1 課題 I

1.1 課題 I-1

1.1.1 GLEW

GLEW とは，グラフィックスハードウェアの拡張機能を使用可能にするライブラリである．特に Windows では，もともとサポートしている OpenGL のバージョンが 1.1 のため，新しい機能を使用することができない．そのため，GLEW を用いてグラフィックスハードウェアが持つ全ての機能を使えるようにする．

1.1.2 GLFW

GLFW とは，デスクトップでの OpenGL，OpenGL ES，Vulkan 開発用のオープンソースのマルチプラットフォームライブラリである．ウィンドウやコンテキストを作成し，入力を管理するためのシンプルな API を提供する．

1.1.3 GLUI

GLUI とは，ダイアログウィンドウで使用されるボタンやチェックボックスなどのコントロールを OpenGL で作成できるライブラリである．

1.1.4 Vulkan

Vukan とは，グラフィックス API のことで，直接 GPU にアクセスできる構造によって，これまでのグラフィックス API に比べてより速い描画をすることが可能となる．

1.1.5 OpenGL ES

OpenGL ES とは，電化製品や車両などの組み込みおよびモバイルシステムで，高度な 2D，3D グラフィックスをレンダリングするためのクロスプラットフォーム API である．

1.1.6 WebGL

WebGL とは，ウェブブラウザ上で OpenGL ES 相当の描画処理を行うことができる低レベルの API である．JavaScript の API として実装されているため，改めてプラグイン等をインストールすることなく実行できる．

1.2 課題 I-2

1.2.1 make ユーティリティ

make とは大規模プログラムのコンパイルを簡略化するツールである．makefile にファイルの関係を記述することで，各ファイルの更新を取得し，必要なものだけをコンパイルすることができる．以下のコマンドで makefile を実行することができる．

```
1 make
```

1.2.2 grep

grep とは，テキストファイルの中から正規表現と一致する行を検索し，出力するコマンドである．以下のコマンドで grep を使用することができる．

```
1 grep［オプション］［検索文字列（正規表現）］［ファイル名］
```

1.2.3 touch

touch とはファイルの最終更新日を変更するコマンドである．以下のコマンドで touch を使用することができる．

```
1 touch［オプション］ファイル 1 ファイル 2 …
```

1.2.4 CMake

CMake とは，ソフトウェアをビルドやテスト，パッケージ化するために設計されたオープンソースのクロスプラットフォームツールである．プラットフォームやコンパイラに依存しないシンプルな設定ファイルを使用することでソフトウェアのコンパイルプロセスを制御し，makefile とワークスペースを生成するために使用される．

1.2.5 Subversion

Subversion とは，データの安全な避難所としての信頼性を特徴とするオープンソースの集中型バージョン管理システムである．個人から大規模なエンタープライズオペレーションまで，さまざまなユーザやプロジェクトのニーズをサポートすることができる．

1.2.6 Git

Git とは，小規模なプロジェクトから大規模なプロジェクトまで，効率的に処理できるように設計されたオープンソースの分散型バージョン管理システムである．非常に高速なパフォーマンスを備えた小さなフットプリントを備えている．

1.3 課題 I-3

C 言語での変数は，その宣言を書く場所によって有効範囲 (スコープ) が異なる．変数のスコープ外からは，その変数を参照することができない．一方，どの関数にも属さない場所で宣言された変数をグローバル変数と呼び，プログラム内のどこからでもアクセスが可能となる．記憶域期間には自動記憶域期間と静的記憶域期間がある．自動記憶域期間を持つ変数は auto を使用して宣言される．この変数の記憶寿命は宣言した関数内に入った時から出る時までである．このように，関数内で宣言され，自動記憶域期間をもつ変数をローカル変数という．一方，static を使用して宣言された変数は静的記憶域期間を持ち，記憶寿命はプログラムの開始から終了までとなる．このような変数をスタティック変数と呼ぶ．

1.4 課題 I-4

星形正多角形を描き，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　星形正多角形の描画と回転

```
1  void display() {
2      glClear(GL_COLOR_BUFFER_BIT);
3      glColor3d(1.0, 1.0, 1.0);
4
5      dt = 2.0 * M_PI / NUM;
6      theta = rotAng;
7
8      for (i = 0; i < NUM; i++) {
9          x[i] = cos(theta);
10         y[i] = sin(theta);
11         theta += dt;
12     }
13
14     for (i = 0; i < NUM; i++) {
15         glBegin(GL_LINES);
16         glVertex2d(x[i], y[i]);
17         glVertex2d(x[(i + 2) % NUM], y[(i + 2) % NUM]);
18         glEnd();
19     }
20
21     glFlush();
22     rotAng += 3.0 * M_PI / 180.0;
23 }
```

定数 NUM を変更することで，星型正多角形の角の数を変えることができる．

星型正多角形を描くには，全頂点において，2 つ隣の頂点に線を引くことで描画することができる．そのため，あらかじめ線を引く頂点の座標を配列に格納し，GL_LINES を用いることで線を引く．変数 theta をインクリメントすることで全頂点においてこの作業を行う．

このプログラムを使用して描画した星型正五角形と星型正六角形を図 1，2 に示す．

1.5 課題 I-5

完全グラフを描き，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

```
1  void display() {
2      glClear(GL_COLOR_BUFFER_BIT);
3      glColor3d(1.0, 1.0, 1.0);
4
5      dt = 2.0 * M_PI / NUM;
```

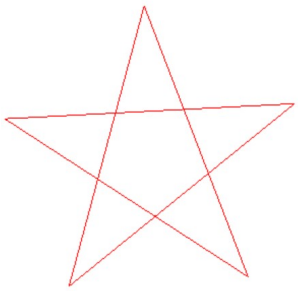

図 1　星型正五角形

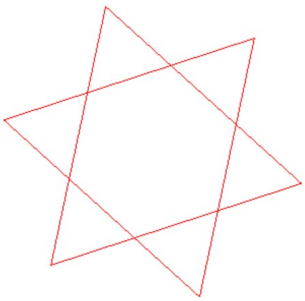

図 2　星型正六角形

```
6       theta = rotAng;
7
8       for (i = 0; i < NUM; i++) {
9           x[i] = cos(theta);
10          y[i] = sin(theta);
11          theta += dt;
12      }
13
14      for (i = 0; i < NUM; i++) {
15          for (j = i + 1; j < NUM; j++) {
16              glBegin(GL_LINES);
17              glVertex2d(x[i], y[i]);
18              glVertex2d(x[j], y[j]);
19              glEnd();
20          }
21      }
22
23      glFlush();
24      rotAng += 3.0 * M_PI / 180.0;
25  }
```

定数 NUM を変更することで，完全グラフの角の数を変えることができる．

完全グラフを描くには，全頂点において，他の頂点全てに線を引くことで描画することができる．そのため，あらかじめ線を引く頂点の座標を配列に格納し，GL_LINES を用いることで線を引く．その際，すでに線を引いてある 2 点間について，重ねて線を引かないように，15 行目に書いてあるように，ループ数を減らしていく．変数 theta をインクリメントすることで全頂点においてこの作業を行う．

このプログラムを使用して描画した星型正五角形と星型正六角形を図 3，4 に示す．

1.6　課題 I-6

カージオイド，サイクロイド，4 尖点の内サイクロイドを描画するプログラムの主要部を，それぞれソースコード 3,4,5 に示す．

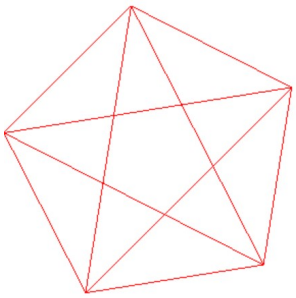

図 3　5 頂点完全グラフ

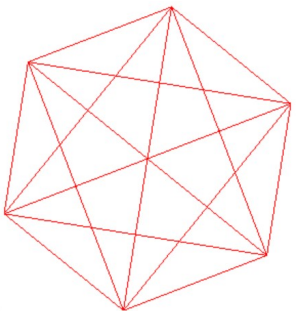

図 4　6 頂点完全グラフ

ソースコード 3　カージオイドの描画

```
1  void display(void){
2          glColor3d(0.0, 0.0, 1.0);
3          glBegin(GL_LINE_STRIP);
4
5          for (theta = 0; theta <= 2 * M_PI; theta += 2 * M_PI / 100) {
6                  x = cos(theta) * (1 + cos(theta));
7                  y = sin(theta) * (1 + cos(theta));
8                  glVertex2d(x, y);
9          }
10
11         glEnd();
12         glFlush();
13 }
```

ソースコード 4　サイクロイドの描画

```
1  void display(void){
2          glColor3d(0.0, 0.0, 1.0);
3          glBegin(GL_LINE_STRIP);
4
5          for (theta = 0; theta <= 2 * M_PI; theta += 2 * M_PI / 100) {
6                  x = theta - sin(theta);
7                  y = 1 - cos(theta);
8                  glVertex2d(x, y);
9          }
10
11         glEnd();
12         glFlush();
13 }
```

ソースコード 5　4 尖点の内サイクロイドの描画

```
1  void display(void){
2          glColor3d(0.0, 0.0, 1.0);
```

```
3           glBegin(GL_LINE_STRIP);
4
5           for (theta = 0; theta <= 2 * M_PI; theta += 2 * M_PI / 100) {
6                   x = (cos(theta)) * (cos(theta)) * (cos(theta));
7                   y = (sin(theta)) * (sin(theta)) * (sin(theta));
8                   glVertex2d(x, y);
9           }
10
11          glEnd();
12          glFlush();
13  }
```

θ は 0 から 2π まで変化させ，100 点間を線で結んでいる．描画したグラフを図 5,6,7 に示す．

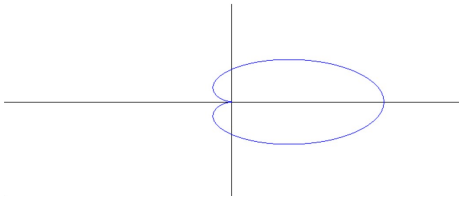

図 5　カージオイド

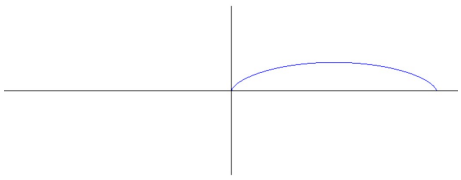

図 6　サイクロイド

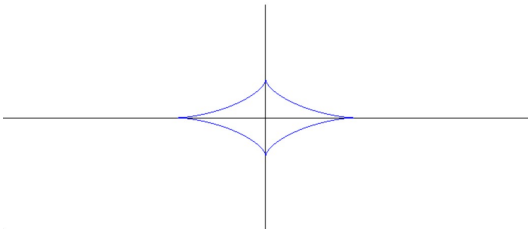

図 7　4 尖点の内サイクロイド

2　課題 II

2.1　課題 II-1

右手系の座標を考える．頂点 A は x 軸上の点だから y，z 座標は 0 となり，x 座標については，辺 AG の長さから求められる．頂点 B は y 軸上の点だから x，z 座標は 0 となり，y 座標については，辺 BG の長さから求められる．頂点 C，D は，y-z 平面の点だから x 座標は 0 となる．また，y 座標は辺 GH の長さから求められ，z 座標は，それぞれ辺 CH，DH の長さから求められる．

正四面体の一辺の長さを w として各辺の長さを求める．図 8 に示すように，三角形の重心は中線を 2:1 に内分するため，辺 BG の長さは $\frac{\sqrt{3}}{3}w$，辺 GH の長さは $\frac{\sqrt{3}}{6}w$ となる．図 9 に示すように，△BGA に三平方の定理を適用すると，辺 AG の長さは $\frac{\sqrt{6}}{3}w$ となる．点 H は辺 CD の中点であるから，辺 CH，辺 DH は $\frac{w}{2}$ と

なる．よって各頂点の座標は以下のようになる．

$$A\text{: } (\frac{\sqrt{6}}{3}w, 0, 0)$$
$$B\text{: } (0, \frac{\sqrt{3}}{3}w, 0)$$
$$C\text{: } (0, -\frac{\sqrt{6}}{3}w, \frac{w}{2})$$
$$D\text{: } (0, -\frac{\sqrt{6}}{3}w, -\frac{w}{2})$$

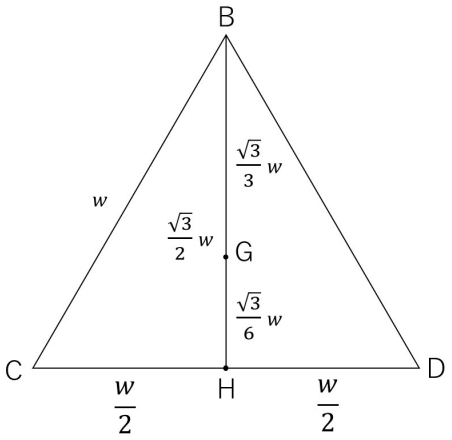

図 8　△BCD

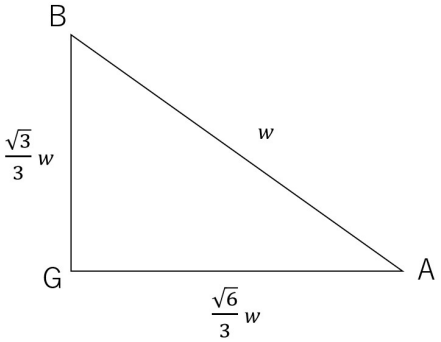

図 9　△BGA

2.2　課題 II-2

一辺の長さを w とする正四面体の重心と原点 O が重なるように正四面体を平行移動させ，w の値を調整することで，原点 O を中心とする半径 1 の球に正四面体を内接させることができる．

平行移動後の △BCD の重心位置 G と原点 O の距離を l とし，w と l を求める．図 10 に示す △BGA と △BGO に三平方の定理を適用すると，以下の式が得られる．

$$\begin{cases} \left(\frac{\sqrt{3}}{3}w\right)^2 + (l+1)^2 = w^2 \\ \left(\frac{\sqrt{3}}{3}w\right)^2 + l^2 = 1^2 \end{cases}$$

これを解くと，以下のようになる．

$$\begin{cases} w = \frac{2\sqrt{6}}{3} \\ l = \frac{1}{3} \end{cases}$$

上記の値を課題 II-1 の結果に適用すると，正四面体の頂点がリスト 14 のように定まることがわかる．

2.3　課題 II-3

ロボットアームを構成するプログラムの主要部をソースコード 6 に示す．

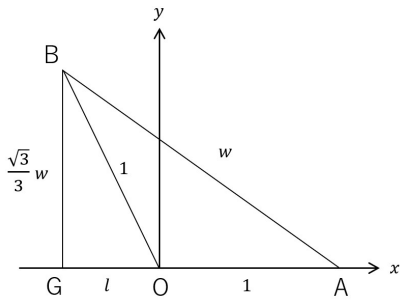

図 10　平行移動後の △BGA

ソースコード 6　ロボットアームの情報

```
1  void init() {
2      glClearColor(1.0, 1.0, 1.0, 1.0);
3      myQuad = gluNewQuadric();
4
5      glNewList(ID_B, GL_COMPILE);
6      glColor3f(0.0, 0.0, 0.0);
7      glPushMatrix();
8      glRotated(90.0, 1.0, 0.0, 0.0);
9      gluCylinder(myQuad, RADIUS_B, RADIUS_B, HEIGHT_B, 10, 2);
10     glPopMatrix();
11     glEndList();
12
13     glNewList(ID_L, GL_COMPILE);
14     glColor3f(0.0, 0.0, 1.0);
15     glPushMatrix();
16     glTranslated(0.0, 0.5 * HEIGHT_L, 0.0);
17     glScalef(WIDTH_L, HEIGHT_L, WIDTH_L);
18     glutWireCube(1.0);
19     glPopMatrix();
20     glEndList();
21
22     glNewList(ID_U, GL_COMPILE);
23     glColor3f(1.0, 0.0, 0.0);
24     glPushMatrix();
25     glTranslated(0.0, 0.5 * HEIGHT_U, 0.0);
26     glScalef(WIDTH_U, HEIGHT_U, WIDTH_U);
27     glutWireCube(1.0);
28     glPopMatrix();
29     glEndList();
30
31 }
```

次に，キーボードコールバック関数をソースコード 7 に示す.

ソースコード 7　キーボードコールバック関数

```
1  void keyin(unsigned char key, int x, int y) {
2      switch (key) {
3      case 'x':
4          rotAng[0] += SPEED;
5          glutPostRedisplay();
6          break;
7
8      case 'z':
9          rotAng[0] -= SPEED;
10         glutPostRedisplay();
11         break;
12
13     case 's':
14         rotAng[1] += SPEED;
15         glutPostRedisplay();
16         break;
17
18     case 'a':
19         rotAng[1] -= SPEED;
20         glutPostRedisplay();
21         break;
22
23     case 'w':
24         rotAng[2] += SPEED;
25         glutPostRedisplay();
26         break;
27
28     case 'q':
29         rotAng[2] -= SPEED;
30         glutPostRedisplay();
31         break;
32
33     default: break;
34     }
35 }
```

特定のキーを入力すると，rotAng の値が変化し，ロボットアームを動かすことができるようになる．q，w，a，s，z，x で台座やロボットアームを動かす．キーボード配列で右側のキーが増加，左側のキーが現象をするようになっている．また，SPEED の値を変化させることで台座の回転角や各関節の傾斜角の変化量を変えることができる．

2.4　課題 II-4

実装したタイマコールバック関数をソースコード 8 に示す．

ソースコード 8　タイマコールバック関数

```
1  static void timer(int dummy) {
2      glutTimerFunc(100, timer, 0);
3      glMatrixMode(GL_MODELVIEW);
4
5      rotAng[0] += 30;
6      rotAng[1] += 10 * cos(time / 5);
7      rotAng[2] += 100 * cos(time);
8      time++;
9
10     glutPostRedisplay();
11 }
```

台座に関しては 2 つの方向に動くと震えているように見えたため，一定方向に素早く回転するようにした．アームに関しては，2 つの方向に動かした方がより踊っているように見えたため，三角関数を用いた．上のアームの動きは早くし，下のアームの動きは少しゆっくりにすることで，全力で踊っているように見せることができた．踊っているロボットアームを図 11 に示す．

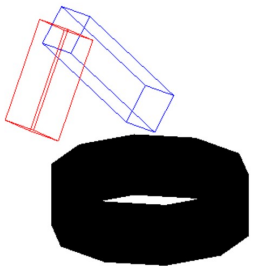

図 11　踊っているロボットアーム

3　課題 III

3.1　課題 III-1

2 つのベクトルの外積は，その 2 つのベクトルが成す平面の法線ベクトルになる．$\overrightarrow{P_1P_2}$，$\overrightarrow{P_2P_3}$ を 2 つのベクトルとし，$\triangle P_1P_2P_3$ をベクトルの成す平面とすると，$\overrightarrow{P_1P_2} \times \overrightarrow{P_2P_3}$ によって，$\triangle P_1P_2P_3$ の単位法線ベクトルを全て求めることができる．

$\triangle P_1P_2P_3$ の全ての単位法線ベクトルを $\overrightarrow{n}$ とすると，以下のようになる．

$$\overrightarrow{n} = \pm\frac{\overrightarrow{P_1P_2} \times \overrightarrow{P_2P_3}}{|\overrightarrow{P_1P_2} \times \overrightarrow{P_2P_3}|}$$

3.2 課題 III-2

空間中の任意の 3 点が与えられたとき，$\triangle P_1P_2P_3$ の 3 点を反時計回りにたどる向きのベクトルから単位法線ベクトルを求めると，その単位法線ベクトルは面の表側を向く．これにより面の表裏を判別できる．

3.3 課題 III-3

Phong モデルとは，物体の反射特性を数式で表現した最初のモデルである．Phong モデルでは，拡散反射成分は入射角の余弦と入射光の強さに比例するとされている．以下に式を示す．

$$I_d = I_i k_d \cos\alpha = I_i k_d (L \cdot N)$$

ここで，I_d は拡散反射光の強さ，I_i は入射光の強さ，k_d は拡散反射率，α は入射角，L は光源方向ベクトル，N は表面法線ベクトルを示す．

また，Phong モデルは，鏡面反射光の強さは入射角の余弦の n 乗と鏡面反射率に比例するとされている．以下に式を示す．

$$S = I_i W \cos^n\gamma$$

ここで S は鏡面反射光の強さ，n はハイライトの特性，γ は光源の反射ベクトルと視線ベクトルのなす角を示す．また，厳密には鏡面反射率は光の波長によって異なるが，一定値 W とされている．

そして，拡散，鏡面反射，環境光の 3 要素を考慮した場合の光の強さは次式のように表される．

$$I = k_d \cos\alpha + I_i W \cos^n\gamma + I_a k_a$$

ここで，I は視点に入射する光の強さで，I_a は環境光の強さ，k_a は環境光定数を示している．

3.4 課題 III-4

頂点の法線ベクトルは，重心からその頂点に向かう向きとなる．よって，原点を中心とする半径 1 の円に内接する正多面体の頂点の法線ベクトルは，その頂点の位置ベクトルと一致する．正四面体，正六面体，正八面体を描画するプログラムの主要部をソースコード 9,10,11 にそれぞれ示す．

ソースコード 9　正四面体の描画

```
1  GLdouble vP[4][3] = {
2      {1.000, 0.000, 0.000},
3      {-0.333, 0.943, 0.000},
4      {-0.333, -0.471, 0.816},
5      {-0.333, -0.471, -0.816}
6  };
7  int tP[4][3] = {
8      {0, 1, 2},{0, 3, 1},
9      {0, 2, 3},{1, 3, 2}
10 };
11
```

```
12 void display() {
13     int i, j;
14     glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
15     glMatrixMode(GL_MODELVIEW);
16     glLoadIdentity();
17 
18     glBegin(GL_TRIANGLES);
19     for (i = 0; i < 4; i++) {
20         for (j = 0; j < 3; j++) {
21             glNormal3dv(vP[tP[i][j]]);
22             glVertex3dv(vP[tP[i][j]]);
23         }
24     }
25     glEnd();
26     glutSwapBuffers();
27 }
```

ソースコード 10　正六面体の描画

```
1  GLdouble vP[8][3] = {
2      {-0.577,-0.577,0.577},{0.577,-0.577,0.577},
3      {0.577,-0.577,-0.577},{-0.577,-0.577,-0.577},
4      {-0.577,0.577,0.577}, {0.577,0.577,0.577},
5      {0.577,0.577,-0.577}, {-0.577,0.577,-0.577}
6  };
7  int tP[6][4] = {
8      {3,2,1,0},{0,1,5,4},{1,2,6,5},
9      {4,5,6,7},{3,7,6,2},{0,4,7,3}
10 };
11 
12 void display() {
13     int i, j;
14     glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
15     glMatrixMode(GL_MODELVIEW);
16     glLoadIdentity();
17 
18     glBegin(GL_TRIANGLES);
19     for (i = 0; i < 6; i++) {
20         for (j = 0; j < 4; j++) {
21             glNormal3dv(vP[tP[i][j]]);
22             glVertex3dv(vP[tP[i][j]]);
23         }
24     }
25     glEnd();
26     glutSwapBuffers();
27 }
```

ソースコード 11　正八面体の描画

```
GLdouble vP[6][3] = {
    {1,0,0},{0,1,0},{0,0,1},
    {-1,0,0},{0,-1,0},{0,0,-1}
};
int tP[8][3] = {
    {0,1,2},{5,1,0},{3,1,5},{2,1,3},
    {2,4,0},{0,4,5},{5,4,3},{3,4,2}
};

void display() {
    int i, j;
    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
    glMatrixMode(GL_MODELVIEW);
    glLoadIdentity();

    glBegin(GL_TRIANGLES);
    for (i = 0; i < 8; i++) {
        for (j = 0; j < 3; j++) {
            glNormal3dv(vP[tP[i][j]]);
            glVertex3dv(vP[tP[i][j]]);
        }
    }
    glEnd();
    glutSwapBuffers();
}
```

vP に頂点の位置，tP にトポロジ情報を格納している．法線ベクトルの指定には，vP をそのまま利用している．

このプログラムを利用して描画した図形をそれぞれ図 12，13，14 に示す．

3.5　課題 III-5

リスト 27 のロケットをアームロボットに変更した．各パーツの回転角度を格納するための rotAng をグローバル変数として定義し，display 関数内には glRotated を加えた．また，タイマーコールバック関数を用いて課題 II-4 のように踊って見えるようにした．変更したプログラムをソースコード 12 に示す．

ソースコード 12　リスト 27 の変更部分

```
GLdouble rotAng[3] = { 0 };

void display(void) {
        glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
        glMatrixMode(GL_MODELVIEW);
        glLoadIdentity();
        gluLookAt(1.0, 1.0, 1.0, 0.0, 0.5, 0.0, 0.0, 1.0, 0.0);
```

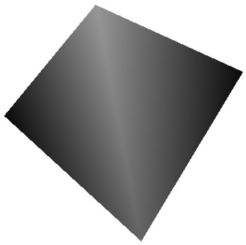

図 12　正四面体

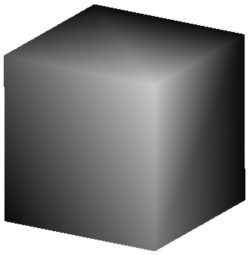

図 13　正六面体

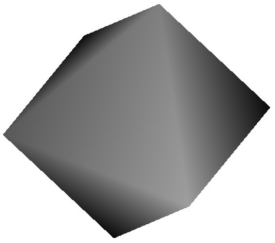

図 14　正八面体

```
8           glRotated(rotAng[0], 0, 1, 0);
9           glCallList(ID_B);
10          glTranslatef(0.0, HEIGHT_B, 0.0);
11          glRotated(rotAng[1], 0, 0, 1);
12          glCallList(ID_L);
13          glTranslatef(0.0, HEIGHT_L, 0.0);
14          glRotated(rotAng[2], 0, 0, 1);
15          glCallList(ID_U);
16          glutSwapBuffers();
17  }
18
19  static void timer(int dummy) {
20          glutTimerFunc(100, timer, 0);
21          glMatrixMode(GL_MODELVIEW);
22
23          rotAng[0] += 30;
24          rotAng[1] += 10 * cos(time / 5);
25          rotAng[2] += 100 * cos(time);
26          time++;
27
28          glutPostRedisplay();
29  }
```

踊っているロケットを図 15 に示す.

図 15　踊っているロケット

4　感想

これまで学習してきた内容に比べて難しく感じる箇所が多い科目ではあったが，楽しく学習することができた．進学先では使う機会が少なくなりそうではあるが，個人的に学習を進めていきたいと思う．

総合課題は学園祭の演劇と時期が近いため，時間を効率的に使って制作を進めていきたい．

5　改善案

全体的に改善するところはあまり思いつかなかった．細かい部分で挙げるとすれば，ワールド座標系とデバイス座標系の変換イメージの図をテキストにも載せてあると良いと思った．また，サポートページにあるスライドに音声があると聞き逃した場所などの復習ができて良いと思った．

参考文献

[1] https://tokoik.github.io/GLFWdraft.pdf
[2] https://www.glfw.org/
[3] https://forest.watch.impress.co.jp/article/1999/07/13/glui.html
[4] https://www.dospara.co.jp/5info/cts_str_pc_vulkan
[5] https://www.khronos.org/opengles/
[6] https://wgld.org/
[7] https://docs.oracle.com/cd/E19957-01/806-4833/Make.html
[8] https://eng-entrance.com/linux-command-grep
[9] https://atmarkit.itmedia.co.jp/ait/articles/1606/14/news013.html
[10] https://cmake.org/
[11] https://subversion.apache.org/
[12] https://git-scm.com/
[13] http://ki-www.cvl.iis.u-tokyo.ac.jp/thesis/senior/inaguma.pdf
[14] 柴田望洋，新・明解 C 言語，SB クリエイティブ株式会社，2014
[15] 高橋章，R04-Ec5 プログラミング演習 II テキスト