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

#define \_USE\_MATH\_DEFINES *// Visual Studio で 円周率として定数 M\_PI を使用するため*

#include <cstdlib>

#include <GLUT/glut.h>

#include <cmath>

*// 色の情報を格納する共用体*

**union** color

{

**struct**

{

**float** r, g, b, a;

};

**float** colors[4];

};

*// 個々のティーポットの色や、傾き角度に関する情報を保持するための構造体*

**struct** TeapotData

{

color ambient, diffuse, specular;

**float** shininess, angle;

};

*// グローバル変数（プログラム中のどこからでもアクセスできる変数）には g\_ を付けている*

**const** **int** g\_NumTeapots = 8;

TeapotData g\_Teapots[g\_NumTeapots];

*// float型の値は、数字の後ろにfを付ける。末尾のゼロは省略できる*

**const** **float** g\_TeapotSize = 1.f;

**const** **float** g\_InnerRadius = 6.f;

**const** **float** g\_OuterRadius = 7.5f;

**const** **float** g\_HeightAmplitude = 0.8f;

**const** **float** g\_HeightOffset = 0.2f;

**const** **float** g\_EyeCenterY = 9.f;

**const** **float** g\_EyeCenterZ = 30.f;

**const** **float** g\_EyeRadius = 8.f;

**float** g\_EyeY, g\_EyeZ;

**const** **int** g\_AnimationIntervalMsec = 10;

**float** g\_RotationDegree = 0.f;

**const** **float** g\_DeltaRotationDegree = 0.3;

**int** g\_WindowWidth = 512;

**int** g\_WindowHeight = 512;

*// 円筒を描画…引数は円の半径、高さ、円の分割数*

*// glutには円筒を描画するための関数が無いので、独自に準備*

**void** displayCylinder(**float** radius, **float** height, **int** nSlices)

{

*// 天頂面*

**const** **float** deltaTheta = 2 \* M\_PI / (**float**)nSlices;

glNormal3f(0, 1, 0);

glBegin(GL\_TRIANGLE\_FAN);

glVertex3f(0, height, 0);

for (**int** i = 0; i <= nSlices; i++)

{

**const** **float** theta = deltaTheta \* i;

glVertex3f(radius \* cosf(theta), height, radius \* sinf(theta));

}

glEnd();

*// 底面*

glNormal3f(0, -1, 0);

glBegin(GL\_TRIANGLE\_FAN);

glVertex3f(0, 0, 0);

for (**int** i = 0; i <= nSlices; i++)

{

**const** **float** theta = deltaTheta \* i;

glVertex3f(radius \* cosf(theta), 0, radius \* sinf(theta));

}

glEnd();

*// 側面*

glBegin(GL\_TRIANGLE\_STRIP);

for (**int** i = 0; i <= nSlices; i++)

{

**const** **float** theta = deltaTheta \* i;

**const** **float** cosTheta = cosf(theta);

**const** **float** sinTheta = sinf(theta);

glNormal3f(cosTheta, 0, sinTheta);

glVertex3f(radius \* cosTheta, height, radius \* sinTheta);

glVertex3f(radius \* cosTheta, 0, radius \* sinTheta);

}

glEnd();

}

**void** display()

{

glClear(GL\_COLOR\_BUFFER\_BIT | GL\_DEPTH\_BUFFER\_BIT);

*// 透視投影変換の設定*

glMatrixMode(GL\_PROJECTION);

glLoadIdentity();

gluPerspective(30.0, g\_WindowWidth / (**double**)g\_WindowHeight, 1, 100.0);

*// モデル座標の操作へモード切り替え*

glMatrixMode(GL\_MODELVIEW);

glLoadIdentity();

gluLookAt(0.0, g\_EyeY, g\_EyeZ, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0);

**float** ambientColor[] = {0.4f, 0.2f, 0.2f, 1.0f};

**float** diffuseColor[] = {1.f, 0.8f, 0.8f, 1.0f};

**float** specularColor[] = {0.4f, 0.3f, 0.3f, 1.0f};

**float** shininess = 5.f;

glMaterialfv(GL\_FRONT, GL\_AMBIENT, ambientColor);

glMaterialfv(GL\_FRONT, GL\_DIFFUSE, diffuseColor);

glMaterialfv(GL\_FRONT, GL\_SPECULAR, specularColor);

glMaterialfv(GL\_FRONT, GL\_SHININESS, &shininess);

*// 屋根*

glPushMatrix();

glTranslatef(0, g\_HeightAmplitude + g\_HeightOffset + 3.f, 0);

glRotatef(-90, 1, 0, 0);

**glRotatef(g\_RotationDegree, 0, 0, 1);** *// 回転させている*

glutSolidCone(g\_OuterRadius, 2.f, 32, 32);

glPopMatrix();

*// 中心の柱*

glPushMatrix();

glTranslatef(0, -1.f, 0);

**glRotatef(g\_RotationDegree, 0, 1, 0);** *// 回転させている*

displayCylinder(0.5f, g\_HeightAmplitude + g\_HeightOffset + 6.5f, 32);

glPopMatrix();

*// 土台*

glPushMatrix();

glTranslatef(0, -2.f, 0);

**glRotatef(g\_RotationDegree, 0, 1, 0);** *// 回転させている*

displayCylinder(g\_OuterRadius, 0.7f, 64);

glPopMatrix();

*// 屋根の上のティーポット*

glPushMatrix();

glTranslatef(0, g\_HeightAmplitude + g\_HeightOffset + 5.5f, 0);

**glRotatef(g\_RotationDegree, 0, 1, 0);** *// 回転させている*

glutSolidTeapot(g\_TeapotSize);

glPopMatrix();

**const** **float** deltaTheta = 360 / (**float**)g\_NumTeapots;

*// ティーポットと柱を1つずつ描画する*

*// ★下記のコードでは、常に同じ位置に描画されるので、全体が回転するように変更する*

for (**int** i = 0; i < g\_NumTeapots; i++)

{

**const** **float** thetaDegree = deltaTheta \* i + g\_RotationDegree; *// ティーポットの位置を決めるための角度*

**const** **float** thetaRad = thetaDegree \* M\_PI / 180.f;

**const** **float** xPos = g\_InnerRadius \* sinf(thetaRad);

**const** **float** zPos = g\_InnerRadius \* cosf(thetaRad);

*// ティーポットの高さ方向の値*

**const** **float** yPos = g\_HeightOffset **+ pow(sinf(thetaRad),2.0)**; *// ★この値を少しずつ変化させることでティーポットが上下に移動する*

*// ティーポットの色の指定*

glMaterialfv(GL\_FRONT, GL\_AMBIENT, g\_Teapots[i].ambient.colors);

glMaterialfv(GL\_FRONT, GL\_DIFFUSE, g\_Teapots[i].diffuse.colors);

glMaterialfv(GL\_FRONT, GL\_SPECULAR, g\_Teapots[i].specular.colors);

glMaterialfv(GL\_FRONT, GL\_SHININESS, &g\_Teapots[i].shininess);

*// ティーポットの描画*

glPushMatrix();

glTranslatef(xPos, yPos, zPos);

glRotatef(thetaDegree, 0, 1, 0);

glRotatef(g\_Teapots[i].angle, 0, 0, 1);

glutSolidTeapot(1.2f \* g\_TeapotSize);

glPopMatrix();

*// ティーポットを支える柱の色の指定*

glMaterialfv(GL\_FRONT, GL\_AMBIENT, ambientColor);

glMaterialfv(GL\_FRONT, GL\_DIFFUSE, diffuseColor);

glMaterialfv(GL\_FRONT, GL\_SPECULAR, specularColor);

glMaterialfv(GL\_FRONT, GL\_SHININESS, &shininess);

*// ティーポットを支える柱の描画*

glPushMatrix();

glTranslatef(xPos, -1.f, zPos);

displayCylinder(0.3f, yPos + 1.f, 32);

glPopMatrix();

}

glutSwapBuffers();

}

**float** frand() { return rand() / (**float**)RAND\_MAX; }

*// 初期設定を行う関数*

**void** init()

{

glClearColor(1, 1, 1, 1);

glClearDepth(100.f);

**float** lightAmbientColor0[] = {0.2f, 0.2f, 0.2f, 0.0f};

**float** lightDiffuseColor0[] = {0.4f, 0.4f, 0.4f, 0.0f};

**float** lightSpecularColor0[] = {0.8f, 0.8f, 0.8f, 0.0f};

**float** lightPosition0[] = {5.0f, 5.0f, 8.0f, 0.0f};

**float** lightAmbientColor1[] = {0.2f, 0.2f, 0.2f, 0.0f};

**float** lightDiffuseColor1[] = {0.4f, 0.4f, 0.4f, 0.0f};

**float** lightSpecularColor1[] = {0.8f, 0.8f, 0.8f, 0.0f};

**float** lightPosition1[] = {-5.0f, 2.0f, 3.0f, 0.0f};

glEnable(GL\_LIGHTING);

glEnable(GL\_LIGHT0);

glLightfv(GL\_LIGHT0, GL\_AMBIENT, lightAmbientColor0);

glLightfv(GL\_LIGHT0, GL\_DIFFUSE, lightDiffuseColor0);

glLightfv(GL\_LIGHT0, GL\_SPECULAR, lightSpecularColor0);

glLightfv(GL\_LIGHT0, GL\_POSITION, lightPosition0);

glEnable(GL\_LIGHT1);

glLightfv(GL\_LIGHT1, GL\_AMBIENT, lightAmbientColor1);

glLightfv(GL\_LIGHT1, GL\_DIFFUSE, lightDiffuseColor1);

glLightfv(GL\_LIGHT1, GL\_SPECULAR, lightSpecularColor1);

glLightfv(GL\_LIGHT1, GL\_POSITION, lightPosition1);

srand(0);

*// 個々のティーポットの色を設定する処理 乱数で決めている*

for (**int** i = 0; i < g\_NumTeapots; i++)

{

g\_Teapots[i].ambient.r = 0.2f \* frand();

g\_Teapots[i].ambient.g = 0.2f \* frand();

g\_Teapots[i].ambient.b = 0.2f \* frand();

g\_Teapots[i].ambient.a = 1.f;

g\_Teapots[i].diffuse.r = 0.2f \* frand() + 0.8f;

g\_Teapots[i].diffuse.g = 0.2f \* frand() + 0.8f;

g\_Teapots[i].diffuse.b = 0.2f \* frand() + 0.8f;

g\_Teapots[i].diffuse.a = 1.f;

g\_Teapots[i].specular.r = 0.3f \* frand() + 0.2f;

g\_Teapots[i].specular.g = 0.3f \* frand() + 0.2f;

g\_Teapots[i].specular.b = 0.3f \* frand() + 0.2f;

g\_Teapots[i].specular.a = 1.f;

g\_Teapots[i].shininess = 2.f + 30 \* frand();

g\_Teapots[i].angle = 15 \* (2.f \* frand() - 1.f);

}

glEnable(GL\_DEPTH\_TEST);

}

*// 一定時間ごとに実行される関数*

**void** timer(**int** val)

{

*// 回転角度の更新*

g\_RotationDegree += g\_DeltaRotationDegree;

**const** **float** rotationRad = 2.f \* g\_RotationDegree \* M\_PI / 180.f;

*// ★ 下のコードでは視点が固定だけど*

*// ここで g\_EyeY と g\_EyeZ の値を変えることで視点位置を変化させることができる*

g\_EyeY = g\_EyeCenterY + g\_EyeRadius **\* sinf(rotationRad)**;

g\_EyeZ = g\_EyeCenterZ + g\_EyeRadius **\* sinf(rotationRad)**;

glutPostRedisplay();

glutTimerFunc(g\_AnimationIntervalMsec, timer, val);

}

*// ウィンドウサイズが変更されたときの処理*

**void** reshape(**int** w, **int** h)

{

if (h < 1)

return;

*// ビューポートをウィンドウサイズに変更*

glViewport(0, 0, w, h);

g\_WindowWidth = w;

g\_WindowHeight = h;

}

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

{

glutInit(&argc, argv);

glutInitDisplayMode(GLUT\_DOUBLE | GLUT\_RGB | GLUT\_DEPTH);

glutInitWindowSize(g\_WindowWidth, g\_WindowHeight);

*// ウィンドウタイトルに表示する文字列を指定する場合*

glutCreateWindow("LA LA LA LOVE SONG feat. Tsukuba");

glutDisplayFunc(display);

glutReshapeFunc(reshape); *// ウィンドウサイズが変更されたときに実行される関数を指定*

glutTimerFunc(g\_AnimationIntervalMsec, timer, 0);

*// 初期設定を行う*

init();

glutMainLoop();

return 0;

}

*グラフ が含まれている画像

自動的に生成された説明*