学籍番号20211351

所属 情報学群情報メディア創成学類 3年

氏名 高橋 健太郎

# 課題2

#include <math.h>

#include <GLUT/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 \* cos((rotateAngle + 0) \* (M\_PI / 180.0))), (0.5 \* sin((rotateAngle + 0) \* (M\_PI / 180.0))), 0);

glRotated(rotateAngle, 0, 0, 1);

glCallList(ID\_DRAW\_STAR);

glPopMatrix();

glPushMatrix();

glColor3d(0.0, 1.0, 0.0);

glTranslated((0.5 \* cos((rotateAngle + 90) \* (M\_PI / 180.0))), (0.5 \* sin((rotateAngle + 90) \* (M\_PI / 180.0))), 0);

glRotated(rotateAngle, 0, 0, 1);

glCallList(ID\_DRAW\_STAR);

glPopMatrix();

glPushMatrix();

glColor3d(0.0, 0.0, 1.0);

glTranslated((0.5 \* cos((rotateAngle + 180) \* (M\_PI / 180.0))), (0.5 \* sin((rotateAngle + 180) \* (M\_PI / 180.0))), 0);

glRotated(rotateAngle, 0, 0, 1);

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(10, 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*

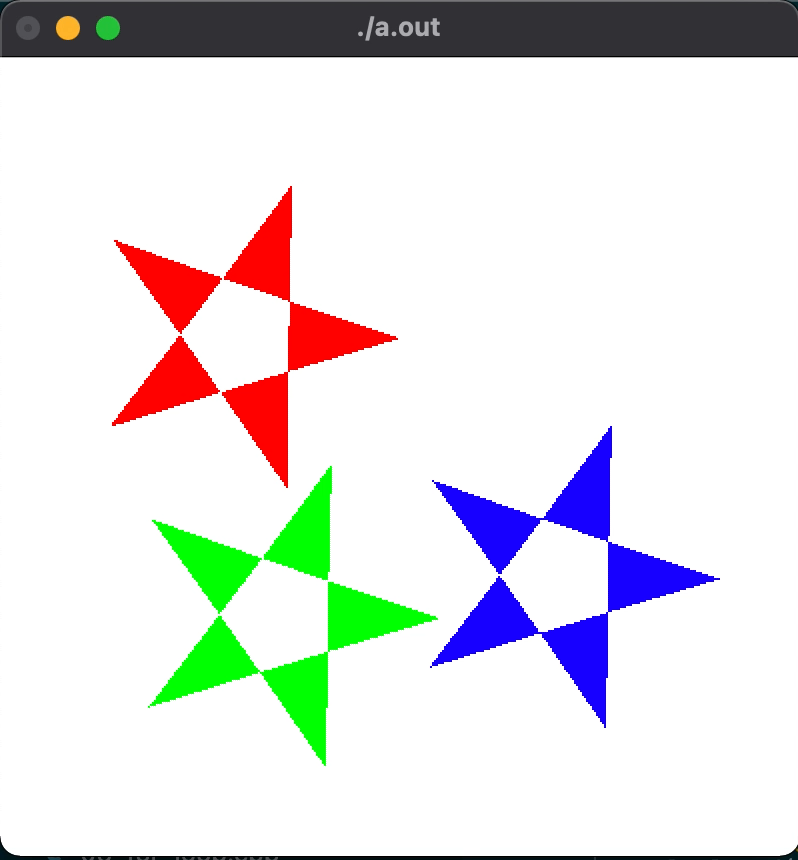
buildDisplayList();

rotateAngle = 0; *// 変数の初期値の設定*

glutMainLoop(); *// イベント待ち*

return 0;

}



# 課題3

#include <math.h>

#include <GLUT/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 \* cos((rotateAngle + 0) \* (M\_PI / 180.0))), (0.5 \* sin((rotateAngle + 0) \* (M\_PI / 180.0))), 0);

glRotated(rotateAngle, 0, 0, 1);

glCallList(ID\_DRAW\_STAR);

glPopMatrix();

glPushMatrix();

glColor3d(0.0, 1.0, 0.0);

glTranslated((0.5 \* cos((rotateAngle + 90) \* (M\_PI / 180.0))), (0.5 \* sin((rotateAngle + 90) \* (M\_PI / 180.0))), 0);

glRotated(rotateAngle, 0, 0, 1);

glCallList(ID\_DRAW\_STAR);

glPopMatrix();

glPushMatrix();

glColor3d(0.0, 0.0, 1.0);

glTranslated((0.5 \* cos((rotateAngle + 180) \* (M\_PI / 180.0))), (0.5 \* sin((rotateAngle + 180) \* (M\_PI / 180.0))), 0);

glRotated(rotateAngle, 0, 0, 1);

glCallList(ID\_DRAW\_STAR);

glPopMatrix();

glPushMatrix();

glColor3d(0.0, 0.0, 0.0);

glTranslated((0.5 \* cos((rotateAngle + 0) \* (M\_PI / 180.0))) + (0.5 \* cos((rotateAngle \* 10) \* (M\_PI / 180.0))), (0.5 \* sin((rotateAngle + 0) \* (M\_PI / 180.0))) + (0.5 \* sin((rotateAngle \* 10) \* (M\_PI / 180.0))), 0);

glRotated(rotateAngle, 0, 0, 1);

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(10, 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*

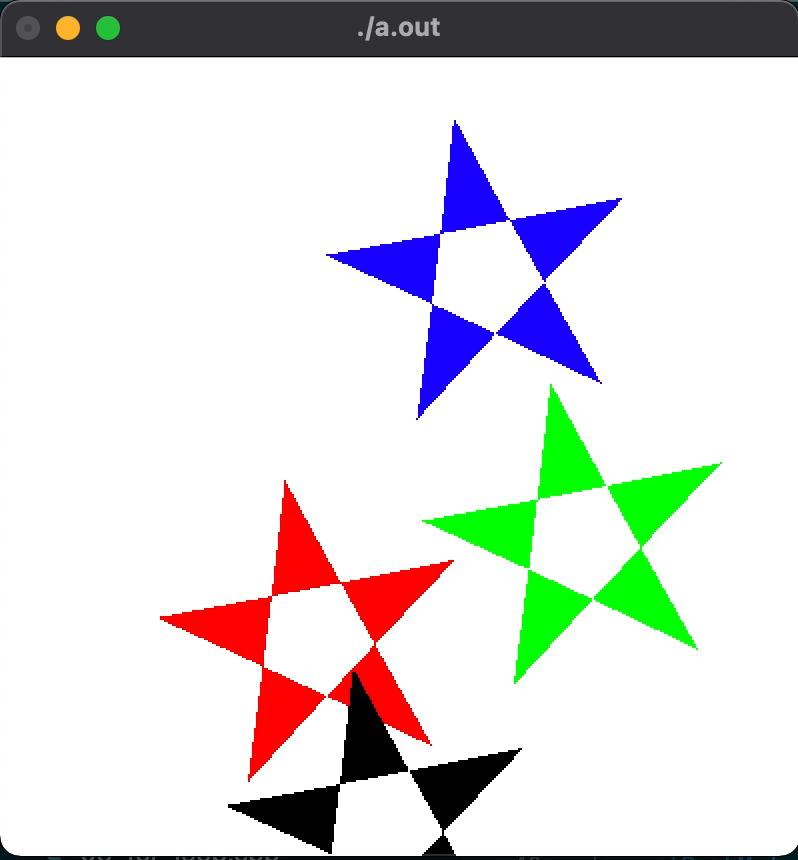
buildDisplayList();

rotateAngle = 0; *// 変数の初期値の設定*

glutMainLoop(); *// イベント待ち*

return 0;

}



# 課題4

#include <math.h>

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

**int** inflateX;

**int** inflateY;

**void** display(**void**)

{

glClearColor(1.0, 1.0, 1.0, 1.0); *//消去色*

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, 0);

glScaled(inflateX \* 0.01, inflateY \* 0.01, 1.0);

glCallList(1);

glPopMatrix();

glPushMatrix();

glColor3d(0.0, 1.0, 0.0);

glTranslated(0.5, 0.0, 0);

glScaled(inflateX \* 0.01, inflateY \* 0.01, 1.0);

glCallList(1);

glPopMatrix();

glutSwapBuffers();

}

**void** timer(**int** value)

{

if (inflateX <= 50)

{

inflateX++;

}

if (inflateY <= 100)

{

inflateY++;

}

glutPostRedisplay();

glutTimerFunc(10, timer, 0);

}

**void** buildDisplayList()

{

glNewList(1, GL\_COMPILE);

glBegin(GL\_QUADS);

glVertex2d(-1.0, 1.0);

glVertex2d(-1.0, -1.0);

glVertex2d(1.0, -1.0);

glVertex2d(1.0, 1.0);

glEnd();

glEndList();

}

**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);

glutCreateWindow(argv[0]);

glutDisplayFunc(display);

glutTimerFunc(100, timer, 0);

buildDisplayList();

inflateX = 1;

inflateY = 1;

glutMainLoop();

return 0;

}

グラフィカル ユーザー インターフェイス

自動的に生成された説明