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

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

@SuppressWarnings("serial")
public class efolioA extends J1_4_Line {

  double cx = WIDTH/2, cy = HEIGHT/2, r = WIDTH/3, theta;
  double delta = 1/r;
  float xrot = 0 , yrot = 0, zrot = 0;
  float xspeed = 0.5f, yspeed = 0.5f;
  GLU glu = new GLU(); // interface para a biblioteca GLU

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

  public void reshape(GLAutoDrawable drawable, int x, int y, int width, int height) {
  //invocada sempre que ocorre um redimensionar da janela

     // gl.glMatrixMode(GL.GL_PROJECTION);
      //gl.glLoadIdentity();

     // gl.glMatrixMode(GL.GL_MODELVIEW);
     // gl.glLoadIdentity();
  }

// Invocado sempre para refrescar a cena
  public void display(GLAutoDrawable drawable) {
    gl.glClearColor (1.0f, 1.0f, 0.0f, 0.0f); // Define o amarelo como cor de fundo

    viewPort1(); // Visualiza cena com o primeiro viewport

    viewPort2(); // Visualiza cena em novo viewport

 // atualiza valor de delta para desenho de círculo e hexágono
    theta = theta+delta;

    gl.glFlush(); // efetiva o desenho, descarregando tudo

    System.out.printf("%5.0f %5.0f\n", x, y);
    // torna refrescamento de desenho mais lento
    try {
        Thread.sleep(20);
    } catch (Exception ignore) {
        }
   }

  // Especifica o desenho da cena
  public void drawScene() {

      //Define o tamanho dos pontos
      gl.glPointSize(1);
```

```
      // Desenha o quadrado no fundo
      drawSquare();

      // Muda a cor ativa para branco e desenha o círculo,o hexágono e as palavras
      gl.glColor3f(1.0f, 1.0f, 1.0f);
      drawCircle(cx, cy, r);
      drawHexagon (cx, cy);
      writeWords();

      //desenha um ponto vermelho grande no círculo
      gl.glPushMatrix();   // para isolar as transformações no ponto vermelho
        gl.glTranslatef(3*WIDTH/6, 3*HEIGHT/6, 0.0f); // Move para ponto pivot
        gl.glRotatef(yrot, 0.0f, 1.0f, 0.0f);   //X
        gl.glTranslatef(-3*WIDTH/6, -3*HEIGHT/6, 0.0f); // devolve a posição de origem

        gl.glPointSize(8);
        gl.glColor3f(1.0f, 0.0f, 0.0f);
        double x = r*Math.cos(theta)+cx;
        double y = r*Math.sin(theta)+cy;
        drawPoint(x, y);
      gl.glPopMatrix();

      // Desenha a esfera no centro com função GLUT
      gl.glPushMatrix();
          gl.glTranslatef(WIDTH/2, HEIGHT/2, 0); // posição final
          gl.glScalef(100.0f, 100.0f, 100.0f);
          gl.glColor3f(1.0f,0.0f,0.0f);
          glut.glutWireSphere(1, 10, 10);
      gl.glPopMatrix();
 }

  // Escreve as palavras no centro da cena
  public void writeWords() {
        gl.glColor3f(0.0f, 1.0f, 0.0f);
        gl.glRasterPos3f(WIDTH/2, HEIGHT/2, 0); // posição final
        glut.glutBitmapCharacter(GLUT.BITMAP_HELVETICA_18, 'B');
        glut.glutBitmapString(GLUT.BITMAP_HELVETICA_18, "itmap");

        gl.glPushMatrix();
            gl.glTranslatef(WIDTH/2-20, HEIGHT/2-30, 0); // posição final
            gl.glScalef(0.2f, 0.2f, 0.2f);
            glut.glutStrokeCharacter(GLUT.STROKE_ROMAN, 'S');
            glut.glutStrokeString(GLUT.STROKE_ROMAN, "troke");
        gl.glPopMatrix();
  }

// Desenha o círculo
  public void drawCircle(double x0, double y0, double r) {
     gl.glPushMatrix();   // para isolar as transformações no círculo
        gl.glTranslatef(3*WIDTH/6, 3*HEIGHT/6, 0.0f); // Move para ponto pivot
        gl.glRotatef(yrot, 0.0f, 1.0f, 0.0f);   //X
        gl.glTranslatef(-3*WIDTH/6, -3*HEIGHT/6, 0.0f); // devolve a posição de origem

        double th = 0;
        while (th<=2*Math.PI) {
            th = th+delta;
            double x = r*Math.cos(th)+x0;
```

```
            double y = r*Math.sin(th)+y0;
            drawPoint(x, y);
        }

    gl.glPopMatrix();
  }

  // Desenha o quadrado
  public void drawSquare() {

      gl.glPushMatrix();   // para isolar as transformações no quadrado
        gl.glTranslatef(3*WIDTH/6, 3*HEIGHT/6, 0.0f); // Move para ponto pivot
        gl.glRotatef(zrot, 0.0f, 0.0f, 1.0f);   //Z
        //gl.glRotatef(xrot, 1.0f, 0.0f, 0.0f);   //X
        //gl.glRotatef(yrot, 0.0f, 1.0f, 0.0f);   //Y
        gl.glTranslatef(-3*WIDTH/6, -3*HEIGHT/6, 0.0f); // devolve a posição de origem

        // Ativa cor azul
        gl.glColor3f(0.0f, 0.0f, 1.0f);

         // Desenha quadrado com GL_POLYGON
        gl.glBegin(gl.GL_POLYGON);
             gl.glVertex3f(WIDTH/6, HEIGHT/6, 0.0f);
             gl.glVertex3f(5*WIDTH/6, HEIGHT/6, 0.0f);
             gl.glVertex3f(5*WIDTH/6,5*HEIGHT/6, 0.0f);
             gl.glVertex3f(WIDTH/6, 5*HEIGHT/6, 0.0f);
         gl.glEnd();
        gl.glPopMatrix();   // termina isolamento de transformações neste objeto

        //Atualiza fatores de rotação
        xrot += xspeed;
        yrot += yspeed;
        zrot += yspeed;
   }

  // Desenha hexágono com segmentos de rectas sem efeito escada
  public void drawHexagon(double x0, double y0) {
    gl.glPushMatrix();   // para isolar as transformações no quadrado
        gl.glTranslatef(3*WIDTH/6, 3*HEIGHT/6, 0.0f); // Move para ponto pivot
        gl.glRotatef(xrot, 1.0f, 0.0f, 0.0f);   //X
        gl.glTranslatef(-3*WIDTH/6, -3*HEIGHT/6, 0.0f); // devolve a posição de origem

        // ponto inicial
        x0 = (int)(r*Math.cos(theta)+cx);
        y0 = (int)(r*Math.sin(theta)+cy);

        for (int i = 1; i<=5; i++) {
            int x = (int)(r*Math.cos(theta+2*i*Math.PI/5)+cx);
            int y = (int)(r*Math.sin(theta+2*i*Math.PI/5)+cy);
            gl.glColor3f(1.0f, 1.0f, 1.0f);
            gl.glPointSize(1);
            antialiasedLine((int)x0, (int)y0, x, y);
            x0 = x;
            y0 = y;
        }
    gl.glPopMatrix();
  }
```

```
// Controla a intensidade luminosa do pixel para evitar efeito escada
 void IntensifyPixel(int x, int y, float D, int flag) {
   float d, r1, g1, b1;

   if (D<0) {
      d = -D; // negativo se o pixel está acima da linha
   } else {
      d = D;
   }
  // calcula a intensidade do pixel conforme a distância d
   r1 = (float)(1-d/2.5f);
   g1 = (float)(1-d/2.5f);
   b1 = (float)(1-d/2.5f);

   gl.glColor3f(r1, g1, b1);
   writepixel(x, y, flag);
}

 // 1º Viewport
 public void viewPort1() {
   int w = WIDTH, h = HEIGHT;

   // executa limpeza de buffer da cor, é feito apenas no 1º viewport!
   // Se não for feito, o quadrado azul borra o fundo...
   gl.glClear (GL.GL_COLOR_BUFFER_BIT);

   gl.glViewport(0, h/2, w/2, h/2);
   gl.glMatrixMode(GL.GL_PROJECTION);
   gl.glLoadIdentity();

   gl.glOrtho(0, WIDTH, 0, HEIGHT, -500.0f, 500.0f);
   drawScene();

   gl.glMatrixMode(GL.GL_MODELVIEW);
   gl.glLoadIdentity();

   // É necessário mandar limpar o depth buffer, senão, nada aparece no viewport!
   gl.glClear (GL.GL_DEPTH_BUFFER_BIT);
}

 // 2º Viewport
 public void viewPort2() {
    int w = WIDTH, h = HEIGHT;

    gl.glViewport(w/2, 0, w/2, h/2);
    gl.glMatrixMode(GL.GL_PROJECTION);
    gl.glLoadIdentity();
    gl.glOrtho(0, WIDTH, 0, HEIGHT, -500.0f, 500.0f);
    drawScene();

    gl.glMatrixMode(GL.GL_MODELVIEW);
    gl.glLoadIdentity();

   // É necessário mandar limpar o depth buffer, senão, nada apareceno viewport!
    gl.glClear (GL.GL_DEPTH_BUFFER_BIT);
}
```

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

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

  }
}
```