Comandos para executar o código:

```
$ g++ -c 202010140-EP03-Q01.c
$ g++ 202010140-EP03-Q01.o -lGL -lglut -lGLU -o <nome-executável>
$ ./<nome-executável>
```

Código:

```c
#include <GL/glut.h>

/* Cria textura listrada */
#define stripedTextureWidth 32
#define stripedTextureHeight 32
static GLubyte
stripedTexture[stripedTextureHeight][stripedTextureWidth][4];

GLfloat angleX = -45.0f;
GLfloat angleY = 30.0f;

static GLfloat xequalzero[] = { 1.0, 0.0, 0.0, 0.0 };
static GLfloat* currentCoeff;
static GLenum currentPlane;
static GLint currentGenMode;

void createStripedTexture(void)
{
    for (int i = 0; i < stripedTextureHeight; i++) {
        for (int j = 0; j < stripedTextureWidth; j++) {
            stripedTexture[i][j][0] = (GLubyte)((j <= 4) ? 255 : 0);
            stripedTexture[i][j][1] = (GLubyte)((j > 4) ? 255 : 0);
            stripedTexture[i][j][2] = (GLubyte)0;
            stripedTexture[i][j][3] = (GLubyte)255;
        }
    }
}

void init(void)
{
    glClearColor(0.0, 0.0, 0.0, 0.0);
    glEnable(GL_DEPTH_TEST);
    glShadeModel(GL_SMOOTH);

    createStripedTexture();
    glPixelStorei(GL_UNPACK_ALIGNMENT, 1);
```

```
    glTexParameteri(GL_TEXTURE_1D, GL_TEXTURE_WRAP_S, GL_REPEAT);
    glTexParameteri(GL_TEXTURE_1D, GL_TEXTURE_MAG_FILTER, GL_LINEAR);
    glTexParameteri(GL_TEXTURE_1D, GL_TEXTURE_MIN_FILTER, GL_LINEAR);

    glTexImage1D(GL_TEXTURE_1D, 0, 4, stripedTextureWidth,
                 0, GL_RGBA, GL_UNSIGNED_BYTE, stripedTexture);

    glTexEnvf(GL_TEXTURE_ENV, GL_TEXTURE_ENV_MODE, GL_MODULATE);
    currentCoeff = xequalzero;
    currentGenMode = GL_OBJECT_LINEAR;
    currentPlane = GL_OBJECT_PLANE;
    glTexGeni(GL_S, GL_TEXTURE_GEN_MODE, currentGenMode);
    glTexGenfv(GL_S, currentPlane, currentCoeff);

    glEnable(GL_TEXTURE_GEN_S);
    glEnable(GL_TEXTURE_1D);
    glEnable(GL_CULL_FACE);
    glEnable(GL_LIGHTING);
    glEnable(GL_LIGHT0);
    glEnable(GL_AUTO_NORMAL);
    glEnable(GL_NORMALIZE);
    glEnable(GL_DEPTH_TEST);
    glFrontFace(GL_CW);
    glCullFace(GL_BACK);
    glMaterialf(GL_FRONT, GL_SHININESS, 64.0);
}

void display(void) {
    glClearColor(0.0f, 0.0f, 0.0f, 0.0f);
    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
    glBegin(GL_LINES);
    glColor3f(0.0f, 0.0f, 1.0f);
    glVertex3f(0.0f, 0.0f, 0.0f);
    glVertex3f(0.7f, 0.0f, 0.0f);
    glColor3f(1.0f, 0.0f, 0.0f);
    glVertex3f(0.0f, 0.0f, 0.0f);
    glVertex3f(0.0f, 0.5f, 0.0f);
    glColor3f(0.0f, 1.0f, 0.0f);
    glVertex3f(0.0f, 0.0f, 0.0f);
    glVertex3f(0.0f, 0.0f, 0.7f);
    glEnd();
    glBegin(GL_POLYGON);
    glColor3f(1.0f, 0.0f, 0.0f);
```

```
    glVertex3f(0.0f, 0.0f, 0.0f);
    glVertex3f(0.0f, 0.2f, 0.0f);
    glVertex3f(0.2f, 0.3f, 0.0f);
    glVertex3f(0.4f, 0.2f, 0.0f);
    glVertex3f(0.4f, 0.0f, 0.0f);
    glEnd();
    glBegin(GL_QUADS);
    glColor3f(0.0f, 0.0f, 1.0f);
    glVertex3f(0.0f, 0.0f, 0.0f);
    glVertex3f(0.0f, 0.0f, 0.4f);
    glVertex3f(0.0f, 0.2f, 0.4f);
    glVertex3f(0.0f, 0.2f, 0.0f);
    glEnd();
    glBegin(GL_QUAD_STRIP);
    glColor3f(0.0f, 1.0f, 0.0f);
    glVertex3f(0.0f, 0.2f, 0.0f);
    glVertex3f(0.0f, 0.2f, 0.4f);
    glVertex3f(0.2f, 0.3f, 0.0f);
    glVertex3f(0.2f, 0.3f, 0.4f);
    glVertex3f(0.4f, 0.2f, 0.0f);
    glVertex3f(0.4f, 0.2f, 0.4f);
    glEnd();
    glColor3f(1.0f, 1.0f, 1.0f);
    glBegin(GL_LINE_STRIP);
    glVertex3f(0.0f, 0.0f, 0.4f);
    glVertex3f(0.4f, 0.0f, 0.4f);
    glVertex3f(0.4f, 0.0f, 0.0f);
    glEnd();
    glBegin(GL_LINES);
    glVertex3f(0.4f, 0.0f, 0.4f);
    glVertex3f(0.4f, 0.2f, 0.4f);
    glEnd();
    glFlush();
}

void reshape(int w, int h) {
    glViewport(0, 0, (GLsizei) w, (GLsizei) h);
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    gluPerspective(30.0, (GLfloat) w/(GLfloat) h, 1.0, 10.0);
    glMatrixMode(GL_MODELVIEW);
    glLoadIdentity();
    glTranslatef(0.0f, 0.0f, -3.0f);
```

```
    glRotatef(angleY, 1.0f, 0.0f, 0.0f);
    glRotatef(angleX, 0.0f, 1.0f, 0.0f);
}

void keyboardEvent(int key, int x, int y) {
    switch (key) {
        case GLUT_KEY_LEFT:
            angleX += 1;
            break;
        case GLUT_KEY_RIGHT:
            angleX -= 1;
            break;
        case GLUT_KEY_DOWN:
            angleY -= 1;
            break;
        case GLUT_KEY_UP:
            angleY += 1;
            break;
    }
    if (angleX == -360) angleX = 0;
    if (angleY == -360) angleY = 0;

    glMatrixMode(GL_MODELVIEW);
    glLoadIdentity();
    glTranslatef(0.0f, 0.0f, -3.0f);
    glRotatef(angleY, 1.0f, 0.0f, 0.0f);
    glRotatef(angleX, 0.0f, 1.0f, 0.0f);
    display();
}

int main(int argc, char* argv[]) {
    glutInit(&argc, argv);
    glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);
    glutInitWindowSize(800, 600);
    glutInitWindowPosition(100, 100);
    glutCreateWindow("EP05-Q01 - Vitor Melo");
    glutSpecialFunc(keyboardEvent);
    init();
    glutDisplayFunc(display);
    glutReshapeFunc(reshape);
    glutMainLoop();
    return 0;
}
```