CG基礎 課題2

# 動作環境の説明

* OS  
  Windows10 Pro 内の Bash on Ubuntu on Windows  
  (4.4.0-43-Microsoft #1-Microsoft Wed Dec 31 14:42:53 PST 2014)
* コンパイル  
  g++ (Ubuntu 5.4.0-6ubuntu1~16.04.4) 5.4.0 20160609  
  Copyright (C) 2015 Free Software Foundation, Inc.
* コーディング  
  Spacemacs 0.200.9 (Emacs24.5.1)

# 課題１

授業用 Web ページにある、それぞれのサンプルコードを実行し、プログラムコードと実行結果の様子を観察しなさい。プログラムの原理を理解するためには、コード内の数値を変更して結果を確認するとよい。  
サンプルコードについてはこの部分の最後にまとめて記載する。

1. 01\_triangle\_rotation.cpp  
   
2. 02\_display\_list\_example.cpp  
   
3. 03\_star\_rotation.cpp   
   

## サンプルコード

1. 01\_triangle\_rotation.cpp

#include <math.h>

#include <GL/glut.h>

int rotateAngle;

void display(void) {

glClearColor(1.0, 1.0, 1.0, 1.0);

glClear(GL\_COLOR\_BUFFER\_BIT);

glMatrixMode(GL\_MODELVIEW);

glLoadIdentity();

glRotated(rotateAngle, 0, 0, 1);

glColor3d(1.0, 0.0, 0.0);

glBegin(GL\_TRIANGLES);

glVertex3d(0, 0, 0);

glVertex3d(0.5, 0, 0);

glVertex3d(0.5, 0.5, 0);

glEnd();

glutSwapBuffers();

}

void timer(int value){

rotateAngle++;

glutPostRedisplay();

glutTimerFunc(100, timer, 0);

}

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

{

glutInit(&argc, argv);

glutInitDisplayMode(GLUT\_RGBA|GLUT\_DOUBLE);

glutInitWindowSize(400, 400);

glutCreateWindow(argv[0]);

glutDisplayFunc(display);

glutTimerFunc(100, timer, 0);

rotateAngle = 0;

glutMainLoop();

return 0;

}

1. 02\_display\_list\_example.cpp

#include <math.h>

#include <GL/glut.h> // ライブラリ用ヘッダファイルの読み込み

// 定数πの定義

#ifndef M\_PI

#define M\_PI 3.14159265358979

#endif

// ディスプレイリストの学習

// 星を描画する描画命令一式を、ディスプレイリストとして作成しておき

// 必要な時に、その命令を呼び出す

#define ID\_DRAW\_STAR 1 // glNewList 関数で使用する識別ID。値は何でも構わない

// 表示部分をこの関数で記入

void display(void) {

glClearColor (1.0, 1.0, 1.0, 1.0); // 消去色指定

glClear(GL\_COLOR\_BUFFER\_BIT); // 画面と奥行き情報を初期化

glMatrixMode(GL\_MODELVIEW);

glLoadIdentity();

glColor3d(1.0, 0.0, 0.0);

glTranslated(0.5, 0, 0);

glCallList(ID\_DRAW\_STAR);

glLoadIdentity();

glColor3d(0.0, 1.0, 0.0);

glTranslated(0, 0.5, 0);

glCallList(ID\_DRAW\_STAR);

glLoadIdentity();

glColor3d(0.0, 0.0, 1.0);

glTranslated(-0.5, 0, 0);

glCallList(ID\_DRAW\_STAR);

glutSwapBuffers(); // バッファの入れ替え

}

// 一定時間ごとに呼び出される関数

void timer(int value) {

glutPostRedisplay(); // 再描画命令

glutTimerFunc(100 , timer , 0); // 100ミリ秒後に自身を実行する

}

// ディスプレイリストを作成する

void buildDisplayList() {

glNewList(ID\_DRAW\_STAR,GL\_COMPILE);

double r0 = 0.15; // 星の内径

double r1 = 0.4; // 星の外径

glBegin(GL\_TRIANGLES);

for(int i = 0; i < 5; i++) { // 5つの三角形で星を表現する

int deg = i \* 72;

glVertex3d(r0 \* cos( (deg - 36) \* M\_PI / 180.0), r0 \* sin( (deg - 36) \* M\_PI / 180.0), 0); // 内側の頂点

glVertex3d(r1 \* cos( deg \* M\_PI / 180.0), r1 \* sin( deg \* M\_PI / 180.0), 0); // 外側の頂点

glVertex3d(r0 \* cos( (deg + 36) \* M\_PI / 180.0), r0 \* sin( (deg + 36) \* M\_PI / 180.0) ,0); // 内側の頂点

}

glEnd();

glEndList();

}

// メインプログラム

int main (int argc, char \*argv[]) {

glutInit(&argc, argv); // ライブラリの初期化

glutInitDisplayMode(GLUT\_RGBA|GLUT\_DOUBLE);

glutInitWindowSize(400 , 400); // ウィンドウサイズを指定

glutCreateWindow(argv[0]); // ウィンドウを作成

glutDisplayFunc(display); // 表示関数を指定

glutTimerFunc(100 , timer , 0); // 100ミリ秒後に実行する関数の指定

buildDisplayList();

glutMainLoop(); // イベント待ち

return 0;

}

1. 03\_star\_rotation.cpp

#include <math.h>

#include <GL/glut.h> // ライブラリ用ヘッダファイルの読み込み

// 定数πの定義

#ifndef M\_PI

#define M\_PI 3.14159265358979

#endif

// ディスプレイリストの学習

// 星を描画する描画命令一式を、ディスプレイリストとして作成しておき

// 必要な時に、その命令を呼び出す

#define ID\_DRAW\_STAR 1 // glNewList 関数で使用する識別ID。値は何でも構わない

int rotateAngle; // 回転角度を記録しておく変数

// 表示部分をこの関数で記入

void display(void) {

glClearColor (1.0, 1.0, 1.0, 1.0); // 消去色指定

glClear(GL\_COLOR\_BUFFER\_BIT); // 画面と奥行き情報を初期化

glMatrixMode(GL\_MODELVIEW);

glLoadIdentity();

glPushMatrix();

glColor3d(1.0, 0.0, 0.0);

glTranslated(0.5, 0, 0);

glRotated(rotateAngle, 0, 0, 1);

glCallList(ID\_DRAW\_STAR);

glPopMatrix();

glPushMatrix();

glColor3d(0.0, 1.0, 0.0);

glTranslated(0, 0.5, 0);

glCallList(ID\_DRAW\_STAR);

glPopMatrix();

glPushMatrix();

glColor3d(0.0, 0.0, 1.0);

glTranslated(-0.5, 0, 0);

glCallList(ID\_DRAW\_STAR);

glPopMatrix();

glutSwapBuffers(); // バッファの入れ替え

}

// 一定時間ごとに呼び出される関数

void timer(int value) {

rotateAngle++; // 回転角度の更新

glutPostRedisplay(); // 再描画命令

glutTimerFunc(100 , timer , 0); // 100ミリ秒後に自身を実行する

}

// ディスプレイリストを作成する

void buildDisplayList() {

glNewList(ID\_DRAW\_STAR,GL\_COMPILE);

double r0 = 0.15; // 星の内径

double r1 = 0.4; // 星の外径

glBegin(GL\_TRIANGLES);

for(int i = 0; i < 5; i++) { // 5つの三角形で星を表現する

int deg = i \* 72;

glVertex3d(r0 \* cos( (deg - 36) \* M\_PI / 180.0), r0 \* sin( (deg - 36) \* M\_PI / 180.0), 0); // 内側の頂点

glVertex3d(r1 \* cos( deg \* M\_PI / 180.0), r1 \* sin( deg \* M\_PI / 180.0), 0); // 外側の頂点

glVertex3d(r0 \* cos( (deg + 36) \* M\_PI / 180.0), r0 \* sin( (deg + 36) \* M\_PI / 180.0) ,0); // 内側の頂点

}

glEnd();

glEndList();

}

// メインプログラム

int main (int argc, char \*argv[]) {

glutInit(&argc, argv); // ライブラリの初期化

glutInitDisplayMode(GLUT\_RGBA|GLUT\_DOUBLE);

glutInitWindowSize(400 , 400); // ウィンドウサイズを指定

glutCreateWindow(argv[0]); // ウィンドウを作成

glutDisplayFunc(display); // 表示関数を指定

glutTimerFunc(100 , timer , 0); // 100ミリ秒後に実行する関数の指定

buildDisplayList();

rotateAngle = 0; // 変数の初期値の設定

glutMainLoop(); // イベント待ち

return 0;

}

# 課題2

　03\_star\_rotation.cpp を実行すると、赤い星だけがその場で回転する。このプログラムコードを改変し、3 つの星すべてが赤い星と同様にそれぞれが回転するようにし、さらに全ての星が原点（画面の中心）のまわりを回るようにしなさい（地球が自転しながら、太陽の周りを回っているイメージ）。  


## コード

#include <math.h>

#include <GL/glut.h>

#ifndef M\_PI

#define M\_PI 3.14159265358979

#endif

#define ID\_DRAW\_STAR 1

#define ID\_DRAW\_STAR2 2

int rotateAngle;

void display(void){

glClearColor (1.0, 1.0, 1.0, 1.0);

glClear(GL\_COLOR\_BUFFER\_BIT);

glMatrixMode(GL\_MODELVIEW);

glLoadIdentity();

// \*

glPushMatrix();

glTranslated(0, 0, 0);

glRotated(rotateAngle, 0, 0, 1);

//1

glPushMatrix();

glColor3d(1.0, 0.0, 0.0);

glTranslated(0.5, 0, 0);

glRotated(rotateAngle, 0, 0, 1);

glCallList(ID\_DRAW\_STAR);

glPopMatrix();

//2

glPushMatrix();

glColor3d(0.0, 1.0, 0.0);

glTranslated(0, 0.5, 0);

glRotated(rotateAngle, 0, 0, 1);

glCallList(ID\_DRAW\_STAR);

glPopMatrix();

//3

glPushMatrix();

glColor3d(0.0, 0.0, 1.0);

glTranslated(-0.5, 0, 0);

glRotated(rotateAngle, 0, 0, 1);

glCallList(ID\_DRAW\_STAR);

glPopMatrix();

glPopMatrix();

glutSwapBuffers();

}

void timer(int value) {

rotateAngle++;

glutPostRedisplay();

glutTimerFunc(100, timer, 0);

}

void buildDisplayList() {

glNewList(ID\_DRAW\_STAR,GL\_COMPILE);

double r0 = 0.15;

double r1 = 0.4;

glBegin(GL\_TRIANGLES);

for(int i = 0; i < 5; i++) {

int deg = i \* 72;

glVertex3d(r0 \* cos( (deg - 36) \* M\_PI / 180.0), r0 \* sin( (deg - 36) \* M\_PI / 180.0), 0);

glVertex3d(r1 \* cos( deg \* M\_PI / 180.0), r1 \* sin( deg \* M\_PI / 180.0), 0);

glVertex3d(r0 \* cos( (deg + 36) \* M\_PI / 180.0), r0 \* sin( (deg + 36) \* M\_PI / 180.0) ,0);

}

glEnd();

glEndList();

glNewList(ID\_DRAW\_STAR2,GL\_COMPILE);

//double r0 = 0.15;

//double r1 = 0.4;

glBegin(GL\_TRIANGLES);

for(int i = 0; i < 5; i++) {

int deg = i \* 72;

glVertex3d(r0 \* cos( (deg - 36) \* M\_PI / 180.0), r0 \* sin( (deg - 36) \* M\_PI / 180.0), 0);

glVertex3d(r1 \* cos( deg \* M\_PI / 180.0), r1 \* sin( deg \* M\_PI / 180.0), 0);

glVertex3d(r0 \* cos( (deg + 36) \* M\_PI / 180.0), r0 \* sin( (deg + 36) \* M\_PI / 180.0) ,0);

}

glEnd();

glEndList();

}

int main (int argc, char \*argv[]) {

glutInit(&argc, argv);

glutInitDisplayMode(GLUT\_RGBA|GLUT\_DOUBLE);

glutInitWindowSize(400 , 400);

glutCreateWindow(argv[0]);

glutDisplayFunc(display);

glutTimerFunc(100 , timer , 0);

buildDisplayList();

rotateAngle = 0;

glutMainLoop();

return 0;

}

# 課題3

上記の課題 3 に加え、小さな黒い星が赤い星の周りをクルクル素早く回っている様子を描くようにしなさい（赤い星の図形で示された地球の周りを月が回っているイメージ）。  


## コード

#include <math.h>

#include <GL/glut.h>

#ifndef M\_PI

#define M\_PI 3.14159265358979

#endif

#define ID\_DRAW\_STAR 1

#define ID\_DRAW\_STAR2 2

int rotateAngle;

void display(void){

glClearColor (1.0, 1.0, 1.0, 1.0);

glClear(GL\_COLOR\_BUFFER\_BIT);

glMatrixMode(GL\_MODELVIEW);

glLoadIdentity();

// \*

glPushMatrix();

glTranslated(0, 0, 0);

glRotated(rotateAngle, 0, 0, 1);

//1

glPushMatrix();

glColor3d(1.0, 0.0, 0.0);

glTranslated(0.5, 0, 0);

glRotated(rotateAngle, 0, 0, 1);

glCallList(ID\_DRAW\_STAR);

glPopMatrix();

//0.5 ... small star

glPushMatrix();

glColor3d(0.0, 0.0, 0.0);

glTranslated(0.5, 0, 0);

glRotated(rotateAngle \* 20, 0, 0, 1);

glTranslated(0.4,0,0);

glRotated(rotateAngle \* 3, 0, 0, 1);

glCallList(ID\_DRAW\_STAR2);

glPopMatrix();

//2

glPushMatrix();

glColor3d(0.0, 1.0, 0.0);

glTranslated(0, 0.5, 0);

glRotated(rotateAngle, 0, 0, 1);

glCallList(ID\_DRAW\_STAR);

glPopMatrix();

//3

glPushMatrix();

glColor3d(0.0, 0.0, 1.0);

glTranslated(-0.5, 0, 0);

glRotated(rotateAngle, 0, 0, 1);

glCallList(ID\_DRAW\_STAR);

glPopMatrix();

glPopMatrix();

glutSwapBuffers();

}

void timer(int value) {

rotateAngle++;

glutPostRedisplay();

glutTimerFunc(100, timer, 0);

}

void buildDisplayList() {

glNewList(ID\_DRAW\_STAR,GL\_COMPILE);

double r0 = 0.15;

double r1 = 0.4;

glBegin(GL\_TRIANGLES);

for(int i = 0; i < 5; i++) {

int deg = i \* 72;

glVertex3d(r0 \* cos( (deg - 36) \* M\_PI / 180.0), r0 \* sin( (deg - 36) \* M\_PI / 180.0), 0);

glVertex3d(r1 \* cos( deg \* M\_PI / 180.0), r1 \* sin( deg \* M\_PI / 180.0), 0);

glVertex3d(r0 \* cos( (deg + 36) \* M\_PI / 180.0), r0 \* sin( (deg + 36) \* M\_PI / 180.0) ,0);

}

glEnd();

glEndList();

glNewList(ID\_DRAW\_STAR2,GL\_COMPILE);

//double r0 = 0.15;

//double r1 = 0.4;

r0 = 0.05;

r1 = 0.1;

double r2 = 0.2;

glBegin(GL\_TRIANGLES);

for(int i = 0; i < 5; i++) {

int deg = i \* 72;

glVertex3d(r0 \* cos( (deg - 36) \* M\_PI / 180.0), r0 \* sin( (deg - 36) \* M\_PI / 180.0), 0);

glVertex3d(r1 \* cos( deg \* M\_PI / 180.0), r1 \* sin( deg \* M\_PI / 180.0), 0);

glVertex3d(r0 \* cos( (deg + 36) \* M\_PI / 180.0), r0 \* sin( (deg + 36) \* M\_PI / 180.0) ,0);

}

glEnd();

glEndList();

}

int main (int argc, char \*argv[]) {

glutInit(&argc, argv);

glutInitDisplayMode(GLUT\_RGBA|GLUT\_DOUBLE);

glutInitWindowSize(400, 400);

glutCreateWindow(argv[0]);

glutDisplayFunc(display);

glutTimerFunc(100 , timer , 0);

buildDisplayList();

rotateAngle = 0;

glutMainLoop();

return 0;

}

# 課題4

　次の条件を満たすような、2 次元図形を画面に表示するプログラムを作成しなさい（できるだけ綺麗な、または楽しい図形を描こう）。  


## コード

#include <math.h>

#include <GL/glut.h>

#ifndef M\_PI

#define M\_PI 3.14159265358979

#endif

#define ID\_DRAW\_BALL 1

#define ID\_DRAW\_BALL2 2

#define ID\_DRAW\_BALL3 3

#define ID\_DRAW\_BOX 4

int rotateAngle;

float r;

void display(void){

// init

glClearColor (1.0, 1.0, 1.0, 1.0);

glClear(GL\_COLOR\_BUFFER\_BIT);

glMatrixMode(GL\_MODELVIEW);

glLoadIdentity();

glPushMatrix();

glColor3d(1.0, 0.4, 0.4);

glRotated(rotateAngle \* 20, 0, 0, 1);

glTranslated(r, 0, 0);

glCallList(ID\_DRAW\_BALL);

glPopMatrix();

glPushMatrix();

glColor3d(0.4, 1.0, 0.4);

glRotated(rotateAngle \* 20, 0, 0, 1);

glTranslated(r/sqrt(2), r/sqrt(2), 0);

glCallList(ID\_DRAW\_BALL2);

glPopMatrix();

glPushMatrix();

glColor3d(0.4, 0.4, 1.0);

glRotated(rotateAngle \* 20, 0, 0, 1);

glTranslated(0, r, 0);

glCallList(ID\_DRAW\_BALL3);

glPopMatrix();

glPushMatrix();

glColor3d(0.7, 0.7, 0.7);

glRotated(rotateAngle \* 20, 0, 0, 1);

glCallList(ID\_DRAW\_BOX);

glPopMatrix();

glutSwapBuffers();

}

void timer(int value) {

rotateAngle++;

r += 0.01;

glutPostRedisplay();

glutTimerFunc(100, timer, 0);

}

void buildCircle(float r2) {

int i, n = 200;

double x, y = 0.5;

glBegin(GL\_POLYGON);

for (i = 0; i < n; ++i) {

x = r2 \* cos(2.0 \* 3.14 \* ((double)i/n));

y = r2 \* sin(2.0 \* 3.14 \* ((double)i/n));

glVertex3f(x, y, 0.0);

}

glEnd();

}

void buildDisplayList() {

glNewList(ID\_DRAW\_BALL, GL\_COMPILE);

buildCircle(0.1);

glEndList();

glNewList(ID\_DRAW\_BALL2, GL\_COMPILE);

buildCircle(0.15);

glEndList();

glNewList(ID\_DRAW\_BALL3, GL\_COMPILE);

buildCircle(0.2);

glEndList();

glNewList(ID\_DRAW\_BOX, GL\_COMPILE);

glBegin(GL\_QUADS);

glVertex2d(0.1, 0.1);

glVertex2d(0.1, -0.1);

glVertex2d(-0.1, -0.1);

glVertex2d(-0.1, 0.1);

glEnd();

glEndList();

}

int main(int argc, char \*argv[]) {

glutInit(&argc, argv);

glutInitDisplayMode(GLUT\_RGBA|GLUT\_DOUBLE);

glutInitWindowSize(400, 400);

glutCreateWindow(argv[0]);

glutDisplayFunc(display);

glutTimerFunc(100, timer, 0);

buildDisplayList();

rotateAngle = 0;

glutMainLoop();

return 0;

}

# 感想

　実験の教室の人口密度が適度になっていて良いと思いました