```
// Solução possível para o efólio-B

import javax.media.opengl.*;
import javax.media.opengl.glu.GLU;
import com.sun.opengl.util.GLUT;

public class efolioB extends J1_4_Line{
    GLU glu = new GLU(); // interface para a biblioteca GLU

    // Definição de palete de cores
    float[] colorBlack  = {0.0f,0.0f,0.0f,1.0f};
    float[] colorCyan   = {0.0f,0.5f,0.5f,1.0f};
    float[] colorWhite  = {1.0f,1.0f,1.0f,1.0f};
    float[] colorGray   = {0.6f,0.6f,0.6f,1.0f};
    float[] colorGreen   = {0.0f,1.0f,0.0f,1.0f};
    float[] colorRed    = {1.0f,0.0f,0.0f,1.0f};
    float[] colorBlue   = {0.0f,0.0f,0.5f, 1.0f};
    float[] colorYellow = {1.0f,1.0f,0.0f,1.0f};
    float[] colorLightYellow = {.5f,.5f,0.0f,1.0f};
    float[] colorMagenta = {1.0f, 0.0f, 1.0f, 1.0f};

    // Tonalidades para a luz
    float blackish[] = {0.3f, 0.3f, 0.3f, 0.3f};
    float whitish[] = {0.8f, 0.8f, 0.8f, 1};

    // Localização de fontes e orientação do spot, localizações infinitas
    float position[] = {1, 1, 1, 0};
    float posLight1[] = { 1.0f, 1.0f, 1.0f, 0.0f };
    float posLight2[] = { 2.0f, 2.0f, 10.0f, 0.0f };
    float spotDirection[] = { -1.0f, -1.0f, 0.0f };

    // Controle de alternância entre ligar/desligar fontes
    boolean move = true;

    // Controle para rotação da fonte de luz 2
    float spin = 0.0f;

    public efolioB() {
        // utiliza o construtor da classe mãe e ativa duplo buffering
        capabilities.setDoubleBuffered(true);
    }

    public void init(GLAutoDrawable glDrawable) {

        super.init(glDrawable);

        // Ativa a luz e demais funcionalidades para sua boa operacionalização
        gl.glEnable(GL.GL_LIGHTING);
        gl.glEnable(GL.GL_NORMALIZE);
        gl.glEnable(GL.GL_CULL_FACE);

        // Parametriza a fonte de luz 0
        gl.glEnable(GL.GL_LIGHT0);
        gl.glLightfv(GL.GL_LIGHT0, GL.GL_POSITION, position,0);
        gl.glLightfv(GL.GL_LIGHT0, GL.GL_AMBIENT, whitish,0);
        gl.glLightfv(GL.GL_LIGHT0, GL.GL_DIFFUSE, colorWhite,0);
        gl.glLightfv(GL.GL_LIGHT0, GL.GL_SPECULAR, colorWhite,0);
```

```
        // Parametriza a fonte de luz 1 - é um foco de luz
        gl.glEnable(GL.GL_LIGHT1 );
        gl.glLightfv(GL.GL_LIGHT1, GL.GL_POSITION, posLight1, 0);
        gl.glLightf(GL.GL_LIGHT1, GL.GL_SPOT_CUTOFF, 60.0F);
        gl.glLightfv(GL.GL_LIGHT1, GL.GL_SPOT_DIRECTION, spotDirection, 0 );
        gl.glLightfv(GL.GL_LIGHT1, GL.GL_AMBIENT, colorGray, 0 );
        gl.glLightfv(GL.GL_LIGHT1, GL.GL_DIFFUSE, colorGray, 0 );
        gl.glLightfv(GL.GL_LIGHT1, GL.GL_SPECULAR, colorWhite,0 );
        gl.glLightf(GL.GL_LIGHT1, GL.GL_CONSTANT_ATTENUATION, 0.2f );

        // Parametriza a fonte de luz 1 - é um foco de luz
        gl.glEnable(GL.GL_LIGHT2 );
        gl.glLightfv(GL.GL_LIGHT2, GL.GL_POSITION, posLight2, 0);
        gl.glLightf(GL.GL_LIGHT2, GL.GL_SPOT_CUTOFF, 30.0F);
        gl.glLightfv(GL.GL_LIGHT2, GL.GL_SPOT_DIRECTION, spotDirection, 0 );
        gl.glLightfv(GL.GL_LIGHT2, GL.GL_AMBIENT, colorYellow, 0 );
        gl.glLightfv(GL.GL_LIGHT2, GL.GL_DIFFUSE, colorYellow, 0 );
        gl.glLightfv(GL.GL_LIGHT2, GL.GL_SPECULAR, colorWhite,0 );
        gl.glLightf(GL.GL_LIGHT2, GL.GL_CONSTANT_ATTENUATION, 0.2f );

        // Materiais para a iluminação padrão para todas
        gl.glMaterialfv(GL.GL_FRONT, GL.GL_AMBIENT, colorBlack,0);
        gl.glMaterialfv(GL.GL_FRONT, GL.GL_DIFFUSE, whitish,0);
        gl.glMaterialfv(GL.GL_FRONT, GL.GL_SPECULAR, colorWhite,0);
        gl.glMaterialf(GL.GL_FRONT, GL.GL_SHININESS, 100.0f);
        gl.glMaterialfv(GL.GL_FRONT, GL.GL_EMISSION, colorBlack,0);
    }

    public void reshape(GLAutoDrawable drawable, int x, int y, int width, int height) {

        // Define o view port, inicializa a pilha de matrizes de projeção e define a mesma em
        // perspectiva
        gl.glViewport(0, 0, WIDTH, HEIGHT);
        gl.glMatrixMode(GL.GL_PROJECTION);
        gl.glLoadIdentity();
        glu.gluPerspective(45, WIDTH/HEIGHT,1.0, 800);
    }

    public void display(GLAutoDrawable drawable) {

        // Ativa buffers para gestão de cor e redesenho de profundidade
        gl.glClear(GL.GL_COLOR_BUFFER_BIT|GL.GL_DEPTH_BUFFER_BIT);

        // Modo de sobreamento é o smooth
        gl.glShadeModel(GL.GL_SMOOTH);

        // Liga e desliga fontes 0 e 1
        if (move)
        {
            gl.glDisable(GL.GL_LIGHT0);
            gl.glEnable(GL.GL_LIGHT1);
            drawScene();
        }
        else {
            gl.glDisable(GL.GL_LIGHT1);
            gl.glEnable(GL.GL_LIGHT0);
```

```
            drawScene();
        }

        // Garante a pilha da model-view estar sem transformações indesejáveis acumuladas
        gl.glMatrixMode(GL.GL_MODELVIEW);
        gl.glLoadIdentity();

        // Especifica posição do observador da cena e do ponto para o qual olha
        glu.gluLookAt(0,70,200, 3*WIDTH/6, 3*HEIGHT/6, 0.0f, 0,0,1);

        // torna refrescamento de desenho mais lento
        try {
            Thread.sleep(100);
        } catch (Exception ignore) {
            }

    }

    // Desenha a cena
    public void drawScene() {
        if (spin > 360.0f) spin = 0.0f;
        ++spin;
        move = !move;
        // Desenha o plano em verde, posicionando e definindo material
        gl.glPushMatrix();
            gl.glTranslatef(3*WIDTH/6 - 30, 3*HEIGHT/6 - 20, 0.0f);
            gl.glScalef(350.0f, 250.0f, 2.0f);
            gl.glMaterialfv(GL.GL_FRONT, GL.GL_EMISSION, colorGreen, 0);
            glut.glutSolidCube(1);
        gl.glPopMatrix();

        // Desenha o cone em vermelho, posicionando e definindo material
        gl.glPushMatrix();
            gl.glTranslatef(3*WIDTH/6, 3*HEIGHT/6, 10.0f);
            gl.glScalef(40.0f, 40.0f, 40.0f);
            gl.glMaterialfv(GL.GL_FRONT, GL.GL_EMISSION, colorRed,0);
            glut.glutSolidCone(1, 1, 20, 20);
        gl.glPopMatrix();

        // Desenha a esfera em amarelo, posicionando e definindo material
        gl.glPushMatrix();
            gl.glTranslatef(3*WIDTH/6 + 150, 3*HEIGHT/6, 15.0f);
            gl.glScalef(40.0f, 40.0f, 40.0f);
            gl.glMaterialfv(GL.GL_FRONT, GL.GL_EMISSION, colorLightYellow,0);
            glut.glutSolidSphere(1, 20, 20);
        gl.glPopMatrix();

        // Desenha o cilindro em magenta, posicionando e definindo material
        gl.glPushMatrix();
            gl.glTranslatef(3*WIDTH/6 - 150, 3*HEIGHT/6, 15.0f);
            gl.glScalef(40.0f, 40.0f, 40.0f);
            gl.glMaterialfv(GL.GL_FRONT, GL.GL_EMISSION, colorCyan, 0);
            glut.glutSolidCylinder(1, 1, 20, 20);
        gl.glPopMatrix();

        // Desenha uma esfera acima do plano com os demais objetos e associa uma fonte de
        luz móvel
```

```
        gl.glPushMatrix ();
            gl.glTranslated (3*WIDTH/6 + 300, 3*HEIGHT/6, 100.0f);
            gl.glMaterialfv(GL.GL_FRONT, GL.GL_EMISSION, colorMagenta, 0);
            gl.glScalef(20.0f, 20.0f, 20.0f);
            gl.glPushMatrix ();
                gl.glRotatef (spin, 1.0f, 0.0f, 0.0f);
                gl.glLightfv(GL.GL_LIGHT2, GL.GL_POSITION, posLight2, 0);
                gl.glTranslated (0.0, 0.0, 1.5);
                gl.glDisable (GL.GL_LIGHTING);
                gl.glColor3d (0.0, 1.0, 1.0);
                glut.glutWireCube (1); // indicador da posição da luz
                gl.glEnable (GL.GL_LIGHTING);
           gl.glPopMatrix ();
           glut.glutSolidSphere(1, 20, 20);
        gl.glPopMatrix ();

        gl.glFlush();
    }

    public static void main(String[] args) {
         efolioB f = new efolioB();

         f.setTitle("Efólio-B");
         f.setSize(WIDTH, HEIGHT);
         f.setVisible(true);
    }

}
```