コンピューターグラフィックス基礎 第三回課題

情報メディア創生学類 3 年 202313625 藤川興昌

実行環境

- Ubuntu 22.04.3 LTS
- gcc version 11.4.0

課題

ソースコード

```
#define _USE_MATH_DEFINES       // Visual Studio ﾍﾍ ﾍ~ﾍﾍﾍﾍﾍﾆﾍﾍﾃ茖 M_PI
ﾍﾍﾍg﾿pﾍﾍﾍ邯ﾍﾍ
#include <cstdlib>
#include <GL/glut.h>
#include <cmath>
#include <stdio.h>

// ﾍF﾿ﾍﾍﾍﾍi﾿[ﾍﾍﾍ鋤﾿pﾍﾍ
union color {
        struct { float r, g, b, a; };
        float colors[4];
};

// ﾍ^X﾿e﾿B﾿[﾿|﾿b﾿g﾿Fﾍﾍ﾿A﾿Xﾍﾍﾍ﾿p﾿x﾿qLﾍﾍﾍﾍﾍﾍ﾿sﾍﾍﾍﾍ邯﾿P\ﾍﾍﾍﾍ
struct TeapotData {
        color ambient, diffuse, specular;
        float shininess, angle;
};

//
﾿Oﾍﾍﾍ[﾿oﾍﾍﾍ﨧ﾍﾍi﾿vﾍﾍﾍO﾿ﾍﾍﾍﾍﾍ﾿ｷﾍﾍﾍﾍﾍ﾿lﾍﾍA﾿N﾿Z﾿X﾿lﾍﾍﾍ﨧﾿
﾿j﾿ｯ﾿ g_ ﾍﾍtﾍﾍﾍĂﾍﾍﾍ
const int g_NumTeapots = 8;
TeapotData g_Teapots[g_NumTeapots];

// float﾿^﾿l﾿AﾍﾍﾍﾍﾍﾍﾍﾍfﾍﾍtﾍﾍﾍﾍB﾿ﾍﾍﾍﾍﾍ[ﾍﾍﾍûﾍﾍ﾿lﾍﾍﾍ
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;

const float g_ChangeHeightSpeed = 3.0;
const float g_DefaultHeightVariation = 5.0;

int g_WindowWidth = 512;
int g_WindowHeight = 512;

// ?~????`??c?????~???a?A?????A?~?????
// glut?~????`?惊?邽??H??????ŁA?Ǝ??ɣ???
void displayCylinder(float radius, float height, int nSlices) {
        // ?V????
        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();

        // ???
        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();

        // ????
        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();
```

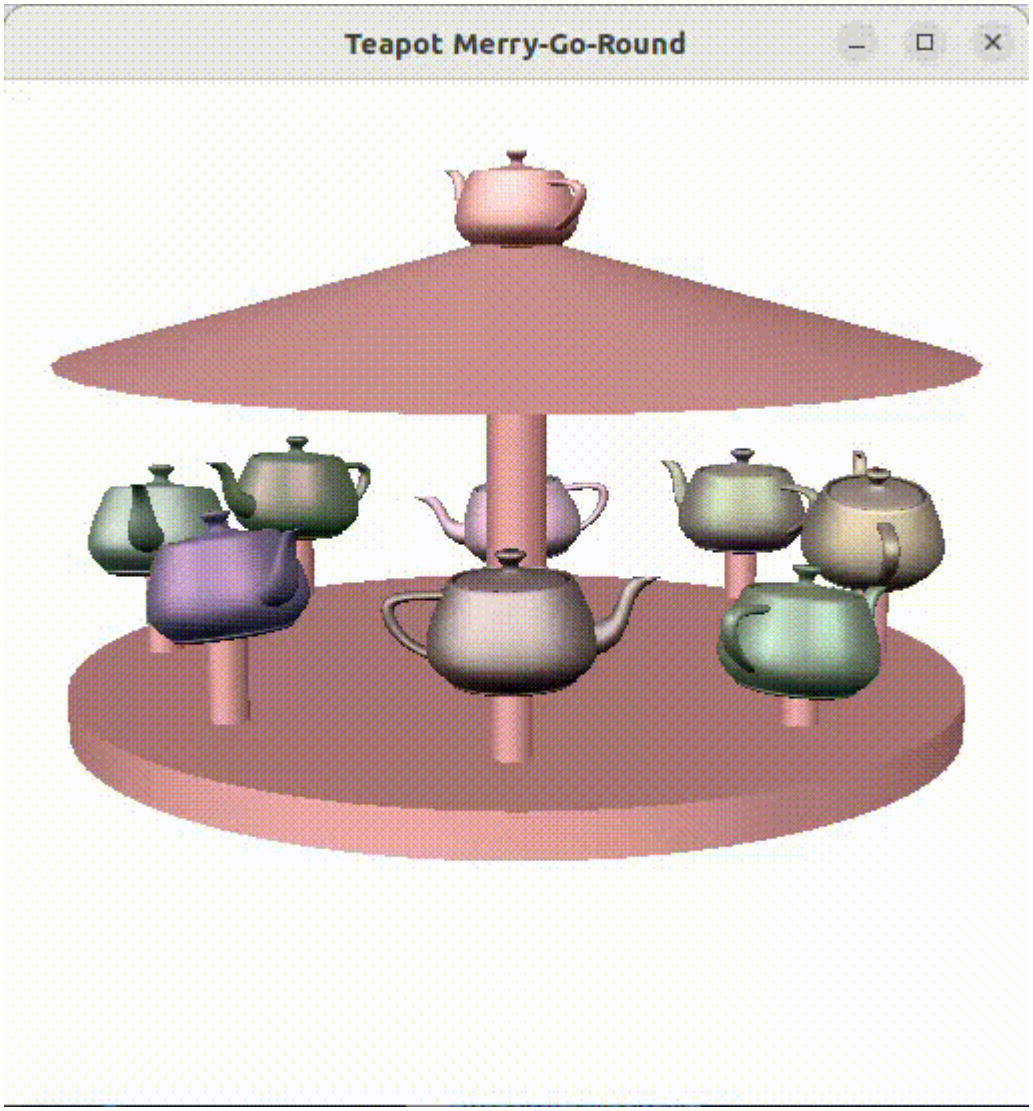
```
}

void display() {
        glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);

        // ********e*i**_*
        glMatrixMode(GL_PROJECTION);
        glLoadIdentity();
        gluPerspective(30.0, g_WindowWidth/(double)g_WindowHeight, 1,
100.0);

        // ***f*****W****h**[*h*_*L*
        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);

        // ****
        glPushMatrix();
        glTranslatef(0, g_HeightAmplitude + g_HeightOffset + 3.f, 0);
        glRotatef(-90, 1, 0, 0);
        glutSolidCone(g_OuterRadius, 2.f, 32, 32);
        glPopMatrix();

        // ***S**
        glPushMatrix();
        glTranslatef(0, -1.f, 0);
        displayCylinder(0.5f, g_HeightAmplitude + g_HeightOffset + 6.5f,
32);
        glPopMatrix();

        // *y**
        glPushMatrix();
        glTranslatef(0, -2.f, 0);
        displayCylinder(g_OuterRadius, 0.7f, 64);
        glPopMatrix();

        // *******e*B*[*|*b*g
        glPushMatrix();
        glTranslatef(0, g_HeightAmplitude + g_HeightOffset + 5.5f, 0);
        glRotatef(g_RotationDegree, 0, 1, 0); // **]******A****
        glutSolidTeapot(g_TeapotSize);
        glPopMatrix();

        const float deltaTheta = 360 / (float)g_NumTeapots;

        // *e*B*[*|*b*g*f****1*(**_`*悞**
        // ******L*R*[*h*ÍA**b*****ʒu*ɛ`*悪****ŁA*S*****]*****
```

```
恩ﾂ｢ｺXﾂ・ﾂ・
        for (int i = 0; i < g_NumTeapots; i++) {
                const float thetaDegree = deltaTheta * i + g_RotationDegree;
// ﾂe・B・[・|・b・g・ｺu・・・・・ﾄ邯・ﾄp・x

                const float thetaRad = thetaDegree * M_PI / 180.f;
                const float xPos = g_InnerRadius * sinf(thetaRad);
                const float zPos = g_InnerRadius * cosf(thetaRad);

                // ・e・B・[・|・b・g・・・・・・・・l
                const float yPos = g_HeightOffset + abs(sin(M_PI *
(g_RotationDegree * g_ChangeHeightSpeed + g_Teapots[i].angle *
g_DefaultHeightVariation) / 180.0)); // ・・・・・l・・・・・・・・ｰω・・・・・・
邱・ﾐﾅe・B・[・|・b・g・・・㊥・ｲ・・・・・

                // ・e・B・[・|・b・g・F・w・・
                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);

                // ・e・B・[・|・b・g・``・・
                glPushMatrix();
                glTranslatef(xPos, yPos, zPos);
                glRotatef(thetaDegree, 0, 1, 0);
                glRotatef(g_Teapots[i].angle, 0, 0, 1);
                glutSolidTeapot(1.2f * g_TeapotSize);
                glPopMatrix();

                // ・e・B・[・|・b・g・・・x・・・釘・F・w・・
                glMaterialfv(GL_FRONT, GL_AMBIENT,   ambientColor);
                glMaterialfv(GL_FRONT, GL_DIFFUSE,   diffuseColor);
                glMaterialfv(GL_FRONT, GL_SPECULAR,  specularColor);
                glMaterialfv(GL_FRONT, GL_SHININESS, &shininess);

                // ・e・B・[・|・b・g・・・x・・・釘・``・・
                glPushMatrix();
                glTranslatef(xPos, -1.f, zPos);
                displayCylinder(0.3f, yPos + 1.f, 32);
                glPopMatrix();
        }

        glutSwapBuffers();
}

float frand() { return rand() / (float)RAND_MAX; }

// ・・・・・ﾍ・・s・・・・・
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);

        // ^X e B [ | b g F ﾝ肢 鏈  ŌĂ
        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);
}

// 莞 ƴ s
void timer(int val) {
        // ]p x X V
        g_RotationDegree += g_DeltaRotationDegree;

        const float rotationRad = 2.f * g_RotationDegree * M_PI / 180.f;
```

```
        // ?? ???R?[?h?ł??_???Œ腎????
        // ?????? g_EyeY ?? g_EyeZ ?l??ς??邱
?Бŏ??_?ʒu??ω???????邱?Б??ł???
        g_EyeY = g_EyeCenterY + 2.5 * sin(M_PI * g_RotationDegree * 5.0 /
180.0);
        g_EyeZ = g_EyeCenterZ + 2.5 * cos(M_PI * g_RotationDegree * 5.0 /
180.0);

        glutPostRedisplay();

        glutTimerFunc(g_AnimationIntervalMsec, timer, val);
}

// ?E?B???h?E?T?C?Y???úX???▫▫Б?????
void reshape(int w, int h) {
        if (h < 1) return;

        // ?r???[?|?[?g???E?B???h?E?T?C?Y?ɞúX
        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);

        // ?E?B???h?E?^?C?g???ɞ\?????镶???????w?肰??▫
        glutCreateWindow("Teapot Merry-Go-Round");

        glutDisplayFunc(display);
        glutReshapeFunc(reshape); // ?E?B???h?E?T?C?Y???úX???▫
▫Б??Ɣ??s?????▫????w??
        glutTimerFunc(g_AnimationIntervalMsec, timer, 0);

        // ?????ゃ???s??
        init();

        glutMainLoop();

        return 0;
}
```

スクリーンショット



発展課題

ソースコード

```
#define _USE_MATH_DEFINES       // Visual Studio ﾍﾍ ﾍ~ﾍﾍﾍﾍﾍﾃﾍﾍﾃ M_PI
ﾍﾍﾍg﾿pﾍﾍﾍ邯ﾍﾍ
#include <cstdlib>
#include <GL/glut.h>
#include <cmath>
#include <stdio.h>

// ﾍF﾿﾿﾿﾿﾿i﾿[ﾍﾍﾍ鋤﾿pﾍﾍ
union color {
        struct { float r, g, b, a; };
        float colors[4];
};

// ﾍ^X﾿e﾿B﾿[﾿|﾿b﾿g﾿FﾍﾍA﾿Xﾍﾍﾍp﾿x﾿qL﾿﾿﾿﾿﾿﾿﾿
```

```
};

//
O[o[ioviOtA[ZXt
j? g_ t
const int g_NumTeapots = 8;
TeapotData g_Teapots[g_NumTeapots];

// float^lAftB[u
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;

const float g_ChangeHeightSpeed = 3.0;
const float g_DefaultHeightVariation = 5.0;

int g_WindowWidth = 512;
int g_WindowHeight = 512;

// ~`c~a[A[A~
// glut?~`[AA
void displayCylinder(float radius, float height, int nSlices) {
        // V
        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();

        //
        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();
```

```
        // ????
        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);

        // ????????e?i??_?
        glMatrixMode(GL_PROJECTION);
        glLoadIdentity();
        gluPerspective(30.0, g_WindowWidth/(double)g_WindowHeight, 1,
100.0);

        // ???f?????W????ph??[?h?_?L?
        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);

        // ????
        glPushMatrix();
        glTranslatef(0, g_HeightAmplitude + g_HeightOffset + 3.f, 0);
        glRotatef(-90, 1, 0, 0);
        glutSolidCone(g_OuterRadius, 2.f, 32, 32);
        glPopMatrix();

        // ???S??
        glPushMatrix();
        glTranslatef(0, -1.f, 0);
        displayCylinder(0.5f, g_HeightAmplitude + g_HeightOffset + 6.5f,
32);
        glPopMatrix();

        // ?y??
        glPushMatrix();
        glTranslatef(0, -2.f, 0);
        displayCylinder(g_OuterRadius, 0.7f, 64);
        glPopMatrix();
```

```
        // ������e�B[�|�b�g
        glPushMatrix();
        glTranslatef(0, g_HeightAmplitude + g_HeightOffset + 5.5f, 0);
        glRotatef(g_RotationDegree, 0, 1, 0); // ��]�����Ă���
        glutSolidTeapot(g_TeapotSize);
        glPopMatrix();

        const float deltaTheta = 360 / (float)g_NumTeapots;

        // ̪e�B[�|̪b̪g̪ƒ̪̪̪1̪(̪̪▫`̪惊̪̪
        // ̪̪̪̪̪L̪R̪[̪h̪ɪ́A̪̪b̪̪̪̪ʒu̪ɕ`̪憲̪̪̪ŁA̪S̪̪̪̪]̪̪̪̪
恩̪ɕúX̪̪̪̪
        for (int i = 0; i < g_NumTeapots; i++) {
                const float thetaDegree = deltaTheta * i + g_RotationDegree;
// ̪e̪B̪[̪|̪b̪g̪ʒu̪̪̪̪̪Ꮋ邽̪Ꮋp̪x

                const float thetaRad = thetaDegree * M_PI / 180.f;
                const float xPos = g_InnerRadius * sinf(thetaRad);
                const float zPos = g_InnerRadius * cosf(thetaRad);

                // ̪e̪B̪[̪|̪b̪g̪̪̪̪̪̪̪̪̪l
                const float yPos = g_HeightOffset + abs(sin(M_PI *
(g_RotationDegree * g_ChangeHeightSpeed + g_Teapots[i].angle *
g_DefaultHeightVariation) / 180.0)); // ̪̪̪̪̪l̪̪̪̪̪̪̪̪̪▫ω̪̪̪̪̪̪
邱̪ƁŃe̪B̪[̪|̪b̪g̪̪̪㊢̪ʖ̪̪̪̪̪

                // ̪e̪B̪[̪|̪b̪g̪F̪w̪̪
                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);

                // ̪e̪B̪[̪|̪b̪g̪`̪̪
                glPushMatrix();
                glTranslatef(xPos, yPos, zPos);
                glRotatef(thetaDegree, 0, 1, 0);
                glRotatef(g_Teapots[i].angle, 0, 0, 1);
                glutSolidTeapot(1.2f * g_TeapotSize);
                glPopMatrix();

                // ̪e̪B̪[̪|̪b̪g̪̪̪x̪̪̪钌̪F̪w̪̪
                glMaterialfv(GL_FRONT, GL_AMBIENT,   ambientColor);
                glMaterialfv(GL_FRONT, GL_DIFFUSE,   diffuseColor);
                glMaterialfv(GL_FRONT, GL_SPECULAR,  specularColor);
                glMaterialfv(GL_FRONT, GL_SHININESS, &shininess);

                // ̪e̪B̪[̪|̪b̪g̪̪̪x̪̪̪钌̪`̪̪
                glPushMatrix();
                glTranslatef(xPos, -1.f, zPos);
                displayCylinder(0.3f, yPos + 1.f, 32);
                glPopMatrix();
        }
```

```
        glutSwapBuffers();
}

float frand() { return rand() / (float)RAND_MAX; }

// ������ے���s���□�
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);

        // ^X e B [ | b g F ے欹鏈 ōĂ
        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);
}

// ??莞?d??Б???s?????▫?
void timer(int val) {
        // ??]?p?x?X?V
        g_RotationDegree += g_DeltaRotationDegree;

        const float rotationRad = 2.f * g_RotationDegree * M_PI / 180.f;

        // ?? ???R?[?h?ł??_???Œ腎????
        // ?????? g_EyeY ?? g_EyeZ ?l??ς??邱
?Бǒ??_?ʒu??ω???????邱?Б??ł???
        g_EyeY = g_EyeCenterY + 2.5 * sin(M_PI * g_RotationDegree * 5.0 /
180.0);
        g_EyeZ = g_EyeCenterZ + 2.5 * cos(M_PI * g_RotationDegree * 5.0 /
180.0);

        glutPostRedisplay();

        glutTimerFunc(g_AnimationIntervalMsec, timer, val);
}

// ?E?B???h?E?T?C?Y???úX???▫▫Б?????
void reshape(int w, int h) {
        if (h < 1) return;

        // ?r???[?|?[?g???E?B???h?E?T?C?Y?ɞúX
        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);

        // ?E?B???h?E?^?C?g???ɞ\??????镶???????w?肰??▫
        glutCreateWindow("Teapot Merry-Go-Round");

        glutDisplayFunc(display);
        glutReshapeFunc(reshape); // ?E?B???h?E?T?C?Y???úX???▫
▫Б??Ɏ??s??????▫?????w??
        glutTimerFunc(g_AnimationIntervalMsec, timer, 0);

        // ?????ۂ???s??
        init();

        glutMainLoop();

        return 0;
}
```

スクリーンショット