 Escola Superior de Tecnologia e Gestão Instituto Politécnico da Guarda	ENUNCIADO DE AVALIAÇÃO	Modelo PED.002.02
---	-------------------------------	------------------------------------

Curso	Engenharia Informática					Ano letivo	2019/2020
Unidade curricular	Computação Gráfica						
Ano curricular	3º	Semestre	1º S	Data	14/02/2020	Duração	2h30

Exame Normal

Este exame tem um peso de 60% na nota final.

1. Escreva uma função em Java para desenhar a curva dada pelas equações
 (2) parametricas indicadas ao lado.
- $$x = x_0 + a \cos t$$
- $$y = y_0 + b \sin t$$
- $$\min < t < \max$$

Resposta:

```
// Since we want to write a function, the variables of the equations should
// be passed as arguments.
void drawCurve(Graphics2D g2, int x0, int y0, float a, float b, float tMin, float tMax, int n)
{
    // Calculate the first point for t = tMin.
    float t = tMin;
    int x1 = (int) (x0 + a * Math.cos(t));
    int y1 = (int) (y0 + b * Math.sin(t));
    int x2;
    int y2;

    // Calculate the increment for t, considering the number of points.
    float dt = (tMax - tMin) / n;

    for (int i = 1; i <= n; i++) {
        // Increment t to calculate the next point.
        t = i * dt;

        // Calculate the new point.
        x2 = (int) (x0 + a * Math.cos(t));
        y2 = (int) (y0 + b * Math.sin(t));

        // Draw a line connecting the new point with the last point.
        g2.drawLine(x1, y1, x2, y2);

        // The new point becomes the last point.
        x1 = x2;
        y1 = y2;
    }
}
```

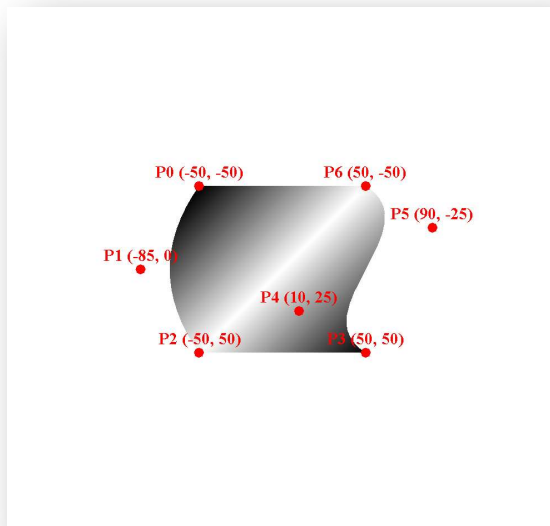
2. Desenhe um esboço que represente o resultado do seguinte código. Indique no esboço as coordenadas
 (2) de todos os pontos de controle.

```
public void paintComponent(Graphics g) {
    super.paintComponent(g);
    Graphics2D g2 = (Graphics2D) g;
    g2.translate(getWidth()/2, getHeight()/2);
    GeneralPath p = new GeneralPath(GeneralPath.WIND_EVEN_ODD);
    p.moveTo(-50f, -50f);
    p.quadTo(-85f, 0f, -50f, 50f);
    p.lineTo(50, 50);
    p.curveTo(10, 25, 90, -25, 50, -50);
    p.lineTo(-50, -50);
    p.closePath();
    GradientPaint paint = new GradientPaint(-50, -50, Color.BLACK, 0, 0, Color.WHITE,
    true);
}
```



```
g2.setPaint (paint) ;  
g2.fill (p) ;  
}
```

Resposta:



3. Desenhe um esboço que represente o resultado do seguinte código. Indique no esboço os eixos do sistema de coordenadas.
(2)

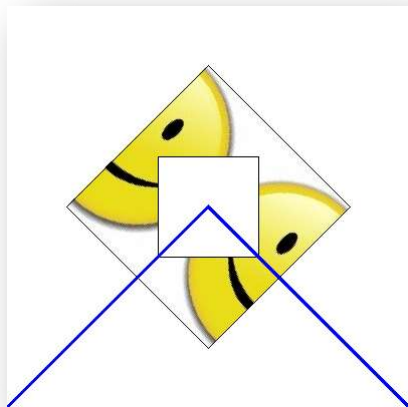
```
public void paintComponent(Graphics g) {  
    super.paintComponent(g);  
    Graphics2D g2 = (Graphics2D) g;  
    int w = this.getWidth();  
    int h = this.getHeight();  
    int l = 100;  
    BufferedImage image = getImage(("test2d/images/Smile.jpg"));  
    Paint paint = new TexturePaint(image, new Rectangle2D.Double(-2*l, -l, 2*l, 2*l));  
    Area a = new Area(new Rectangle2D.Double(-l, -l, 2*l, 2*l));  
    Area b = new Area(new Rectangle2D.Double(-l/2, -l/2, l, l));  
  
    AffineTransform at = new AffineTransform();  
    at.rotate(Math.toRadians(45));  
    b=b.createTransformedArea(at);  
    a.subtract(b);  
  
    g2.translate(w/2, h/2);  
    g2.rotate(Math.toRadians(45));  
  
    g2.setPaint (paint);  
    g2.fill(a);  
    g2.setColor (Color.black);  
    g2.draw(a);  
}
```



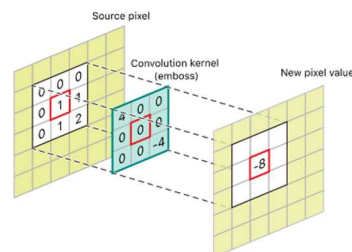


Resposta:

As linhas azuis representam os eixos.



4. A operação de convolução aplica um filtro específico a uma imagem. O filtro é representado por uma matriz chamada Kernel (geralmente com tamanho 3x3). Dê um exemplo de um kernel para aplicar um filtro que causa o chamado efeito de suavização (usado para eliminar “grainha” na imagem, por exemplo) e explique sucintamente como é que o filtro funciona e consegue suavizar a imagem.



Resposta:

O kernel do filtro de suavização é uma matriz 3x3 com todos os valores iguais a 1/9.

$$\begin{bmatrix} \frac{1}{9} & \frac{1}{9} & \frac{1}{9} \\ \frac{1}{9} & \frac{1}{9} & \frac{1}{9} \\ \frac{1}{9} & \frac{1}{9} & \frac{1}{9} \end{bmatrix}$$

A operação de convolução transforma o valor de um dado pixel calculando o somatório da multiplicação dos valores do kernel pelos correspondentes valores dos vizinhos ao redor do pixel e do próprio pixel. Se esse cálculo for feito com o kernel do filtro de suavização apresentado na matriz anterior, o valor resultante é a média aritmética de todos os valores dos pixels. Isto é, o valor do pixel que está a ser processado passa a ser o valor da média dos seus vizinhos e dele próprio. Isso faz com que o pixel fique com um valor semelhante aos valores dos seus vizinhos e isso suaviza a imagem. Por exemplo, num caso extremo em que o pixel tem um valor completamente diferente dos valores dos seus vizinhos (é um pixel de “grainha”), ao ser aplicado o filtro, o pixel passará a ter um valor parecido com os seus vizinhos e o pixel de grainha desaparece.

5. Considere que se pretende aplicar um fator de escala (E_x , E_y) a um objeto 2D qualquer, mas de modo a que as coordenadas do centro do objeto (X_c , Y_c), não se alterem com a transformação. Isto é, pretende-se que o objeto fique centrado no mesmo ponto, após a aplicação da transformação. Apresente a matriz 3x3 da transformação composta que ao ser multiplicada pelos pontos do objeto produz o resultado desejado.

Resposta:



$$TC = \begin{bmatrix} 1 & 0 & X_c \\ 0 & 1 & Y_c \\ 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} Ex & 0 & 0 \\ 0 & Ey & 0 \\ 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} 1 & 0 & -X_c \\ 0 & 1 & -Y_c \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} Ex & 0 & X_c - Ex.X_c \\ 0 & Ey & Y_c - Ey.Y_c \\ 0 & 0 & 1 \end{bmatrix}$$

A transformação composta (TC) corresponde a aplicar três transformações em sequência. Primeiro é aplicada uma translação com os valores $(-X_c, -Y_c)$ para que o ponto central coincida com a origem. Depois é aplicada a variação de escala com os valores (Ex, Ey) . Como o ponto central coincide com a origem, a variação de escala não afetará este ponto. Por fim é aplicada uma translação com os valores (X_c, Y_c) para voltar a colocar o ponto central na posição original.

6. Complete o código a seguir para criar uma animação na qual um objeto quadrado roda em torno do centro da janela numa orbida de raio igual a 100.

```
class MyPanel extends JPanel implements { // 1°
    float ang = 0;

    public MyPanel() {
        setPreferredSize(new Dimension(400, 400));
    }

    public void paintComponent(Graphics g) {
        super.paintComponent(g);
        Graphics2D g2 = (Graphics2D) g;

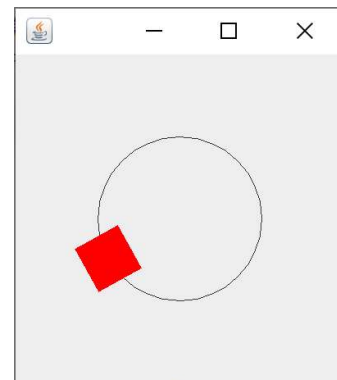
        g2.drawOval(-100, -100, 200, 200);

        Shape s = new Rectangle2D.Double(-30, -30, 60, 60);
        AffineTransform tx = new AffineTransform();
        tx. // 2°
        tx. // 3°
        s = // 4°

        g2.setPaint(Color.RED);
        g2.fill(s);
    }

    public void { // 5°
        while (true) { // 6°
            // 7°

            try {
                Thread.sleep(50);
            } catch (InterruptedException e) {
                e.printStackTrace();
            }
        }
    }
}
```



Resposta:

```
1° Runnable
2° Thread thread = new Thread(this);
3° thread.start();
4° g2.translate(200, 200);
5° tx.rotate(Math.toRadians(ang));
6° tx.translate(100, 0);
7° s = tx.createTransformedShape(s);
8° run()
9° ang = (ang + 1) % 360;
10° repaint();
```



7. Desenhe a figura correspondente à geometria definida pelo código a seguir. Deve indicar no desenho as coordenadas e os índices dos vértices. Desenhe também as normais da geometria.

```
GeometryInfo gi = new GeometryInfo(GeometryInfo.TRIANGLE_STRIP_ARRAY);  
int[] stripIndex = { 4, 4, 4 };  
gi.setStripCounts(stripIndex);  
Point3f[] giCoords = new Point3f[7];  
giCoords[0] = new Point3f(0f, 0f, 0f);  
giCoords[1] = new Point3f(0f, 0f, 1f);  
giCoords[2] = new Point3f(0f, 1f, 1f);  
giCoords[3] = new Point3f(0f, 1f, 0f);  
giCoords[4] = new Point3f(1f, 1f, 0f);  
giCoords[5] = new Point3f(1f, 0f, 0f);  
giCoords[6] = new Point3f(1f, 0f, 1f);  
gi.setCoordinates(giCoords);  
int[] giIndices = { 0, 3, 1, 2, 0, 5, 3, 4, 1, 6, 0, 5};  
gi.setCoordinateIndices(giIndices);
```

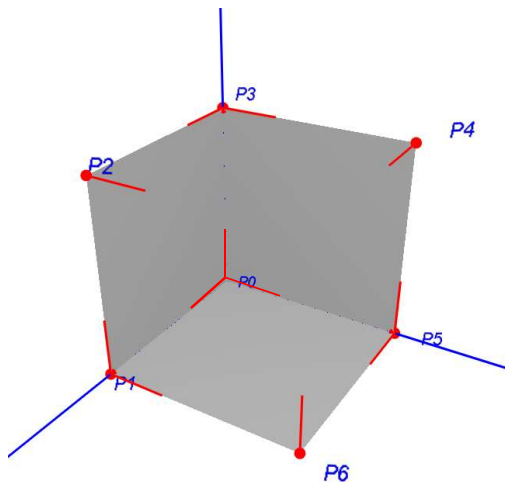
```
NormalGenerator ng = new NormalGenerator();  
ng.generateNormals(gi);
```

Considere que a cena é iluminada por uma luz ambiente de cor branca e o objeto usa a seguinte aparência.

```
Appearance ap = new Appearance();  
ap.setMaterial(new Material());
```

Resposta:

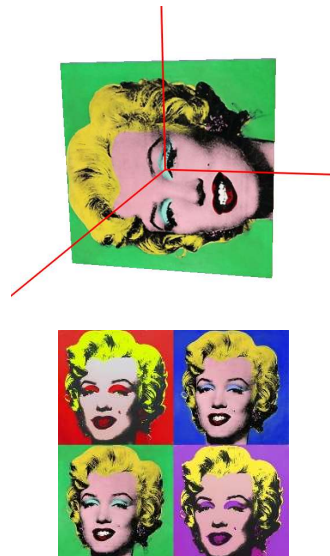
Como a aparência é baseada no material por defeito e na luz, as faces são renderizadas com o tom típico do material por defeito. As linhas vermelhas representam as normais da geometria.





8. Complete o código a seguir para criar um painel quadrado, onde é mapeada uma parte da imagem de textura, conforme mostrado na figura ao lado. Considere que todos os vertices do painel têm coordenada $z = 0$;

```
QuadArray p = new QuadArray(4, QuadArray.COORDINATES
); //1°
p.setCoordinate(0, new Point3d(-1, -1, 0));
p.setCoordinate(1, new Point3d(    )); //2°
p.setCoordinate(2, new Point3d(    )); //3°
p.setCoordinate(3, new Point3d(    )); //4°
p.setTextureCoordinate(0,0,new TexCoord2f(    )); //5°
p.setTextureCoordinate(0,1,new TexCoord2f(    )); //6°
p.setTextureCoordinate(0,2,new TexCoord2f(    )); //7°
p.setTextureCoordinate(0,3,new TexCoord2f(    )); //8°
```



Resposta:

1° | QuadArray.**TEXTURE_COORDINATE_2**

As coordenadas dos vértices devem ser indicadas seguindo o sentido contrário aos ponteiros do relógio para que a face renderizada seja a face virada para nós.

2° 1, -1, 0

3° 1, 1, 0

4° -1, 1, 0

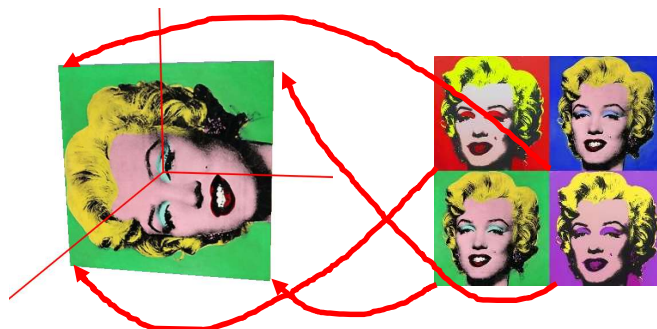
De modo a mapear o pedaço da imagem de textura como se pretende, é necessário seguir a seguinte ordem para as coordenadas de textura que corresponde à correspondência ilustrada na seguinte figura.

5° 0f, 0.5f

6° 0f, 0f

7° 0.5f, 0f

8° 0.5f, 0.5f



9. Desenhe o grafo de cena de um programa que usa um objeto DistanceLOD para configurar três níveis de detalhes para exibir um objeto de texto 3D da string "Java". O primeiro nível usa uma aparência do tipo Material para iluminar o objeto com luz. O segundo nível usa uma aparência com base numa cor única. O terceiro nível substitui o objeto por uma Box com a mesma aparência usada no segundo nível. Considere também que existe um objeto MouseRotate para interagir com o objeto.

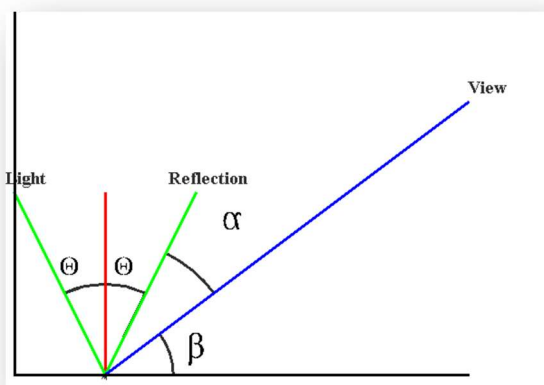


10. Considere que um ponto de coordenadas (1, 0, 0) na superfície de um objeto tem um normal na direção (0, 1, 0) e é iluminado por uma luz com intensidade (1, 1, 0), localizada em (0, 2, 0). A vista está posicionada em (5, 3, 0). Se os coeficientes de reflexão especular do material escolhido para a aparência do objeto são (0,3, 0,5, 0,2) e o índice de brilho é 94, qual é o valor da reflexão especular nesse ponto? Não é necessário calcular os valores, apenas apresentar as equações. Como caracterizaria o material deste objeto? Muito brilhante ou pouco brilhante? Justifique.

$$I = I_a K_a + I_p K_d \cos \theta + I_p K_s \cos^n \alpha$$

Resposta:

A seguinte figura ilustra o modelo de Phong da situação descrita no enunciado.



A componente da reflexão especular é dada pela equação

$$I_s = I_p K_s \cos^n \alpha$$

A qual se desdobra nas 3 componentes Red, Green e Blue

$$\begin{aligned} I_s^{Red} &= I_p^{Red} K_s^{Red} \cos^n \alpha \\ I_s^{Green} &= I_p^{Green} K_s^{Green} \cos^n \alpha \\ I_s^{Blue} &= I_p^{Blue} K_s^{Blue} \cos^n \alpha \end{aligned}$$

O enunciado diz-nos que

$$\begin{aligned} I_p &= (1, 1, 0) \\ K_s &= (0.3, 0.5, 0.2) \end{aligned}$$

Assim, as componentes da reflexão especular são as seguintes:

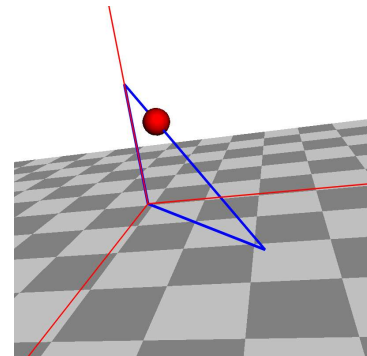
$$\begin{aligned} I_s^{Red} &= 1 \times 0.3 \times \cos^{94} \alpha \\ I_s^{Green} &= 1 \times 0.5 \times \cos^{94} \alpha \\ I_s^{Blue} &= 0 \times 0.2 \times \cos^{94} \alpha \end{aligned}$$

Com

$$\alpha = 90 - \theta - \beta = 90 - \tan^{-1} \frac{1}{2} - \tan^{-1} \frac{3}{5}$$



11. Complete o código a seguir para programar um comportamento de
(2) PositionPathInterpolator que faz com que uma esfera siga o caminho triangular mostrado na figura ao lado. Pretende-se que a esfera leve 4 segundos para percorrer a hipotenusa do triângulo e 3 segundos para percorrer cada uma das pernas do triângulo. O movimento da esfera será sempre na mesmo sentido? Justifique.



```
...
Appearance ap = //1°
Primitive sphere = new Sphere(0.05f, //2°
Primitive. //2°
, 24, ap);

TransformGroup t = new TransformGroup();
t. //3°
t.addChild(sphere);
root.addChild(t);

Alpha a = new Alpha(-1, 10000);
a.setMode(Alpha.INCREASING_ENABLE |
Alpha.DECREASING_ENABLE);
a.setDecreasingAlphaDuration(10000);
float[] k = //4°
Point3f[] p = {new Point3f(0f, 1f, 0f), } //5°

tr = //6°
PositionPathInterpolator i = new
PositionPathInterpolator(a, t, tr, k, p); //7°
i. //8°
t. //9°
...
```

Resposta:

- 1° new Appearance();
- 2° GENERATE_NORMALS
- 3° setCapability(TransformGroup.ALLOW_TRANSFORM_WRITE);

Quando a esfera inicia o movimento o $\alpha = 0$, no final da hipotenusa é 0.4 (correspondente a 4 segundos), no final do cateto é $0.4 + 0.3$ (4 + 3 segundos) e no final do outro cateto é $0.4 + 0.3 + 0.3 = 1$ (4 + 3 + 3 segundos).

4° {0f, 0.4f, 0.7f, 1.0f};

5° new Point3f(1f, 0f, 1f), new Point3f(0f, 0f, 0f), new Point3f(0f, 1f, 0f)

6° new Transform3D();


7° Era o tr, mas já lá está.

8° setSchedulingBounds(bounds);

9° addChild(i)

O movimento da esfera não será sempre no mesmo sentido porque o α tem uma fase de incremento e uma fase de decremento. Assim, sempre que a esfera termina de percorrer o triângulo num dado sentido, inverte o sentido e inicia novo percurso.

12. A API Java 3D permite implementar objetos com geometrias que podem ser alteradas dinamicamente durante a execução da aplicação. Isso pode ser feito de um modo mais restritivo através da classe **Morph**, ou de um modo mais genérico e personalizado através da implementação da classe interface **GeometryApdater** e da criação de uma classe que implementa um **behavior** personalizado.
- (1)

 <p>Escola Superior de Tecnologia e Gestão Instituto Politécnico da Guarda</p>	<h2 style="text-align: center;">ENUNCIADO DE AVALIAÇÃO</h2>	<p style="text-align: right;">Modelo PED.002.02</p>
---	---	---

Considere o segundo modo e descreva sucintamente, por palavras suas, como se pode implementar esse segundo modo com base nas duas classes referidas.

Para implementar uma geometria dinâmica devemos armazenar as suas coordenadas num array e passar uma referência desse array para a classe escolhida para criar a geometria. Depois, a maneira correta de fazer alterações nas coordenadas armazenadas no array é implementar a interface GeometryUpdater e programar a alteração do array da geometria no método updateData() que será executado a pedido pelo motor de renderização. Para efetuar esse pedido, é implementado um behavior personalizado que ao ser ativado por uma condição de despertar (por exemplo terem passado X frames), chama a função updateData() da classe GeometryArray escolhida para criar a geometria. Notar que as funções têm o mesmo nome, mas pertencem a classes diferentes e uma implementa o algoritmo que altera o array da geometria e a outra dá a ordem ao motor de renderização para proceder à alteração, mas será o motor de renderização a executar a alteração em sincronia com a renderização da cena.