プログラミング演習IV課題レポート

池田　成

平成30年11月6日

第1日 2Dグラフィックス

1. プログラミングのツール:BCCにはMAKEユーティリティの他，grepやtouchなどプログラム開発を行う際に役に立つコマンドラインツールが含まれている．それぞれの機能や利用法について学習せよ．

Borland C++ Compiler 5.5のコマンドラインツールの中でも主要なものの機能を表1に示す．

表 1: コマンドラインツールの機能

ツール名	機能
bcc32	C/C++ファイルをコンパイルする
ilink32	オブジェクトモジュール，ライブラリモジュールを結合し，実行プログラムを作成
cpp32	プリプロセッサを通した後のコード見る
Imlib	DLLやモジュール定義ファイルからインポートライブラリを作成
make	MAKEユーティリティ
tdump	ファイルを16進ダンプする
tlib	ライブラリの作成や変更を行う
grep	複数のファイルから指定したキーワードを探し出す
touch	ファイルのタイムスタンプを最新にする

2. C言語：変数の「有効範囲（スコープ）」,「記憶域期間（記憶寿命）」について学習し，一般的な（ローカル）変数とグローバル変数，スタティック変数の違いを整理せよ．

有効範囲とは変数が使える範囲のことを指し，記憶期間は変数がメモリ上に存在している期間のことを指す．　ローカル変数は関数が終了するとメモリから消去されてしまう．しかしstatic修飾子を付けてスタティック変数にすることでプログラム実行中は変数がメモリに保持されるようになる．このとき，有効範囲はローカル変数と同様に宣言した関数内である．異なる関数で呼び出したい場合には有効範囲がプログラム全体であるグローバル変数を使用する．またグローバル変数の記憶期間はスタティック変数と同様である．

3. 図1，図2（[1] p.7）のような図形（星型正多角形）を描き，回転させるようなプログラムを作成せよ．

星型正多角形を描画し，回転させるプログラムの主要部をソースコード1に示す．

ソースコード 1: 星型正多角形の描画と回転

```
#define Vertex_Num 5 //星形正多角形の頂点の数
double rotAng=0.0;//多角形の回転角

void display(void) {
  int i;
  double theta,dt,x,y;

  glClear(GL_COLOR_BUFFER_BIT);
```

```
  glColor3d(1.0,0.0,0.0);
  dt=2.0*M_PI/Vertex_Num;
  theta=rotAng;
  glBegin(GL_LINE_LOOP);
  for(i=0;i<Vertex_Num;i++){
    x=cos(theta);
    y=sin(theta);
    glVertex2d(x,y);
    theta += dt*2;
  }
  glEnd();
  glFlush();
  rotAng+=2*M_PI/600;
}
```

4. 図 **3**，図 **4**（**[1] p.7**）のような図形（完全グラフ）を描き，回転させるようなプログラムを作成せよ．

完全グラフを描画し，回転させるプログラムの主要部をソースコード 2 に示す．

ソースコード 2: 完全グラフの描画と回転

```
#define Vertex_Num  8//星形正多角形の頂点の数
double rotAng=0.0;

void display(void) {
  int i,j,k;
  double theta,theta0,dt,x0,y0,x1,y1;

  glClear(GL_COLOR_BUFFER_BIT);
  glColor3d(1.0,0.0,0.0);
  dt=2.0*M_PI/Vertex_Num;
  theta=rotAng;
  for(i=0;i<Vertex_Num;i++){
    //全ての点について同様の処理
      glBegin(GL_LINES);
      for(j=0;j<Vertex_Num-1;j++){
        //他のすべての点へ直線を描画
        x0=cos(theta);
        y0=sin(theta);
        glVertex2d(x0,y0);
        theta0=theta+dt*(j+1);
        x1=cos(theta0);
        y1=sin(theta0);
        glVertex2d(x1,y1);
      }
      glEnd();
    theta+=(2*M_PI/Vertex_Num);
  }
  glFlush();
  rotAng+=2*M_PI/600;
  //関数が呼び出されるたびに回転
}
```

5. リスト **12**（**[1] p.7**）を解析し，数学関数のグラフを描くプログラムを作成せよ．例えば θ を 0 から 2π まで変化させながら，次の関数をプロットするとどんな図形が描かれるだろうか（コラム参照）．余力がある学生は，座標軸に目盛（文字）を加えてみよ．

数学関数のグラフを描画するプログラムの主要部をリスト 3 に示す．

ソースコード 3: 数学関数のグラフの描画

```
void display(void) {

  double x,y,theta;

  glClear(GL_COLOR_BUFFER_BIT);
  glColor3d(1.0,0.0,0.0);
  //x 軸^^c3^^9d 軸の描画
  glBegin(GL_LINES);
  glVertex2d(0.0,0.0);
  glVertex2d(10.0,0.0);
  glVertex2d(0.0,-10.0);
  glVertex2d(0.0,10.0);
  for(x=1.0;x<=10.0;x+=1.0){
      glVertex2d(x,-0.05);
      glVertex2d(x,0.05);
```

```
  }
  for(y=-10.0;y<=10.0;y+=1.0){
      glVertex2d(-0.05,y);
      glVertex2d(0.05,y);
  }
  glEnd();

  glColor3d(0.0,1.0,0.0);
  glBegin(GL_LINE_STRIP);
  for(theta=0;theta<=2*M_PI;theta+=0.1){
    //数学関数の計算
    glVertex2d(x,y);
  }
  glEnd();
  glFlush();
}
```

第 2 日 3D グラフィックス

1. 図 **6**（**[1] p.14**）に展開図を示す正四面体 $ABCD$ について，$\triangle BCD$ の重心 G を原点 O と一致させ，GA を x 軸，GB を y 軸とする座標系を考える．正四面体の一辺の長さを w とするとき，各頂点の **3** 次元座標を導出せよ．

正四面体の断面図 $\triangle BCD$，及び $\triangle ABH$ をリスト 1 に示す．はじめに $\triangle BCD$ を考える．I, J はそれぞれ BD, BC の中点であるので，中点連結定理により G は BH を $2:1$ に内分することが分かる．このとき，$BH = \frac{\sqrt{3}}{2}\ w$ となるのは自明なものとすると，

$$GH = \frac{1}{3} \times \frac{\sqrt{3}}{2} w = \frac{\sqrt{3}}{6}\ w$$
$$GB = \frac{2}{3} \times \frac{\sqrt{3}}{2} w = \frac{\sqrt{3}}{3}\ w$$

である．次に $\triangle BCD$ を考える．$AB = BH = \frac{\sqrt{3}}{2}\ w$ であるので，三平方の定理より

$$GA = \sqrt{AB^2 - GB^2} = \sqrt{w^2 - \frac{1}{3}w^2} = \frac{\sqrt{6}}{3} w$$

従って，$A(\frac{\sqrt{6}}{3}\ w, 0, 0)$，$B(0, \frac{\sqrt{3}}{3}\ w, 0)$，$C(0, \frac{-\sqrt{3}}{6}\ w, \frac{1}{2}\ w)$，$D(0, \frac{-\sqrt{3}}{6}\ w, \frac{-1}{2}\ w)$ となる．

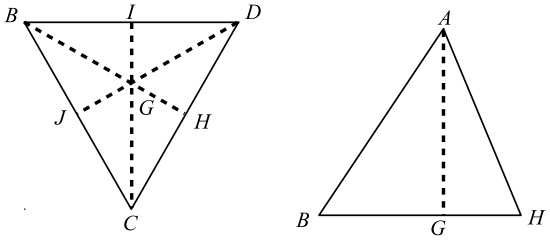

図 1: 正四面体の断面図

2. 前問で求めた結果をもとに，原点 O を中心とする半径 1 の球に内接する正四面体の頂点がリスト **14**（**[1] p9**）のように定まることを導出せよ．

正四面体において外接球の中心点 G' と各頂点の距離は全て等しく，その距離を r とおく．このとき，前問の結果より $\triangle ABH$ は図 2 のように表される．従って，三平方の定理より

$$\begin{aligned} G'B^2 &= GB^2 + G'G^2 \\ r^2 &= \frac{1}{3}w^2 + \left(\frac{\sqrt{6}}{3}w - r\right)^2 \\ r^2 &= \frac{1}{3}w^2 + \frac{2}{3}w^2 - \frac{2\sqrt{6}}{3}rw + r^2 \\ r &= \frac{\sqrt{6}}{4}w \\ \therefore GG' &= \frac{\sqrt{6}}{12} \end{aligned}$$

前問で求めた各頂点の 3 次元座標を x 軸方向に $-\sqrt{6}/12\ w$ シフトし，原点 O と中心点 G' を一致させると，$A(\sqrt{6}/4\ w, 0, 0)$, $B(-\sqrt{6}/12\ w,\ \sqrt{3}/3\ w,\ 0)$，$C(-\sqrt{6}/12\ w, -\sqrt{3}/6\ w, 1/2\ w)$, $D(-\sqrt{6}/12\ w, -\sqrt{3}/6\ w, -1/2\ w)$ となる．ここで半径 $r = 1$ すなわち $w = 2\sqrt{6}/3$ とおけば，$A(1, 0, 0)$，$B(-1/3, 2\sqrt{2}/3, 0)$，$C(-1/3, -\sqrt{2}/3, \sqrt{6}/3)$, $D(-1/3, -\sqrt{2}/3, -\sqrt{6}/3)$ となり，リスト 14（[1] p9）と一致していることが分かる．

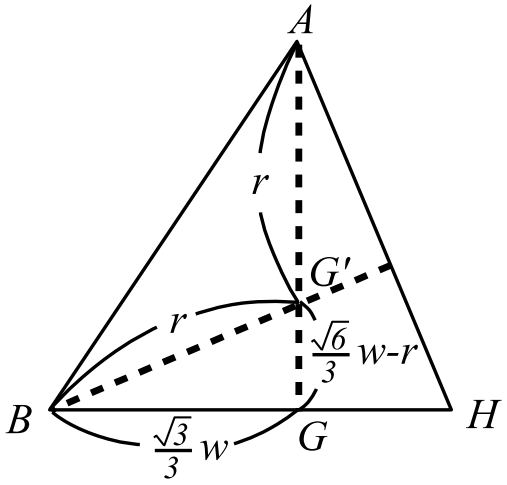

図 2: 正四面体の断面図と外接球の中心点 G'

3. 9.2（[1] p13）のアームロボットについて，キー入力で台座の回転と，各関節の傾斜角を制御できるような　対話プログラムを作成してみよう．

アームロボットの描画，及び制御するプログラムの主要部をソースコード 4 に示す．keyin 関数でキーボードによる入力を受け付け，入力の状態をグローバル変数に保存．その後 display 関数内で保存された入力の状態に応じて回転の制御を行っている．

ソースコード 4: 対話型ロボットアームの描画

```
GLfloat rotAng[3];
int b_stat=0,l_stat=0,u_stat=0,pause=0;
int dance=0;

void display(void)
{

  glClear(GL_COLOR_BUFFER_BIT);

  glMatrixMode(GL_MODELVIEW);
  glLoadIdentity();
        gluLookAt(1,1,1.5, 0,0,0, 0,1,0);
        glRotated(rotAng[0], 0, 1, 0);

  glRotated(rotAng[0],0,1,0);
  glCallList(ID_B);

        if(pause==0){
                if(b_stat==1) rotAng[0]+=3.0;
                if(b_stat==-1) rotAng[0]-=3.0;
        }/*台座の回転角の制御*/

  glTranslated(0,HEIGHT_B,0);
  glRotated(rotAng[1],0,0,1);
  glCallList(ID_L);

        if(pause==0){
                if(l_stat==1) rotAng[1]+=3.0;
                if(l_stat==-1) rotAng[1]-=3.0;
        }/*下腕の回転角の制御*/

  glTranslated(0,HEIGHT_L,0);
  glRotated(rotAng[2],0,0,1);
  glCallList(ID_U);

        if(pause==0){
                if(u_stat==1) rotAng[2]+=3.0;
                if(u_stat==-1) rotAng[2]-=3.0;
        }/*上腕の回転角の制御*/

        //if(dance==1) Dance();
  //dance フラグが成立したらランダムに回転角を制御

        glutSwapBuffers();

}

void keyin(unsigned char key,int x,int y)
{
        switch (key) {
         case 'q':
```

```
        case 'Q':exit(0);break;
        //exit
        case 'w':b_stat=1;;break;
        //台座回転
        case 'e':b_stat=-1;break;
        //台座逆回転
        case 'r':b_stat=0;break;
        //台座回転停止
        case 's':l_stat=1;break;
        //下腕回転
        case 'd':l_stat=-1;break;
        //下腕逆回転
        case 'f':l_stat=0;break;
        //下腕回転停止
        case 'x':u_stat=1;break;
        //上腕回転
        case 'c':u_stat=-1;break;
        //上腕逆回転
        case 'v':u_stat=0;break;
        //上腕回転停止
        case 'z':{dance=!dance;b_stat=0;
                l_stat=0;u_stat=0;break;
        }
        //ランダムに回転
        case 'p':pause=!pause;break;
        //全体一時停止
       }
}
```

4. タイマコールバック関数を使って，アームロボットを制御するようなプログラムを作成してみよう．アームロボットが踊っているかのように見せるには，どんな工夫ができるだろうか．

アームロボットの踊りを制御するプログラムの主要部をソースコード 5 に示す．ソースコード 4 のプログラムにこの関数を加えることでキーボードからｚを入力するとランダムにロボットアームが動き出す．

ソースコード 5: ロボットアームのダンスプログラム

```
void Dance()
{
        int i;

  if(pause==0){
    for(i=0;i<3;i++){
            srand((unsigned)time(NULL)*(i+9));
      rotAng[i] += rand()%36-18;
      printf("[%d]:%d\n",i,rand()%36-18);
        }
       }
}
```

第 3 日 3D モデルと照光処理

1. 空間中の任意の 3 点 $P_i = (x_i, y_i, x_i), i = 1, 2, 3$ が与えられたとき，$\triangle P_1P_2P_3$ の単位法線ベクトルをすべて求める式を導出せよ．

まず 2 つのベクトルを求めると

$$\overrightarrow{P_1P_2} = (x_2 - x_1, y_2 - y_1, z_2 - z_1)$$
$$\overrightarrow{P_2P_3} = (x_3 - x_2, y_3 - y_2, z_3 - z_2)$$

$\overrightarrow{P_1P_2}, \overrightarrow{P_2P_3}$ の外積 $\overrightarrow{N}$ を求めると

$$\overrightarrow{N} = \overrightarrow{P_1P_2} \times \overrightarrow{P_2P_3}$$

従って，すべての単位法線ベクトル $\overrightarrow{n}$ は

$$\overrightarrow{n} = \pm\frac{\overrightarrow{N}}{|\overrightarrow{N}|}$$

で表される．

2. 空間中の任意の 3 点 $P_i = (x_i, y_i, x_i), i = 1, 2, 3$ が与えられたとき，$\triangle P_1P_2P_3$ のどちらが表（CCW）で，どちらが裏（CW）と判定できるか，判定方法を導出せよ．

前問の結果より $\overrightarrow{N}$ は表向きの法線ベクトルである．ここで，視線方向のベクトル $\overrightarrow{v}$ を導入すると，次にように $\overrightarrow{N}$ と $\overrightarrow{v}$ の内積の符号によって表裏の判別が行える．

- $\overrightarrow{N} \cdot \overrightarrow{v} < 0$ なら表（CCW）
- $\overrightarrow{N} \cdot \overrightarrow{v} > 0$ なら裏（CW）

3. phong の照光モデルについて，詳しく調べよ．

phong の照光モデルとは，Bui Tuong Phong によって開発された物体の反射特性を数式で表現したモデルのことである．このモデルでは，反射を鏡面反射，拡散反射，環境反射の 3 つの成分で考えることで簡潔に表し，コンピュータビジョンでは広く用いられている．

拡散反射は乱反射とも呼ばれ，反射光は全方位に散乱するためにその強さは観測者の視点位置によらない．また，拡散反射成分 L_d は次のように θ の余弦に比例する．

$$L_d = I_i k_d \cos\theta = I_i k_d \big(\overrightarrow{N} \cdot \overrightarrow{L}\big) \tag{1}$$

ここで，I_i は入射光の照度，k_d は拡散反射係数，$\overrightarrow{N}$ は反射面の法線ベクトル，$\overrightarrow{L}$ は面との交点から光源へ向かうベクトル，θ は $\overrightarrow{N}$ と $\overrightarrow{L}$ のなす角である．なお，$\big(\overrightarrow{N} \cdot \overrightarrow{L}\big) < 0$，すなわち光源が面の裏側から当たっているような場合は拡散反射光成分は 0 として扱う．

鏡面反射は金属などのなめらかな表面で生じるハイライトでも知られ，反射光は指向性を持つのでその強さは視点に依存して変化する．また，鏡面反射成分 L_s は次のように ϕ の余弦の α 乗に比例する．α は光沢度といいハイライトの鋭さを表す定数（$1 \leq \alpha$）である．

$$L_s = I_i k_s \cos^\alpha \phi = I_i k_s (\overrightarrow{V} \cdot \overrightarrow{R})^\alpha \tag{2}$$

ここで，I_i は入射光の照度，k_s は鏡面反射係数，$\overrightarrow{V}$ は視線ベクトルの逆ベクトル，$\overrightarrow{R}$ は入射光の正反射ベクトル，ϕ は $\overrightarrow{V}$ と $\overrightarrow{R}$ のなす角である．なお，$\big(\overrightarrow{N} \cdot \overrightarrow{L}\big) < 0$，及び $(\overrightarrow{V} \cdot \overrightarrow{R}) < 0$ の場合は鏡面反射光成分は 0 として扱う．加えて正反射ベクトル $\overrightarrow{R}$ は以下のように計算することができる．

$$\overrightarrow{R} = 2\big(\overrightarrow{N} \cdot \overrightarrow{L}\big)\overrightarrow{N} - \overrightarrow{L} \tag{3}$$

環境光は光源から発せられ 1 回以上の反射を経て物体表面に届くので間接光とも呼ばれ，環境光の強さは一定とし，その反射光の強さも方向によらず一定とする．つまり，環境反射成分 L_a は次のように表される．

$$L_a = k_a I_a \tag{4}$$

ここで，k_a は環境反射係数，I_a は環境光の照度である．なお，k_a は物体表面ごとに，I_a はシーンごとに異なる値となる．

従って，phong の照光モデルでは反射特性 L_r をまとめて

$$\begin{aligned} L_r &= L_a + L_d + L_s \\ &= k_a I_a + I_i k_d \cos\theta + I_i k_s \cos^\alpha \phi \end{aligned} \tag{5}$$

のように表す．

4. 正四面体，正六面体，正八面体をスムーズシェーディングで表示するために，各頂点の単位法線ベクトルをどのように定めればよいか考え，実装せよ．

調査中

5. リスト 27（[1] pp.19-20）を参考に，アームロボットをシェーディング表示するものに変更してみよ．

ソースコード 4 で作成したロボットアームをシューディング表示するためのプログラムの主要部をソースコード 6 に示す．display 関数で glCallList() 関数を用いることで init 関数で作成したディスプレイリストを呼び出している．

ソースコード 6: ロボットアームのシューディング

```
typedef struct materialStruct{
  GLfloat ambient[4],diffuse[4],
          speculer[4];
  GLfloat shininess;
}materialStruct;
```

```
materialStruct brassMaterials={
  {0.33,0.22,0.03,1},
  {0.78,0.57,0.11,1},
  {0.99,0.91,0.81,1},
  27.8
};
materialStruct redPlasticMaterials={
  {0.3,0.0,0.0,1},
  {0.6,0.0,0.0,1},
  {0.8,0.6,0.6,1},
  32.0
};
materialStruct greenPlasticMaterials={
  {0.0,0.3,0.0,1},
  {0.0,0.6,0.0,1},
  {0.6,0.8,0.6,1},
  32.0
};
GLUquadric *myQuad;

void materials(materialStruct *m)
{
  glMaterialfv(GL_FRONT_AND_BACK,
               GL_AMBIENT,
               m->ambient);
  glMaterialfv(GL_FRONT_AND_BACK,
               GL_DIFFUSE,
               m->diffuse);
  glMaterialfv(GL_FRONT_AND_BACK,
               GL_SPECULAR,
               m->speculer);
  glMaterialf(GL_FRONT_AND_BACK,
              GL_SHININESS,
              m->shininess);
}

void init(void) {
        myQuad = gluNewQuadric();
        gluQuadricDrawStyle(myQuad, GLU_FILL);
        glEnable(GL_DEPTH_TEST);
        glEnable(GL_LIGHTING);
        glEnable(GL_LIGHT0);
        glClearColor(0.0, 0.0, 0.0, 1.0);
        gluLookAt(1,-1,-1, 0,1,1, 0,0,1);
//台座
        glNewList(ID_B, GL_COMPILE);
        glPushMatrix();
        glTranslatef(0.0, 0.5*HEIGHT_L, 0.0);
        glScalef(WIDTH_B, HEIGHT_B, WIDTH_B);
        glRotated(90, 1.0, 0, 0);
        materials(&brassMaterials);
        gluCylinder(myQuad,1,1,2,10,5);
        glPopMatrix();
        glEndList();
//下腕
        glNewList(ID_L, GL_COMPILE);
        glPushMatrix();
        glTranslatef(0.0, 0.5*HEIGHT_L, 0.0);
        glScalef(WIDTH_L, HEIGHT_L, WIDTH_L);
        materials(&redPlasticMaterials);
        glutSolidCube(1.0);
        glPopMatrix();
        glEndList();
//上腕
        glNewList(ID_U, GL_COMPILE);
        glPushMatrix();
        glTranslatef(0.0, 0.5*HEIGHT_L, 0.0);
        glScalef(WIDTH_U, HEIGHT_U, WIDTH_U);
        materials(&greenPlasticMaterials);
        glutSolidCube(1.0);
        glPopMatrix();
        glEndList();
}
```

1　感想

今回のプログラミング演習を通して OpenGL をある程度使いこなせるようになった．またコンピュータグラフィクスの知識が深まったとも感じている．今後の卒業研究などにも今回学んだことは生かせると思うのでこの授業を受講してよかったと思っている．しかし中間発表や学祭などで忙しくなってしまい課題全てを終えることが出来なかった．もっと早めに準備していればと後悔している．

2　今後の改善案

テキスト p7 の図 1 から図 4 が小さすぎてみづらかった．また 2 日目の課題 3 と課題 4 の違いがテキストではわかりづらかった．

参考文献

[1] 高橋章，“H30-Ec5 プログラミング演習 IV テキスト”，2018/9/11.

[2] Borland C++ Compiler 5.5 インストールと使い方，http://www6.plala.or.jp/mnagaku/cmag/ac20005/，(2018/11 閲覧).

[3] 東京電機大学，“ 工学部第二部情報通信工学実験　グラフィックス応用テキスト”，https://www.vcl.jp/~kanazawa/raytracing/?page_id=240，(2018/11 閲覧).

[4] プログラミング入門サイト 〜bituse〜，http://bituse.info/c/27，(2018/11 閲覧).