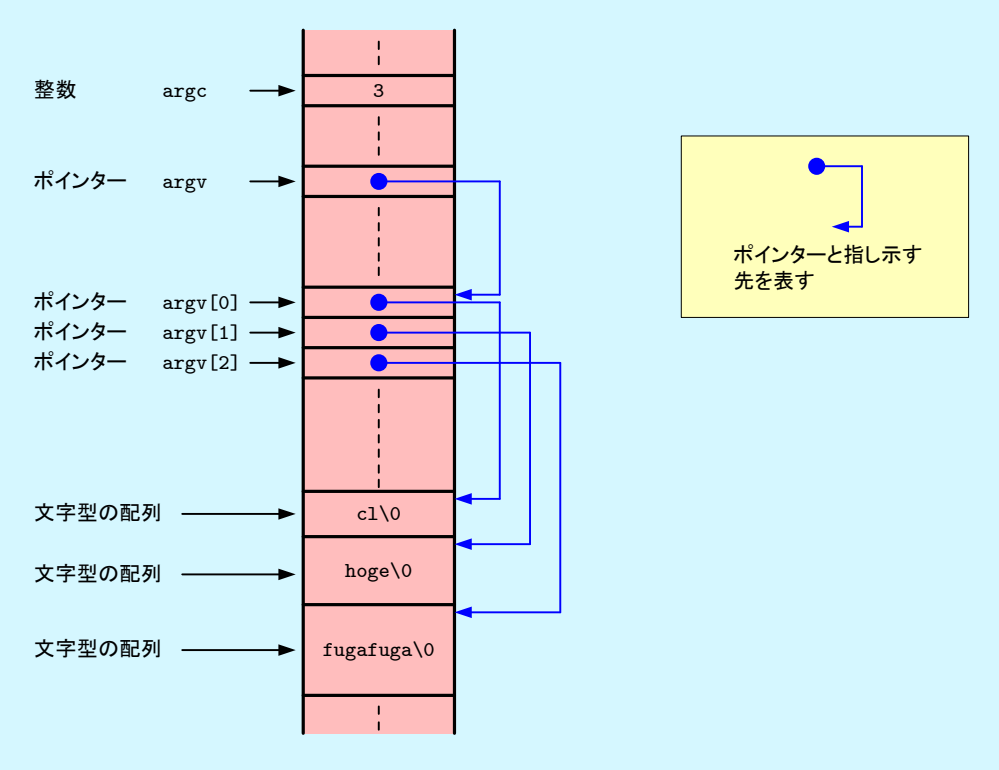

図 1: コマンドライン中のメモリーの様子.「./cl hoge fugafuga」と実行した場合.

3 GLUT を使ったグラフィックス

今までのプログラムの出力は，文字のみであった．ここでは，図の OpenGL を使ったコンピューターグラフィックスの基礎を学習する．

3.1 GLUT とは

GLUT を説明する前に，OpenGL を説明する必要があるだろう．OpenGL はコンピューターグラフィックスで有名なシリコングラフィックス社が開発したコンピューターグラフィックスの API(Application Program

Interface) である．プログラマーは OpenGL の関数をコールするだけで，簡単にコンピューターグラフィックスの作成ができる．

OpenGL は，Windows の DirectX と同じようなものである．DirectX は Windows でしか使えないが，OpenGL はどのような OS でも使える．Windows や Macintosh，Unix でも，同じように OpenGL を使って，コンピューターグラフィックスの作成が可能である．

GLUT は OpenGL のユーティリティツールキットで，OpenGL の初期化やウィンドウ操作，イベント処理などを簡単に記述することができる．OpenGL を実際に使うと，結構面倒な部分があり，それを簡単にしたものと思えば良い．

3.1.1 参考とすべき WEB ページと図書

GLUT を使ったプログラムは，和歌山大学の床井浩平先生の”GLUT による「手抜き」OpenGL 入門”[1]が大変参考になる．ここでの講義で分からないところがあれば，床井先生の WEB ページを見ると良い．

また，床井先生の著書「GLUT による OpenGL 入門」 [2] も分かりやすくて良い．GLUT を使う場合，まず最初にそろえるべき書籍である．ここでの講義もこの書籍を元に作成している．

3.2 二次元グラフィックス

いろいろ説明するよりも，実際のプログラムとその結果を見た方がわかりやすいであろう．図 2 が GLUT を使って，作成したウィンドウである．そのプログラムのソースをリスト 2 に示す．

リスト 2: 赤で塗りつぶしたウィンドウを作成するプログラム

```
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)
```

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

```
27 {
28   glClear(GL_COLOR_BUFFER_BIT);
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 }
```

実行結果　図 2 のウィンドウが作成される．

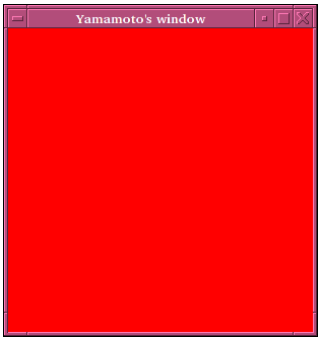

図 2: リスト 2 のプログラムの実行結果．赤く塗りつぶされたウィンドウが表示される．

3.2.1　どのように動作するか

GLUT を使ったプログラムは，イベント駆動型のプログラムになる．これまで学習してきたプログラムは，プログラムで書かれた順序通りに処理を行っていた．それに対して，イベント駆動型のプログラムは，イベント—キーボードやマウスの操作—によって，処理を行う．

リスト 2 の例では，作成した赤く塗りつぶされたウィンドウの再描画が必要なときに，`draw()` 関数が実行される．たとえば，他のウィンドウに隠れていて，それが前に出るときである．実際にその関数が実行されることを確かめることができる．リスト 2 の 27 行目と 28 行目の間に，

```
printf("function draw() has been called.\n");
```

を書き加える．そして，プログラムを実行させる．赤く塗りつぶされたウィンドウを操作すると，端末に

```
function draw() has been called.
function draw() has been called.
function draw() has been called.
function draw() has been called.
function draw() has been called.
```

と表示されるであろう．再描画が必要なときに，draw() 関数が呼び出されたことが分かるだろう．

リスト 2 のプログラムは，無限ループ glutMainLoop(); の中で，イベントが起きるのを待っており，そのイベントに応じた処理を自動的に行っている．これを今までのプログラミングスタイルで記述するとなると大変面倒なことになる．GLUT を使うと自動的に，イベントの処理を行う．プログラマーは，イベントと処理の内容を記述するだけでよい．さまざまな面倒な処理は，GLUT が自動的に行うのである．プログラマーは，楽ができる仕組みとなっている．

3.2.2 Makefile をつかおう

ところで，リスト 2(paint_out.c) をコンパイルして実行ファイルを作成するためには，次のようにコマンドを打ち込む必要がある [2].

```
gcc -o paint_out -I/usr/X11R6/include -Wall paint_out.c -L/usr/X11R6/lib \
-lglut -lGLU -lGL -lXmu -lXi -lXext -lX11 -lm
```

GLUT のライブラリーを使うため，その記述が必要である．

プログラムを直す毎にこの長いコマンドを打ち込むのはうんざりする．このようなときに，make を使う．make については，教科書 [1] の pp.34–36 に記述しているの詳細は述べない．また，いろいろな WEB ページにも解説があるので，それを見ると良いだろう．

私がリスト 2(paint_out.c) をコンパイルしたときの Makefile をリスト 3 に示す．ソースファイルと Makefile を同じディレクトリーに置いて,「make」と打ち込めば，実行ファイルができる.「make clean」とすれば，実行ファイルを削除できる．他のファイルをコンパイルしたい場合は，Makefile 中の SRC の横のファイル名を書き直せば良い．

Makefile の大体の部分は，見れば分かるであろう．注意すべきことは，行頭に [Tab] を打ち，スペースを空けずにコマンド—リスト 3 では gcc と rm—を書くことである．[Tab] の代わりにスペースで記述してはならない．想像がつくと思うが，#があると，そこから行末までコメント文になる．

リスト 3: Makefile の例.「make」で実行ファイルの作成ができる．また,「make clean」で実行ファイルの削除ができる．

```
1  CFLAGS = -I/usr/X11R6/include -Wall
2  LIBS = -L/usr/X11R6/lib -lglut -lGLU -lGL -lXmu -lXi -lXext -lX11 -lm
3  SRC = paint_out.c       # ここにソースファイル名を書くだけ
4
5  #--- 実行ファイルを作成するコマンド ----------
6  $(SRC:.c=):$(SRC)
7          gcc -o $(SRC:.c=) $(CFLAGS) $(SRC) $(LIBS)
8
9  #--- make clean で実行ファイルを削除できる ---------
10 clean:
11         rm $(SRC:.c=)
```

3.2.3 2 次元図形の例

二次元グラフの作図の例を示す．いろいろな二次元グラフの作図命令は，床井さんの”GLUT による「手抜き」OpenGL 入門”[2]の 2 次元図形を描く[3]を見よ．ここでは，2 つの例を示すにとどめる．

折れ線による描画 GL_LINE_STRIP を使うと，頂点 (vertex) を直線で結ぶ図形を書くことができる．それを使って，グルグルを描くプログラム例をリスト 4 に示す．これを見て分かるとおり，glBegin() と glEnd() の間に頂点を指定—glVertex() を使う—すれば，それらが直線で結ばれる．頂点の座標は，glVertex(x 座標,y 座標) と指定する．ウィンドウの左下が $(-1, -1)$ で右上が $(1, 1)$ である．

リスト 4: 折れ線をつかって，グルグルを描くプログラム．

```
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 // 折れ線でグルグルを描く
25 //=============================================================
26 void draw(void)
27 {
28   glClear(GL_COLOR_BUFFER_BIT);
29   glColor3d(0.0, 0.0, 1.0);                  // 線の色指定(RGB) 青
30   glBegin(GL_LINE_STRIP);                    // 開始 頂点を直線で結ぶ
31   glVertex2d(-0.9, -0.9);                    // 頂点の指定
32   glVertex2d(-0.9,  0.9);
33   glVertex2d( 0.9,  0.9);
34   glVertex2d( 0.9, -0.9);
35   glVertex2d(-0.6, -0.9);
36   glVertex2d(-0.6,  0.6);
37   glVertex2d( 0.6,  0.6);
38   glVertex2d( 0.6, -0.6);
39   glVertex2d(-0.3, -0.6);
40   glVertex2d(-0.3,  0.3);
41   glEnd();                                   // 終了
42   glFlush();                                 // 描画
43 }
```

[2] http://www.wakayama-u.ac.jp/~tokoi/opengl/libglut.html
[3] http://www.wakayama-u.ac.jp/~tokoi/opengl/libglut.html#5

```
44
45 //=============================================================
46 // 色の指定
47 //=============================================================
48 void set_color(void)
49 {
50   glClearColor(0.9, 1.0, 1.0, 1.0);   //赤緑青と透明度
51 }
```

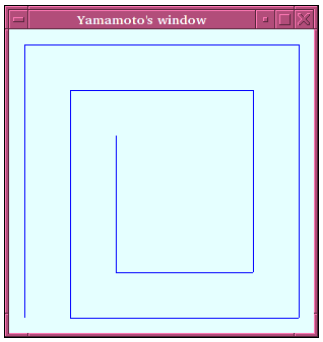

図 3: リスト 4 のプログラムの実行結果．折れ線を使って，ぐるぐるを描いている．

複数の図形の描画　描きたい図形の数だけ `glBegin()` と `glEnd()` を記述すれば，複数の図形を描くことができる．

リスト 5: 三角形と四角形を描くプログラム．

```
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 // 折れ線でグルグルを描く
25 //=============================================================
26 void draw(void)
```

```
27 {
28   glClear(GL_COLOR_BUFFER_BIT);
29
30   // ----- 三角形 ----------
31   glColor3d(0.0, 0.7, 0.0);              // 線の色指定(RGB) 赤
32   glBegin(GL_TRIANGLES);                 // 開始 三角形
33   glVertex2d(-0.7, -0.7);                // 頂点の指定
34   glVertex2d( 0.0,  0.6);
35   glVertex2d( 0.7, -0.7);
36   glEnd();                               // 終了
37
38   // ----- 四角形 ----------
39   glColor3d(1.0, 0.0, 0.0);              // 線の色指定(RGB) 赤
40   glBegin(GL_QUADS);                     // 開始 四角形
41   glVertex2d(-0.7,  0.5);                // 頂点の指定
42   glVertex2d(-0.7,  0.7);
43   glVertex2d(-0.5,  0.7);
44   glVertex2d(-0.5,  0.5);
45   glEnd();                               // 終了
46
47   glFlush();                             // 描画
48 }
49
50 //=============================================================
51 // 色の指定
52 //=============================================================
53 void set_color(void)
54 {
55   glClearColor(0.9, 1.0, 1.0, 1.0);   //赤緑青と透明度
56 }
```

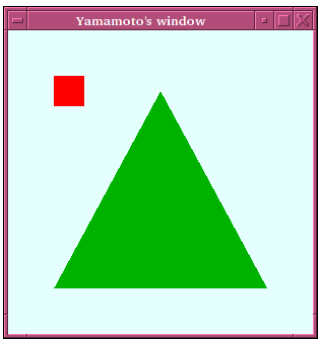

図 4: リスト 5 のプログラムの実行結果．三角形と四角形の複数の図形を描画している．

4 プログラム作成の練習

[**練習 1**] コマンドライン引数に関する練習問題である．キーボードから与えられた引数を逆順に表示するプログラムを作成せよ．

[**練習 2**] 床井さんのページ図形のタイプ[4]をみて，以下のタイプの図形を作図せよ．

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

- GL_POINTS：点を打つ．
- GL_LINES：2 点を対にして, その間を直線で結びます．
- GL_LINE_STRIP：折れ線を描く．
- GL_LINE_LOOP：折れ線で閉じた図形．
- GL_TRIANGLES：3 点を組にして, 三角形を描く．
- GL_QUADS：4 点を組にして, 四角形を描く．
- GL_TRIANGLE_STRIP：一辺を共有しながら帯状に三角形を描く．
- GL_QUAD_STRIP：一辺を共有しながら帯状に四角形を描く．
- GL_TRIANGLE_FAN：一点を共有しながら扇状に三角形を描く．
- GL_POLYGON：多角形を描く．

[練習 3] 3.2.1 節の「どのように動作するか」を確かめよ．

5 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] (復予)make の記述がある教科書 [?]pp.34–36 を 3 回読め．レポートには「3 回読んだ」と書け．

[問 2] (復予) コマンドライン引数の記述がある教科書 [?]pp.136–138 を 3 回読め．レポートには「3 回読んだ」と書け．

[問 3] (復) イベント駆動型プログラムとはどのようなものか?—調べよ．

[問 4] (復) ここでの学習内容でわからないところがあれば，具体的に記述せよ．

提出要領はいつものとおり．ただし，期限は 11 月 21 日 (水)AM 8:45，課題名は「課題　はじめての GLUT」とすること．

参考文献

[1] 内田智史監修, (株) システム計画研究所編. C 言語によるプログラミング　応用編　第 2 版. (株) オーム社, 2006.

[2] 床井浩平. GLUT による OpenGL 入門. (株) 工学社, 2006.