R05プログラミング演習II学習内容自己管理シート

提出年月日： 令和5年10月27日（金）
出席番号： 35
氏名： 本間三暉

○学習時間

日付	学習時間	学習内容
9/11(月)	9	一日目の内容
9/12(火)	9	二日目の内容
9/13(水)	9	三日目の内容
9/14(木)	9	課題
合計	36	←30時間を越えること!!

○演習

第1日	演習No	1	2	3	4	5					
	実施	B	B	B	B	B	←実施状況や完成度/理解度を客				
第2日	演習No	6	7	8	9	10	11	12	13	14	15
	実施	B	B	B	B	B	B	B	B	B	B
第3日	演習No	16	17	18	19	20	21	22			
	実施	B	B	B	B	B	B	B			

○課題　　↓実施状況や完成度/理解度を自己評価

課題	No.	実施	実施内容および成果
課題I	1	B	情報を探すのに苦労したが，公式サイトを読んで理解した
	2	B	ぼんやりと知っている情報がより深く理解できた.
	3	B	受験のときにやった内容の復習ができてよかった.
	4	B	quick&dirtyなやり方で行った．もっとうまいやり方はあ
	5	B	4と同様に，もっとうまいやり方がありそうだが，実装し
	6	B	数学で習った内容があって嬉しくなった.
課題II	1	B	話として知っていた内容を計算して理解することができた
	2	B	計算ミスがひどかったが，計算方法は間違っていなかった
	3	B	完全に理解した.
	4	B	その都度アドリブで踊っている感じが可愛かった.
課題III	1	B	物理でやった内容を活かせた
	2	B	高専で習ったことを活かせた
	3	B	理解するのに時間がかかった
	4	B	苦労した
	5	B	科学技術英語の論文の内容と似た内容だと意識できた.
総合課題			簡単な絵と入力場所を見比べて，色合わせをするもの.

	D	C	B	A	S	計
演習	0	0	22	0	0	22
課題	0	0	15	0	0	15

1　課題 I

1.1　課題 I-1

OpenGL **関連技術**: 本テキストで扱う GLUT，GLU の他に OpenGL に関連するライブラリとして GLEW，GLFW，GLUI などがある．様々な実行環境で 2D/3DCG を実現するためのプログラミング技術として Vulkan，OpenGL ES，WebGL などがある．これらについて，機能や各ライブラリの関係性などを調査して整理してみよ．

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

プログラミングのツール:GCC に含まれる make ユーティリティ，デバッガの機能や機能や利用法について学習せよ．また，コマンドラインからのプログラムの開発では，grep や touch などのツールがあると便利である．更に最近ではクロスプラットフォーム開発を支援する CMake，バージョン管理を支援する Subversion や Git などを利用することも一般化している．これらの機能や利用法について学習せよ．

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**: 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) 変数とグローバル関数，スタティック変数の違いについて学習せよ．

1.3.1 有効範囲 (スコープ)

「有効範囲 (スコープ)」とは，変数などに与えられる識別子が通用する範囲のこと．

1.3.2 記憶域期間 (記憶寿命)

「記憶域期間 (記憶寿命)」には自動記憶域期間と静的記憶域期間がある．

自動記憶域期間とは auto を使用して宣言した変数が持つ記憶域期間のことで，オブジェクトは宣言したブロックに入ったときから終了するまで生存する．

静的記憶域期間とは，ファイル有効範囲のオブジェクトか static を使用して宣言したときにオブジェクトが持つ記憶域期間のことで，プログラムの開始から終了するまで生存する．

1.3.3 ローカル変数

main 関数など，宣言された関数内でのみ値を保持し，呼び出しできる変数．宣言された関数外で呼び出すことはできない．

1.3.4 グローバル変数

main 関数を代表とするような関数の外で宣言されたもの．プログラム内ならどこからでもアクセスできる．

1.3.5 スタティック変数

静的記憶域期間を持つ変数．プログラムが始まってから終わるまで値を保持し続ける．strtok 関数などに用いられていて，strtok 関数の扱いに注意が必要な原因である．

1.4 課題 I-4

星型正多角形を描き，回転させるプログラムを作成せよ．

1.4.1 プログラム

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．

Listing 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．

1.5 課題 I-5

完全グラフを描き，回転させるプログラムを作成せよ．

1.5.1 プログラム

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．

Listing 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;
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，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．

1.6 課題 I-6

リスト 12 を解析し，数学関数を描くプログラムを作成せよ．

1.6.1 カージオイド

カージオイドを描画するプログラムの主要部をソースコード 3 に示す．

Listing 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  }
```

1.6.2 サイクロイド

サイクロイドを描画するプログラムの主要部をソースコード 4 に示す.

Listing 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 }
```

1.6.3 4 尖点の内サイクロイド

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 5 に示す.

Listing 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 }
```

2　課題 II

2.1　課題 II-1

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，GBWindows では 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，拡張点の三次元座標を導出せよ．

2.1.1　導出

△BCD の高さは一辺が w の正三角形なので，$\sqrt{3}w/2$ となる．△GBD が二等辺三角形となる．三平方の定理を用いて △GBD の等辺の長さは $w/\sqrt{3}$ となる．これらのことから正四面体の高さは三平方の定理を用いて，$\sqrt{2/3}w$ となる．よって各頂点の位置は次となる．

$$\boldsymbol{a} = \left(\sqrt{\frac{2}{3}}w, 0, 0\right) \quad \boldsymbol{b} = \left(0, \frac{w}{\sqrt{3}}, 0\right) \quad \boldsymbol{c} = \left(0, -\frac{w}{2\sqrt{3}}, \frac{w}{2}\right) \quad \boldsymbol{d} = \left(0, -\frac{w}{2\sqrt{3}}, -\frac{w}{2}\right)$$

2.2　課題 II-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テキストのリスト 14 のように定まることを導出せよ．

2.2.1　導出

正四面体の重心 $\boldsymbol{g}'$ が原点と一致させるために前問の正四面体を移動させる．重心 $\boldsymbol{g}'$ は，

$$\boldsymbol{g}' = \frac{\boldsymbol{a}+\boldsymbol{b}+\boldsymbol{c}+\boldsymbol{d}}{4} = \left(\frac{1}{4}\sqrt{\frac{2}{3}}w, 0, 0\right)$$

移動後の各頂点を $\boldsymbol{a}'$，$\boldsymbol{b}'$，$\boldsymbol{c}'$，$\boldsymbol{d}'$ とすると各頂点の位置は，

$$\boldsymbol{a}' = \boldsymbol{a} - \boldsymbol{g}' = \left(\frac{1}{2}\sqrt{\frac{3}{2}}w, 0, 0\right) \qquad \boldsymbol{b}' = \boldsymbol{b} - \boldsymbol{g}' = \left(-\frac{1}{4}\sqrt{\frac{2}{3}}w, \frac{w}{\sqrt{3}}, 0\right)$$

$$\boldsymbol{c}' = \boldsymbol{c} - \boldsymbol{g}' = \left(-\frac{1}{4}\sqrt{\frac{2}{3}}w, -\frac{w}{2\sqrt{3}}, \frac{w}{2}\right) \quad \boldsymbol{d}' = \boldsymbol{d} - \boldsymbol{g}' = \left(-\frac{1}{4}\sqrt{\frac{2}{3}}w, -\frac{w}{2\sqrt{3}}, -\frac{w}{2}\right)$$

半径 1 の球に内接するとき $\boldsymbol{a}' = (1, 0, 0)$ となるので $w = 2\sqrt{2/3}$ となる．よって各頂点の位置は，

$$\boldsymbol{a}' = (1, 0, 0) \quad \boldsymbol{b}' = \left(-\frac{1}{3}, \frac{2\sqrt{2}}{3}, 0\right) \quad \boldsymbol{c}' = \left(-\frac{1}{3}, -\frac{\sqrt{2}}{3}, \frac{\sqrt{6}}{3}\right) \quad \boldsymbol{d}' = \left(-\frac{1}{3}, -\frac{\sqrt{2}}{3}, -\frac{\sqrt{6}}{3}\right)$$

この位置から各頂点の値がテキストのリスト 14 のように定まることがわかる．

2.3 課題 II-3

テキストの **9.2** 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.

2.3.1 プログラム

アームロボットの各パーツの登録部分を 6 に示す．また，キーボードコールバック関数を 7 に示す.

Listing 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 }
```

Listing 7: キーボードコールバック関数

```
void keyin(unsigned char key, int x, int y) {
    switch (key) {
    case 'x':
        rotAng[0] += SPEED;
        glutPostRedisplay();
        break;

    case 'z':
        rotAng[0] -= SPEED;
        glutPostRedisplay();
        break;

    case 's':
        rotAng[1] += SPEED;
        glutPostRedisplay();
        break;

    case 'a':
        rotAng[1] -= SPEED;
        glutPostRedisplay();
        break;

    case 'w':
        rotAng[2] += SPEED;
        glutPostRedisplay();
        break;

    case 'q':
        rotAng[2] -= SPEED;
        glutPostRedisplay();
        break;

    default: break;
    }
}
```

2.4 課題 II-4

タイマーコールバックを使って，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．

2.4.1 プログラム

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.

Listing 8: タイマコールバック関数

```
static void timer(int dummy) {
    glutTimerFunc(100, timer, 0);
    glMatrixMode(GL_MODELVIEW);

    rotAng[0] += 30;
    rotAng[1] += (rand() % 10) * cos(time / 5);
    rotAng[2] += (rand() % 100) * cos(time);
    time++;

    glutPostRedisplay();
}
```

3 課題 III

3.1 課題 III-1

空間中の任意の 3 点 $\mathrm{P}_i(x_i, y_i, z_i)$，$i = 1, 2, 3$ が与えられたとき，$\triangle \mathrm{P}_1\mathrm{P}_2\mathrm{P}_3$ の単位法線ベクトルをすべて求める式を導出せよ.

3.1.1 導出

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 $\mathrm{P}_i(x_i, y_i, z_i)$，$i = 1, 2, 3$ が与えられたとき，$\triangle \mathrm{P}_1\mathrm{P}_2\mathrm{P}_3$ 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.

3.2.1 導出

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 の照光モデルについて，詳しく調べよ．

3.3.1 結果

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

3.4.1 プログラム

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．

Listing 9: 正四面体を描画するプログラム

```
1  GLdouble vP
       [4][3]={{1.000,0.000,0.000},{-0.333,0.943,0.000}
2  ,{-0.333,-0.471,0.816},{-0.333,-0.471,-0.816}};
3  int tP[4][3]={{0,1,2},{0,3,1},{0,2,3},{1,3,2}};
4
5  void display(void){
6    int i,j;
7    glClear(GL_COLOR_BUFFER_BIT|GL_DEPTH_BUFFER_BIT);
8    glMatrixMode(GL_MODELVIEW);
9
10   glBegin(GL_TRIANGLES);
11   for(i=0;i<4;i++){
12     for(j=0;j<3;j++){
13       glNormal3dv(vP[tP[i][j]]);
14       glVertex3dv(vP[tP[i][j]]);
15     }
16   }
17   glEnd();
18   glutPostRedisplay();
19   glutSwapBuffers();
20
21 }
```

Listing 10: 正六面体を描画するプログラム

```
1  GLdouble vP
       [8][3]={{-0.577,-0.577,0.577},{0.577,-0.577,0.577}
2  ,{0.577,-0.577,-0.577},{-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},{-0.577,0.577,-0.577}};

4  int tP
       [6][4]={{3,2,1,0},{0,1,5,4},{1,2,6,5},{4,5,6,7},{3,7,6,2},{0,4,7,3}};

5
6
7  void display(void){
8    int i,j;
9    glClear(GL_COLOR_BUFFER_BIT|GL_DEPTH_BUFFER_BIT);
10   glMatrixMode(GL_MODELVIEW);
11
12   glBegin(GL_QUADS);
13   for(i=0;i<6;i++){
14     for(j=0;j<4;j++){
15       glNormal3dv(vP[tP[i][j]]);
```

```
16        glVertex3dv(vP[tP[i][j]]);
17      }
18    }
19    glEnd();
20    glutPostRedisplay();
21    glutSwapBuffers();
22
23  }
```

Listing 11: 正八面体を描画するプログラム

```
1   GLdouble vP
        [6][3]={{1,0,0},{0,1,0},{0,0,1},{-1,0,0},{0,-1,0},{0,0,-1}};

2   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}};

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

vP に頂点の位置，tP にトポロジ情報を格納している．正六面体，正八面体の頂点の位置，トポロジ情報は，演習 19 で求めたものである．法線ベクトルの指定には，vP をそのまま利用している。

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．

Listing 12: リスト 27 の変更部分

```
1  GLdouble rotAng[3] = { 0 };
2
3  void display(void) {
4          glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
5          glMatrixMode(GL_MODELVIEW);
6          glLoadIdentity();
7          gluLookAt(1.0, 1.0, 1.0, 0.0, 0.5, 0.0, 0.0, 1.0,
               0.0);
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] += (rand() % 10) * cos(time / 5);
25         rotAng[2] += (rand() % 100) * cos(time);
26         time++;
27
28         glutPostRedisplay();
29 }
```

4 感想

今まで学習していた内容もしっかり扱った上で，更に進んだ内容を学べれて良かったと感じる．色々なことが重なってしまったことでこの科目のみに集中できる時間をあまり取れなかったことが悔やまれる．総合制作はこのレポートとは違い，ギリギリで低品質なものにならない様に進めていきたい．

5　改善案

科目名から授業で扱う内容がなんとなくわかると良いと思った.「プログラミング演習 II」ではプログラミングをすることしか分からず，ややもったいないと感じる.

参考文献

1. 高橋章,R05-Ec5 プログラミング演習 II テキスト
2. https://tokoik.github.io/GLFWdraft.pdf
3. https://www.glfw.org/
4. https://forest.watch.impress.co.jp/article/1999/07/13/glui.html
5. https://www.dospara.co.jp/5info/cts_str_pc_vulkan
6. https://www.khronos.org/opengles/
7. https://wgld.org/
8. https://docs.oracle.com/cd/E19957-01/806-4833/Make.html
9. https://eng-entrance.com/linux-command-grep
10. https://atmarkit.itmedia.co.jp/ait/articles/1606/14/news013.html
11. https://cmake.org/
12. https://subversion.apache.org/
13. https://git-scm.com/
14. http://ki-www.cvl.iis.u-tokyo.ac.jp/thesis/senior/inaguma.pdf