Examples of types of questions about Computer Graphics and Java 2D.

Cap 1 – Introdução

1. Identify the main fields of Computer Graphics
A–Image Synthesis, B–Image Processing, C–Image Analysis and D–Computational Visualization that are applicable to the following applications

1. Locate small bright spots in a mammogram image.
2. Display the graphics of a computer game.
3. Display a simulation of the solar system with the sun and nine planets in motion
4. Recognize the brain region in a MRI scan and display a 3D model of the brain.
5. Eliminate the "Red eye" effect in photographs.
6. Make a computer identification of a person from a photograph.

More than one field can be applicable to a single application. Give the answer in the form
1 – C
2 – A
. . .

1 – B, C
2 – A
3 – A
4 – D
5 – B
6 – B, C

Cap. 2 – Desenho de primitivas e Shape

2. Write an algorithm in pseudocode or Java to draw an ellipse based on the following parametric equations.

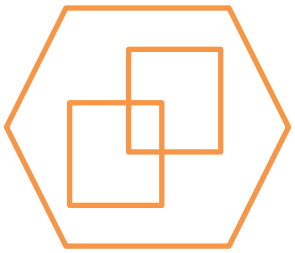
$$x = x_0 + a \cos t$$

$$y = y_0 + b \sin t$$

Java

```
int a = 50, b = 100;
int t0 = 0, tn = 2 * Pi;
int x1 = x0 + a * Math.cos(t0);
int y1 = y0 + b * Math.sin(t0);
int x2 = 0;
int y2 = 0;
int n = 100;
float dt = (tn - t0)/n;
for (i=1; i < n; i++) {
   t = i * dt;
   x2 = x0 + a * Math.cos(t);
   y2 = y0 + b * Math.sin(t);
   g2.awLine(x1, y1, x2, y2);
   x1 = x2;
   y1 = y2;
}
```

3. Determine the interior of the shape in the next figure using the even–odd rule.

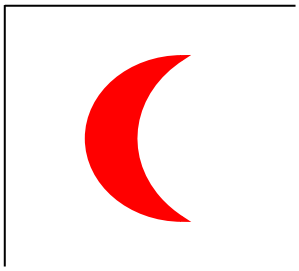


4. Draw a sketch that represents the output of the following code.

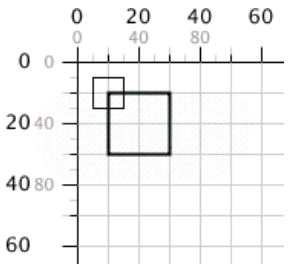
```
public void paintComponent(Graphics g) {
    super.paintComponent(g);
    Graphics2D g2 = (Graphics2D)g;
    g2.translate(100, 50);
    GeneralPath path = new GeneralPath(GeneralPath.WIND_EVEN_ODD);
    path.moveTo(150,0);
    path.curveTo(-50, 50, -50, 250, 150, 300);
    path.quadTo(50, 150, 150, 0);
    path.closePath();
    g2.fill(path);
  }
```

After translation:

```
    path.moveTo(250,50);
    path.curveTo(50, 100, 50, 300, 250, 350);
    path.quadTo(150, 200, 250, 50);
    path.closePath();
```



5. The following figure represents a scale transformation applied to the small square. Present the matrix of a compose transformation to apply the scale transformation but without changing the coordinates of the up-left corner.



$CT = T_{(10, 10)} \times S_{(2, 2)} \times T_{(-10, -10)}$

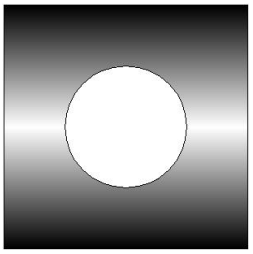
$$CT = \begin{bmatrix} 1 & 0 & 10 \\ 0 & 1 & 10 \\ 0 & 0 & 1 \end{bmatrix} . \begin{bmatrix} 2 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 1 \end{bmatrix} . \begin{bmatrix} 1 & 0 & -10 \\ 0 & 1 & -10 \\ 0 & 0 & 1 \end{bmatrix}$$

6. Write the Java code to put in paintComponent() to display a square centered at the center of window and rotated by 45°.

```
//By transforming the coordinates space
//g2.translate(200, 200);
//g2.rotate(Math.toRadians(45));
g2.rotate(Math.toRadians(45), 200, 200);
g2.drawRect(100, 100, 200, 200);

//By transforming a shape
/*
Shape s = new Rectangle2D.Float(-100, -100, 200, 200);
AffineTransform at = new AffineTransform();
at.setToIdentity();
at.translate(200, 200);
at.rotate(Math.toRadians(45));
g2.draw(at.createTransformedShape(s));
*/
```

7. Draw a sketch that represents the output of the following code.

```
public void paintComponent(Graphics g) {
    super.paintComponent(g);
    Graphics2D g2 = (Graphics2D)g;
    int w = this.getWidth();
    int h = this.getHeight();
    int r = Math.min(w, h) - 100;
    Area a = new Area(new Rectangle(r, r));
    a.subtract(new Area(new Ellipse2D.Double(r/4, r/4, r/2, r/2)));
    g2.translate((w-r)/2, (h-r)/2);
    GradientPaint paint = new GradientPaint(0, 0, Color.black, 0, r/2,
    Color.white, true);
    g2.setPaint(paint);
    g2.fill(a);
    g2.setColor(Color.black);
    g2.draw(a);
  }
```



8. The image on the right represents the data processed by the convolution operation applied to an image. The Iij are the original intensity of the pixels of an image and Kij are the values of a specific kernel. Preset the equation of the convolution operation to calculate the final intensity value of pixel (2, 2).

Imagem

I_{11}	I_{12}	I_{13}	I_{14}	I_{15}	I_{16}
I_{21}	I_{22}	I_{23}	I_{24}	I_{25}	I_{26}
I_{31}	I_{32}	I_{33}	I_{34}	I_{35}	I_{36}
I_{41}	I_{42}	I_{43}	I_{44}	I_{45}	I_{46}
I_{51}	I_{52}	I_{53}	I_{54}	I_{55}	I_{56}
I_{61}	I_{62}	I_{63}	I_{64}	I_{65}	I_{66}

Kernel

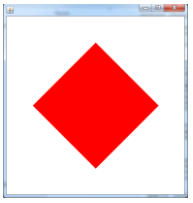
K_{11}	K_{12}	K_{13}
K_{21}	K_{22}	K_{23}
K_{31}	K_{32}	K_{33}

$$O_{22} = I_{11} \times K_{11} + I_{12} \times K_{12} + I_{13} \times K_{13} + I_{21} \times K_{21} + I_{22} \times K_{22} + I_{23} \times K_{23} + I_{31} \times K_{31} + I_{32} \times K_{32} + I_{33} \times K_{33}$$

9. Complete the following function to create an output image by inverting the color of an input image. The color of every pixel is turned to its opposite color; that is, a color with components r, g, b is changed to 255 – R, 255 – G, 255 – B.

```
BufferedImage RGBInvert(BufferedImage imgIn) {
  BufferedImage imgOut = new BufferedImage(imgIn.getWidth(),
                             imgIn.getHeight(), imgIn.getType());

  //1° Create objects to access pixel data
  WritableRaster rasterImgIn = imgIn.getRaster();
  WritableRaster rasterImgOut = imgOut.getRaster();

  int rgba[] = new int[4];
  for (int x = 0; x < imgIn.getWidth(); x++) {
    for (int y = 0; y < imgIn.getHeight(); y++) {
      //2° Get the color of each pixel
      rasterImgIn.getPixel(x, y, rgba);

      //3° Transform the color
      rgba[0] = 255 - rgba[0];
      rgba[1] = 255 - rgba[1];
      rgba[2] = 255 - rgba[2];

      //4° Set the new pixel color
      rasterImgOut.setPixel(x, y, rgba);
    }
  }

  return imgOut;
}
```

10. Complete the following code to create an animation where a shape of a square centered at the origin rotates continuously.



```
R:
class MyPanel extends JPanel implements Runnable {

        int ang = 0; // 1°
        public MyPanel() {
                setPreferredSize(new Dimension(400, 400));
                setBackground(Color.white);
                Thread thread = new Thread(this); // 2°;
                thread.start(); // 3°;
        }
```

```
	public void paintComponent(Graphics g) {
		super.paintComponent(g);
		Graphics2D g2 = (Graphics2D) g;
		int w = getWidth();
		int h = getHeight();
		AffineTransform tx = new AffineTransform();
		tx.rotate(Math.toRadians(ang));
		tx.scale(0.5, 0.5);// 4°
		Rectangle2D rect = new Rectangle2D.Double(-200, -200, 400, 400);
		Shape shape = tx.createTransformedShape(rect); // 5°
		g2.translate(200, 200); // 6°
		g2.setColor(Color.RED); // 7°
		g2.fill(shape);
	}

	@Override
	public void run() {
		while (true) {
			ang += 5 % 360; // 8°;
			repaint(); // 9°
			try {
				Thread.sleep(50);
			} catch (InterruptedException e) {
				e.printStackTrace();
			}
		}
	}
}
```