GLUTを利用したアニメーション

山本昌志*

2007年11月28日

概 要

glut を使って，アニメーションを描く方法を学ぶ．

1　本日の学習内容

先週の講義内容は，学生に受けが良かったので，もう少しおもしろいコンピューターグラフィックスを作ることにする．本日は，簡単なアニメーションを作成することを学ぶ．本日の学習のゴールは，以下の通り．

- コンピューターでのアニメーションの作成方法の概要が分かる．
- 簡単なアニメーションの作成ができる．

2　アニメーション

2.1　アニメーションの仕組み

時間とともに絵を移動させることにより，アニメーションを作成することができる．例えば，図1のように，プロペラの角度を少しずつ変化させると風車のアニメーションができあがる．

コンピューターでアニメーションを作成する場合，シーン毎の座標をデータとして指定すると大変面倒である．シーン毎に膨大なデータが必要となるからである．データを軽減するために，動きは計算により求めることが多い．

*独立行政法人　秋田工業高等専門学校　電気情報工学科

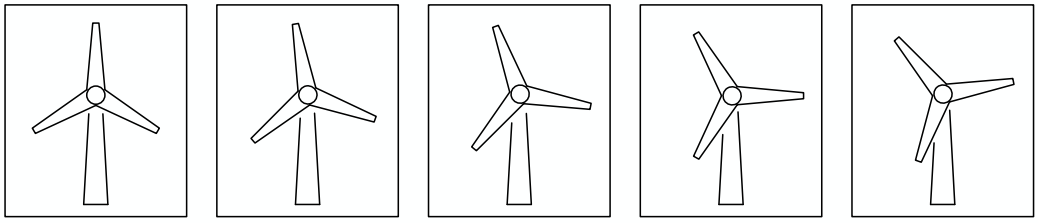

図 1: プロペラの角度が少しずつ回転する図を重ねるとアニメーションになる．

2.2 アニメーションのプログラム例

それでは，実際にアニメーションのプログラムを見てみよう．リスト 1 が簡単なアニメーションの例で，これを実行すると図 2 の風車のプロペラが回る．

アニメーションは，次のようにして描いている．

- 27 行目の `glutIdleFunc(anime);` により，アニメーションを描く関数 (anime()) をコールバック関数として指定している．これにより，PC が暇なとき; CPU の空き時間にアニメーションを描くコールバック関数を呼び出すことができる．CPU の空き時間毎に絵を書き直すことになる．
- 37–66 行目の `draw()` 関数は，風車のタワーとプロペラを描く．プロペラの角度は，グローバルカン変数 `theta` を使って決めている．この変数の値を変えると，プロペラの角度が変わる．
- 86–91 行目の `anime()` 関数が実行されると，新たに絵を描く．88 行目で風車のプロペラの角度 (`theta`) を 0.02[deg] ずつ増加させている．89 行目は，角度が 360[deg] を超えた場合の処理である．この行は無くても動作する．90 行目の `glutPostRedisplay()` 関数により，`glutDisplayFunc()` に登録したコールバック関数;`draw()` を呼び出す．これにより再描画ができる．

リスト 1: アニメーションのプログラム例．風車のプロペラが回る．

```
1  #include <stdio.h>
2  #include <GL/glut.h>
3  #include <math.h>
4
5  //---- プロトタイプ宣言 ----
6  void draw(void);                          // 図を描く
7  void resize(int w, int h);                // サイズの調整
8  void anime(void);                         // アニメーション
9  void set_color(void);                     // 塗りつぶし色の設定
10
11 //---- グローバル変数 ------
12 double theta = 0.0;                       // プロペラの角度, 初期値
13 double deg = M_PI/180;                    // 度をラジアンに
14
15 //============================================================
16 // main関数
```

```
17 //=============================================================
18 int main(int argc, char *argv[])
19 {
20   glutInitWindowPosition(100,200);          // 初期位置(x,y)指定
21   glutInitWindowSize(200,200);              // 初期サイズ(幅, 高さ)指定
22   glutInit(&argc, argv);                    // GLUT 初期化
23   glutInitDisplayMode(GLUT_RGBA | GLUT_DOUBLE);          // 表示モードの指定
24   glutCreateWindow("windmill");             // windowをタイトルを付けてを開く
25   glutDisplayFunc(draw);                    // イベントにより呼び出し
26   glutReshapeFunc(resize);                  // サイズ変更のときに呼び出す関数指定
27   glutIdleFunc(anime);                      // 暇なときに実行(アニメーション)
28   set_color();                              // 塗りつぶす色指定
29   glutMainLoop();                           // GLUTの無限ループ
30
31   return 0;
32 }
33
34 //=============================================================
35 // 図形を描く
36 //=============================================================
37 void draw(void)
38 {
39   double l1=0.7, l2=0.05;
40   double dthe;
41   int i;
42
43   glClear(GL_COLOR_BUFFER_BIT);
44   // ---- タワー ----------
45   glColor3d(0.5, 1.0, 0.5);                 // 線の色指定(RGB)
46   glBegin(GL_POLYGON);
47   glVertex2d(-0.1, -0.9);
48   glVertex2d(-0.05, 0.1);
49   glVertex2d( 0.05, 0.1);
50   glVertex2d( 0.1, -0.9);
51   glEnd();
52
53   // ---- プロペラ ----------
54   glColor3d(1.0, 1.0, 1.0);                 // 線の色指定(RGB)
55   for(i=0; i<3; i++){
56     glBegin(GL_POLYGON);
57     dthe=i*120;                             // プロペラの3軸の角度
58     glVertex2d(0.0, 0.0);
59     glVertex2d(l2*cos((theta+dthe-60)*deg), l2*sin((theta+dthe-60)*deg));
60     glVertex2d(l1*cos((theta+dthe)*deg),    l1*sin((theta+dthe)*deg));
61     glVertex2d(l2*cos((theta+dthe+60)*deg), l2*sin((theta+dthe+60)*deg));
62     glEnd();
63   }
64
65   glutSwapBuffers();                         // 描画
66 }
67
68
69 //=============================================================
70 // リサイズ
71 //      この関数は window のサイズが変化したら呼び出される
72 //      引数
73 //              w:ウィンドウの幅
74 //              h:ウィンドウの高さ
75 //=============================================================
76 void resize(int w, int h)
77 {
78   glLoadIdentity();              // 変換行列を単位行列に
```

```
79    gluOrtho2D(-w/200.0, w/200.0, -h/200.0, h/200.0); // world座標系の範囲
80    glViewport(0, 0, w, h);          // ウィンドウ座標系を指定
81  }
82
83  //=============================================================
84  // PCが暇なときに実行する. これが実行されるとアニメーションが描かれる
85  //=============================================================
86  void anime(void)
87  {
88    theta += 0.02;
89    if(theta>=360) theta-=360;                    // 360度を超えた処理
90    glutPostRedisplay();
91  }
92
93  //=============================================================
94  // 色の指定
95  //=============================================================
96  void set_color(void)
97  {
98    glClearColor(0.0, 0.0, 1.0, 1.0);          //赤緑青と透明度
99  }
```

実行結果

図 2 に示す風車が回る.

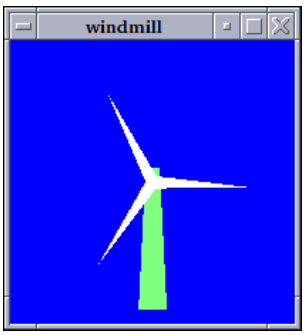

図 2: リスト 1 のプログラムの実行結果. 風車が回る.

2.3 プログラムの内容説明

2.3.1 各行の動作

復習もかねて, リスト 1 のプログラムの内容を説明しておく.

- **1–3 行目**　プログラムのコンパイルに必要なヘッダーファイルを読み込んでいる.
- **6–9 行目**　コンパイルの時に使われるプロトタイプ宣言. `set_color()` 関数を除いて, 後はコールバック関数である.

- **12 行目** `theta = 0.0;`　`theta` はプロペラの角度を決める変数．初期値は，ゼロとしている．関数 `draw()` で，この変数は使われ，プログラムの終了まで必要である．ローカル変数として宣言すると，以前の角度のデータが失われる．そのため，グローバル変数としている．ローカル変数で `static` 修飾子をつけてもよい．

- **13 行目** `deg = M_PI/180`　単位を [deg] から [rad] に変換するグローバル変数．ローカル変数で宣言しても良いが，呼び出す毎に計算するのは不経済．

- **20 行目** `glutInitWindowPosition(100,200);`　ウィンドウの初期位置を指定．ウィンドウの左上がディスプレイの左上から (200 ピクセル,200 ピクセル) の位置になる．

- **20 行目** `glutInitWindowSize(200,200);`　初期のウィンドウサイズを決めている．ここでは，200 ピクセル ×200 ピクセル．

- **21 行目** `glutInit(&argc, argv);`　GLUT の初期化．これはおまじないと考える．

- **23 行目** `glutInitDisplayMode(GLUT_RGBA | GLUT_DOUBLE);`　表示モードを決めている．ここでは，色が RGBA(赤，緑，青，透明度) でダブルバッファを指定している．アニメーションを描くときにはダブルバッファーとする．

- **24 行目** `glutCreateWindow("windmill");`　「windmill」というタイトルで新規にウィンドウを作成する．

- **25 行目** `glutDisplayFunc(draw);`　ウィンドウの内容を再描画するときに呼び出すコールバック関数を指定している．ここでは，ウィンドウの再描画必要なとき，関数 `draw()` を呼び出すことになる．

- **26 行目** `glutReshapeFunc(resize);`　ウィンドウのサイズの変更，または移動したときに呼び出すコールバック関数を指定している．ここでは，関数 `resize()` が呼び出される．

- **27 行目** `glutIdleFunc(anime);`　他に処理すべきイベントがないときに呼び出されるコールバック関数を指定している．ここでは，こんぴゅーたーが暇なときに，関数 `anime()` が呼び出される．

- **28 行目** `set_color();`　画面をクリアーするときの色指定の関数を呼び出している．

- **29 行目** `glutMainLoop();`　イベントループ; イベントを監視して，それに応じたコールバック関数を呼び出す無限ループを開始する．

- **37–66 行目** `void draw(void)`　ウィンドウ内に絵を描く関数．

- **76–81 行目** `void resize(int w, int h)`　ウィンドウのサイズの変更や移動が行われたときの処理を行う関数．

- **86–91 行目** `void anime(void)`　角度を変化させて，再度，絵を描く関数 `draw()` を呼び出している．

- **96–99 行目** `void set_color(void)`　画面をクリアーするときの色指定の関数．

2.3.2 新たな関数

アニメーションを描くために，新たに加わった関数は，以下の三つである．

void glutIdelFunc(vod (func*)void))**
: イベントキューにイベントが無いときに実行されるコールバック関数を登録する．すなわち，コンピューターが暇なときに実行される関数を登録するのである．引数は難しそうなことを書いているが，関数名で OK．関数名はポインターと成っている．

void glutPostRedisplay(vod)
: 現在のウィンドウの再描画が必要とマークする．次の機会に，`glutDisplayFunc()` で登録したコールバック関数を実行する．ここでは，`draw()` 関数が実行される．

void glutSwapBuffers()
: 裏バッファーと表バッファーを交換する．アニメーションを作成する場合は，2 枚のバッファーに交互に得を作成する．

2.3.3 プロペラの描き方

プロペラが回転しているように見せるためには，その絵を角度 θ を使って表現しなくてはならない．この角度を連続的に変えることにより，回転している絵を描く．

プロペラは，`GL_POLYGON` を使って描く．しかし，これは凸の図形しか描くことができないので，プロペラの羽を一つずつ描いてそれを足しあわせて一つのプロペラとする．

ひとつの羽は，図 3 のような座標系を用いる．角度 θ を用いて，図中の 4 つの座標を表現する必要がある．三角関数の知識を使うと，次のようになる．

$$(0,\quad 0) \qquad \text{座標 (1)} \tag{1}$$

$$(\ell_2 \cos(\theta - \pi/3),\quad \ell_2 \sin(\theta - \pi/3)) \qquad \text{座標 (2)} \tag{2}$$

$$(\ell_1 \cos(\theta),\quad \ell_1 \sin(\theta)) \qquad \text{座標 (3)} \tag{3}$$

$$(\ell_2 \cos(\theta + \pi/3),\quad \ell_2 \sin(\theta + \pi/3)) \qquad \text{座標 (4)} \tag{4}$$

これを，`glVertex2d` で決めれば一枚の羽が描ける．他の 2 枚の羽は，これに $2\pi/3$[rad] と $4\pi/3$[rad] の角度を加えれば良い．

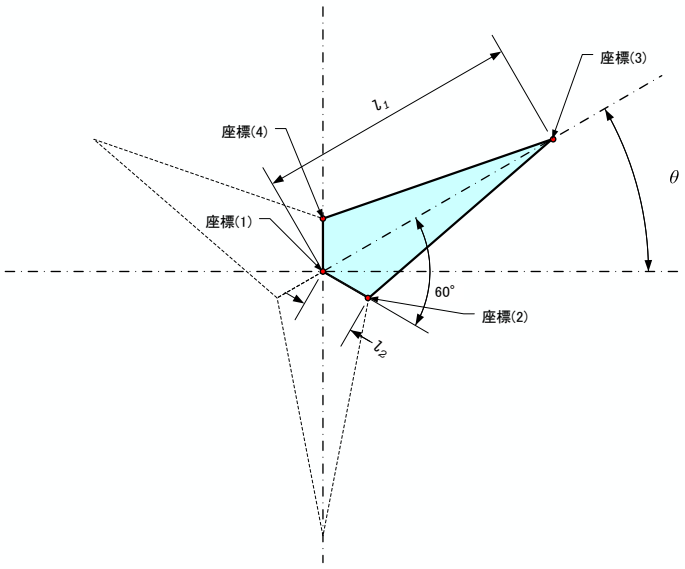

図 3: プロペラの座標系.

3　プログラム作成の練習

[**練習 1**]　リスト 1 のプログラムを実行させて，アニメーションのプログラムの動作を実感せよ.

[**練習 2**]　リスト 1 のプログラムを変えて，以下を理解せよ.

- 風車の角度の変化の量を変えて，その動作を確認せよ.
- 風車を 4 枚羽にせよ.

[**練習 3**]　簡単なアニメーションのプログラムを作成せよ.

4　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) アニメーションの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絵を提出すること.

提出要領はいつものとおり．ただし，期限は 12 月 19 日 (水)AM 8:45，課題名は「課題　GLUT を利用したアニメーション」とすること.