Técnicas de Programação II

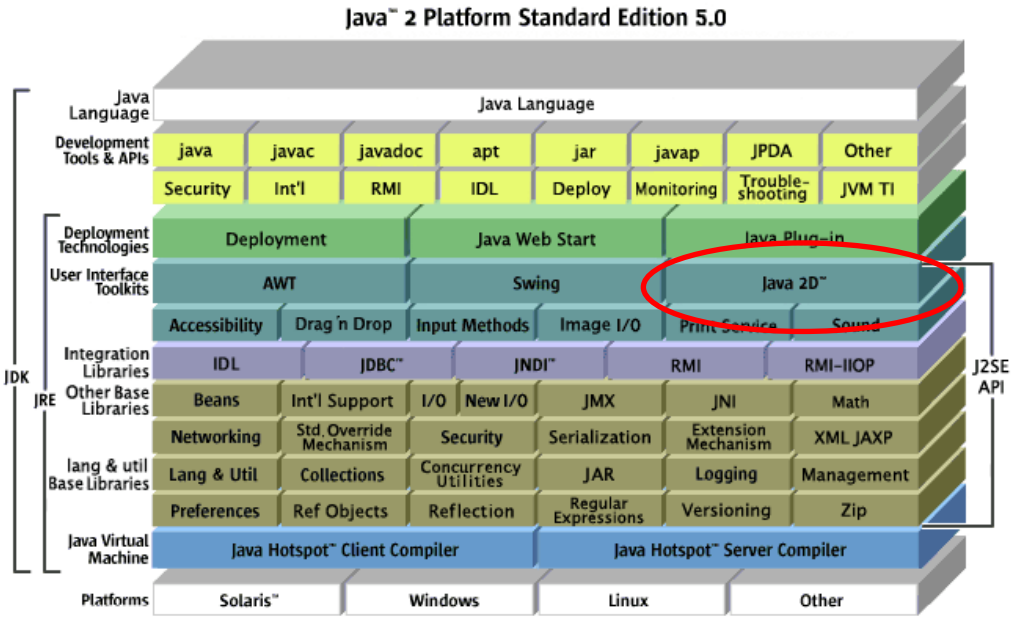
Aula 05 – Java 2D

Edirlei Soares de Lima
<edirlei.lima@uniriotec.br>

Java 2D

- Java 2D é uma API da linguagem Java que fornece funcionalidades básicas para o desenho de objetos gráficos 2D.

- Prove um conjunto de funções para a renderização de formas geométricas básicas (linhas, arcos, retângulos, etc.) e imagens.



Java 2D – Programa Base

CanvasPanel.java

```
package Java2D;
import java.awt.Color;
import java.awt.Graphics;
import java.awt.Graphics2D;
import javax.swing.JPanel;

public class CanvasPanel extends JPanel implements Runnable{
    public CanvasPanel()
    {
        setDoubleBuffered(true);
        setFocusable(true);
        load();
        new Thread(this).start();
    }
    @Override
    public void paintComponent(Graphics g)
    {
        super.paintComponent(g);
        draw(g);
    }
    ...
```

Java 2D – Programa Base

CanvasPanel.java

```
...
@Override
public void run()
{
    double btime, dtime = 0;
    btime = System.currentTimeMillis();
    while(true)
    {
        update(dtime/1000);
        repaint();
        try {
            Thread.sleep(1);
        } catch (InterruptedException e) {
            System.out.println(e.getMessage());
        }
        dtime = (System.currentTimeMillis() - btime);
        btime = System.currentTimeMillis();
    }
}
...
```

Java 2D – Programa Base

CanvasPanel.java

```
    ...

    private void load()
    {
        setBackground(Color.BLACK);
    }

    private void update(double dt)
    {

    }

    private void draw(Graphics g)
    {
        Graphics2D g2d = (Graphics2D)g;

        g2d.setColor(Color.WHITE);
        g2d.drawString("Hello World!", 360, 300);
    }
}
```

Java 2D – Programa Base

MainFrame.java

```
package Java2D;

import javax.swing.JFrame;
import javax.swing.SwingUtilities;

public class MainFrame extends JFrame
{

    public MainFrame()
    {
        setTitle("Java 2D");
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        add(new CanvasPanel());
        setSize(800, 600);
        setLocationRelativeTo(null);
    }

    ...
```

Java 2D – Programa Base

MainFrame.java

```
    ...

    public static void main(String[] args)
    {
        SwingUtilities.invokeLater(new Runnable(){
            @Override
            public void run(){
                new MainFrame().setVisible(true);
            }
        });
    }
}
```

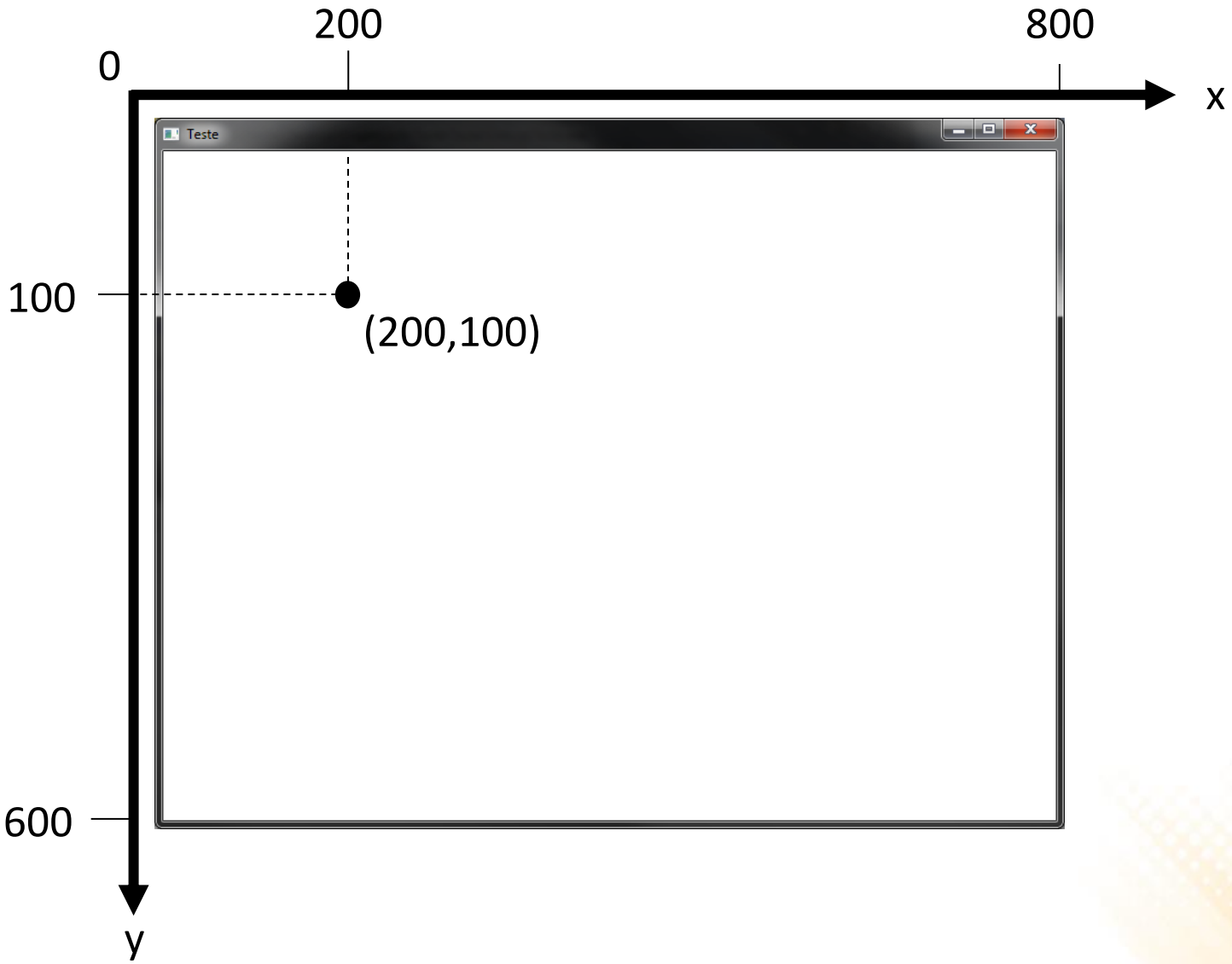
Java 2D – Hello World


```
private void load()
{
  setBackground(Color.BLACK);
}

private void draw(Graphics g)
{
  Graphics2D g2d = (Graphics2D)g;

  g2d.setColor(Color.WHITE);
  g2d.drawString("Hello World!", 360, 300);
}
```

Java 2D – Coordenadas de Tela



Método `draw`

- É método `draw` é executado sempre que um novo frame precisa ser renderizado.
- Ele é executado continuamente para a renderização dos frames que serão exibidos na tela.
- Todas as funções para desenho de objetos gráficos deve ser executadas no método `draw`.

```
private void draw(Graphics g)
{
  Graphics2D g2d = (Graphics2D)g;
  g2d.setColor(Color.WHITE);
  g2d.drawString("Hello World!", 360, 300);
}
```

Método `load`

- É método `load` é executado apenas uma vez no momento que o programa é iniciado.

- A função é geralmente usada para:
  - Carregar recursos (imagens, áudio, etc.)
  - Inicializar variáveis
  - Definir configurações

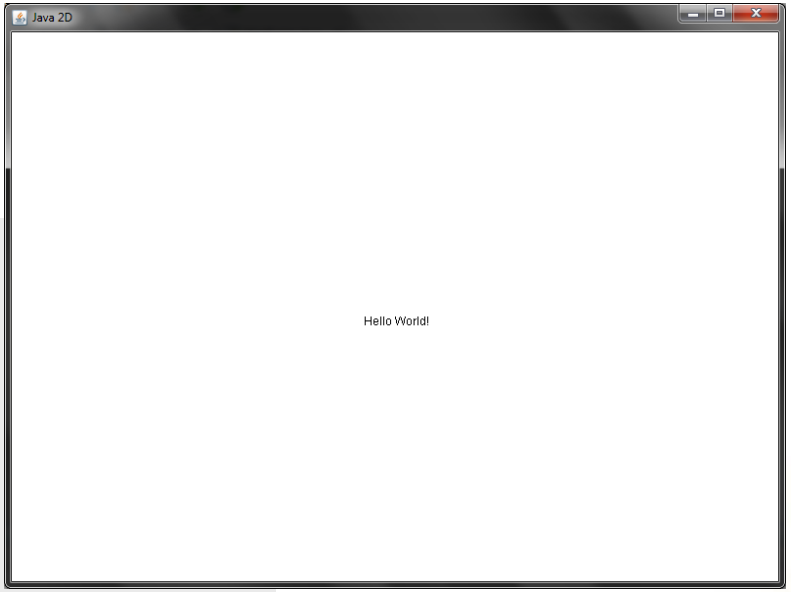
```
private void load()
{
  setBackground(Color.BLACK);
}
```

De Volta ao “Hello World”

```
private void load()
{
  setBackground(Color.WHITE);
}

private void draw(Graphics g)
{
  Graphics2D g2d = (Graphics2D)g;

  g2d.setColor(Color.BLACK);
  g2d.drawString("Hello World!", 360, 300);
}
```



Método `update(double dt)`

- O método `update(double dt)` é continuamente executado em loop enquanto o programa estiver aberto. O parâmetro `dt` indica o tempo que se passou desde a última vez que essa função foi chamada (usualmente um valor bem pequeno)

- A função é geralmente usada para:
  - Animação
  - Cálculos de física
  - Inteligência artificial de inimigos

Calcula o deslocamento em X de forma independente da velocidade de execução do programa

```
private void update(double dt)
{
  px = px + (100 * dt);
}
```

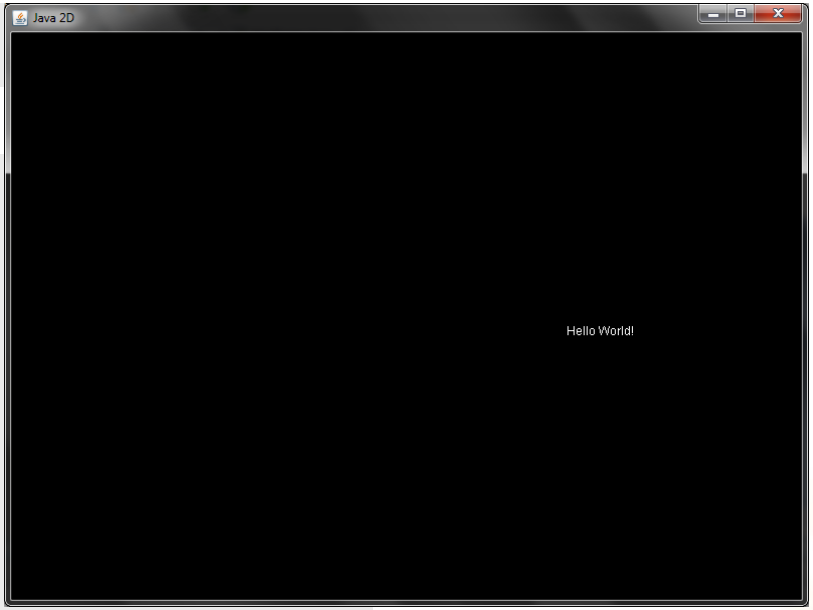
De Volta ao “Hello World”

```
private double px = 0;
```

```
private void load()
{
  setBackground(Color.BLACK);
}

private void update(double dt)
{
  px = px + (100 * dt);
}

private void draw(Graphics g)
{
  Graphics2D g2d = (Graphics2D)g;
  g2d.setColor(Color.WHITE);
  g2d.drawString("Hello World!", (int)px, 300);
}
```



Classe `Graphics2D`

- A classe `Graphics2D` (que é derivada da classe `Graphics`), contem diversas funções dedicadas a operações gráficas:
  - Desenho de linhas, formas geométricas, texto, imagens, etc.

- É possível consultar a lista completa de funções das classes `Graphics2D e Graphics` nos seguintes endereços:

http://docs.oracle.com/javase/7/docs/api/java/awt/Graphics2D.html

http://docs.oracle.com/javase/7/docs/api/java/awt/Graphics.html

Classe `Graphics2D`

- Desenhando **formas geométricas** básicas:
  - Método para desenhar formas geométricas (contornos):

```
void draw(Shape s)
```

  - Método para desenhar formas geométricas (preenchidas):

```
void fill(Shape s)
```

Classe `Graphics2D`

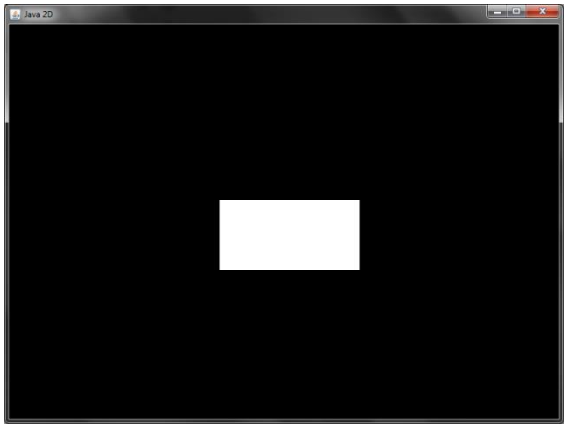
- Desenhando **formas geométricas** básicas:
  - **Retângulo:**

```
Rectangle2D.Double(x, y, width, height)
```

  - Exemplo:

```
g2d.draw(new Rectangle2D.Double(300, 250, 200, 100));
```

```
g2d.fill(new Rectangle2D.Double(300, 250, 200, 100));
```



Classe `Graphics2D`

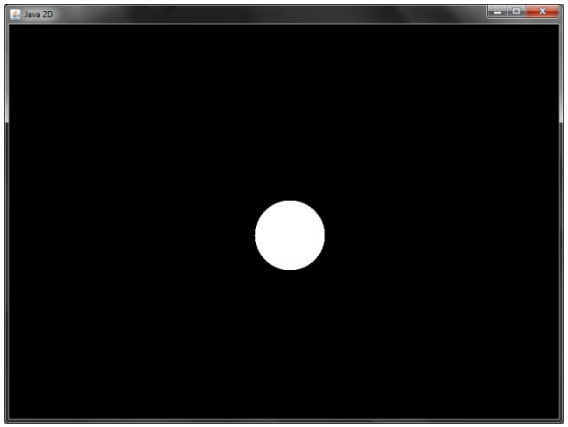
- Desenhando **formas geométricas** básicas:
  - **Elipse:**

```
Ellipse2D.Double(x, y, width, height)
```

  - Exemplo:

```
g2d.draw(new Ellipse2D.Double(350, 250, 100, 100));
```

```
g2d.fill(new Ellipse2D.Double(350, 250, 100, 100));
```



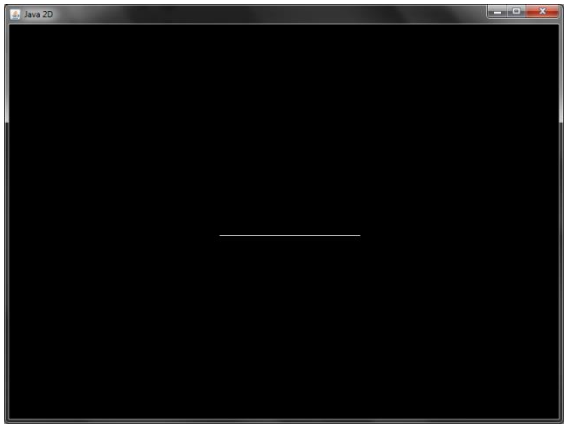
Classe `Graphics2D`

- Desenhando **formas geométricas** básicas:
  - **Linha:**

```
Line2D.Double(x1, y1, x2, y2)
```

  - Exemplo:

```
g2d.draw(new Line2D.Double(300, 300, 500, 300));
```



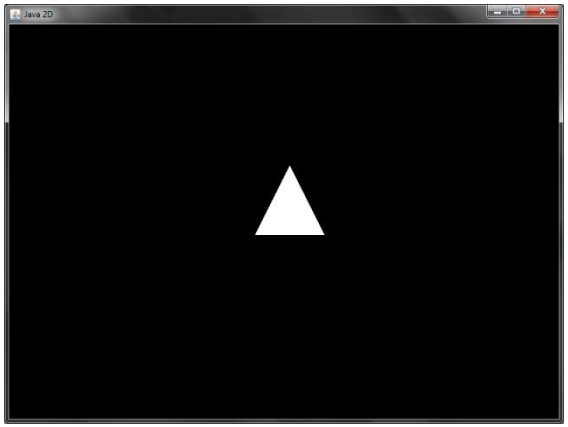
Classe `Graphics2D`

- Desenhando **formas geométricas** básicas:
  - **Polígono:**

```
void fillPolygon(Polygon p)
```

  - Exemplo:

```
int x1Points[] = {350, 450, 400};
int y1Points[] = {300, 300, 200};
g2d.fillPolygon(new Polygon(x1Points, y1Points, 3));
```



Classe `Graphics2D`

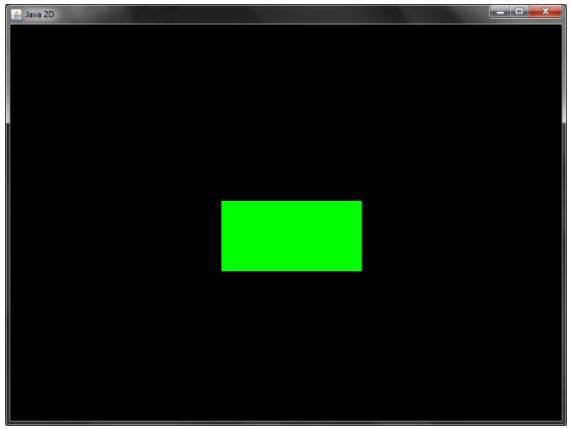
- Desenhando **formas geométricas** básicas:
  - **Modificando a cor das formas geométricas:**

```
void setColor(Color c)
```

  - Exemplo:

```
g2d.setColor(new Color(0, 255, 0));

g2d.fill(new Rectangle2D.Double(300, 250, 200, 100));
```



Exemplo 1 – Primitivas Geométricas

```
private void draw(Graphics g)
{
  Graphics2D g2d = (Graphics2D) g;

  g2d.setColor(new Color(0, 134, 0));
  g2d.fill(new Rectangle2D.Double(100, 100, 600, 400));

  int x1Points[] = {120, 400, 680, 400};
  int y1Points[] = {300, 120, 300, 480};
  g2d.setColor(new Color(252, 252, 0));
  g2d.fillPolygon(new Polygon(x1Points, y1Points, 4));

  g2d.setColor(new Color(0, 0, 140));
  g2d.fill(new Ellipse2D.Double(280, 180, 240, 240));
}
```

Exemplo 1 – Primitivas Geométricas

Tipo `Image`

- Jogos não são criados somente com formas geométricas básicas. Normalmente a arte do jogo é definida por um **conjunto de imagens**.
- Java oferece uma classe para armazenar imagens chamada `Image`.

```
Image img;
```

- Podemos **carregar uma nova imagem** através do comando:

```
img = new ImageIcon(this.getClass().
                    getResource("image.png")).getImage();
```

- Podemos **desenhar uma imagem** através do comando:

```
g2d.drawImage(im, 200, 200, this);
```

Tipo Image

- **Exemplo:**

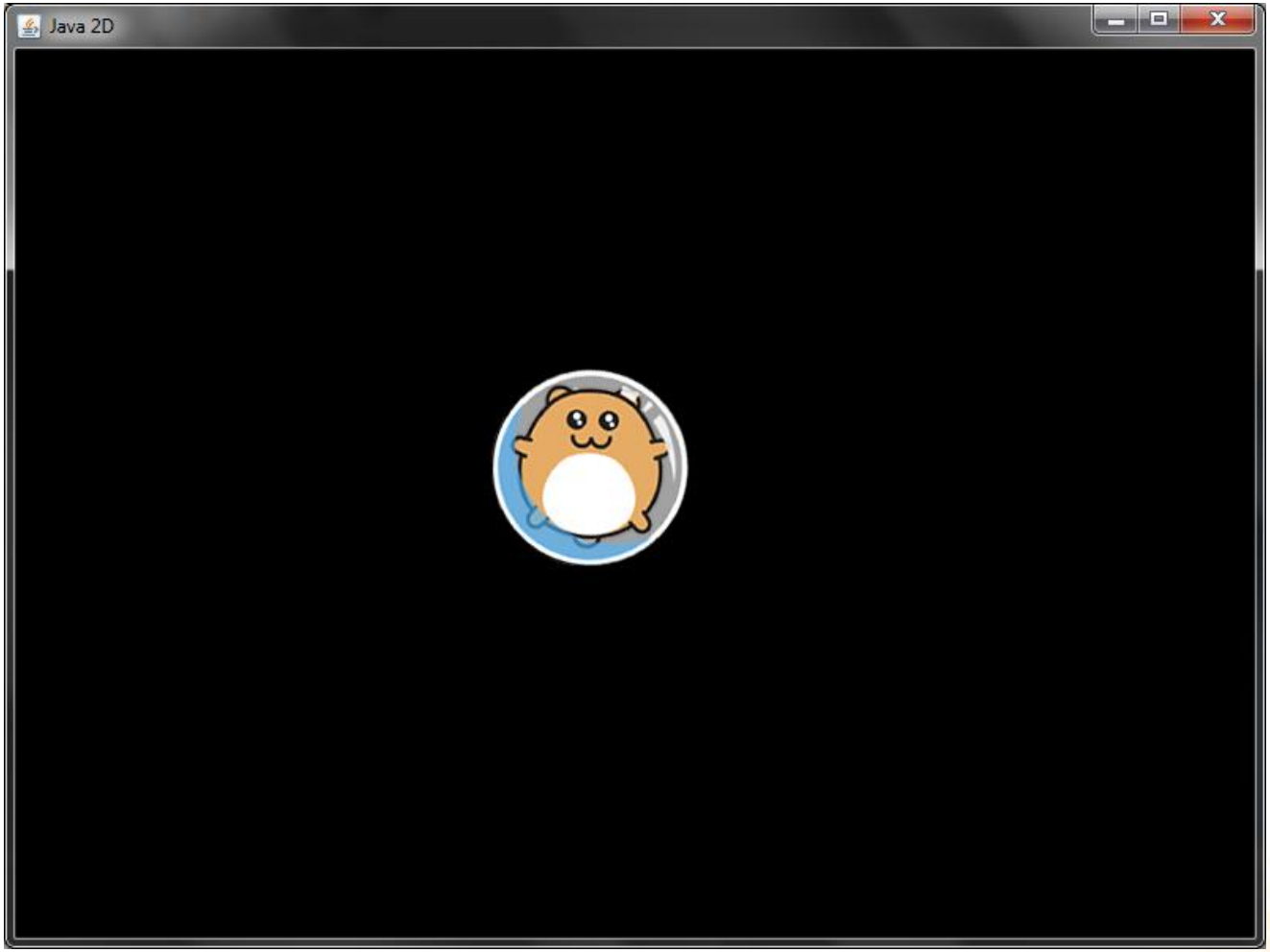
```
private Image hamster;
```

```
private void load()
{
  setBackground(Color.BLACK);
  hamster = new ImageIcon(this.getClass().
                  getResource("hamster.png")).getImage();
}

private void draw(Graphics g)
{
  Graphics2D g2d = (Graphics2D) g;
  g2d.drawImage(hamster, 200, 200, this);
}
```

http://www.inf.puc-rio.br/~elima/intro-eng/hamster.png

Tipo `Image`



Java 2D – Interação pelo Teclado

```
 private boolean[] key_states = new boolean[256];
```

```
private class KeyboardAdapter extends KeyAdapter
{
  @Override
  public void keyReleased(KeyEvent e)
  {
    key_states[e.getKeyCode()] = false;
  }

  @Override
  public void keyPressed(KeyEvent e)
  {
    key_states[e.getKeyCode()] = true;
  }
}
```

Java 2D – Interação pelo Teclado

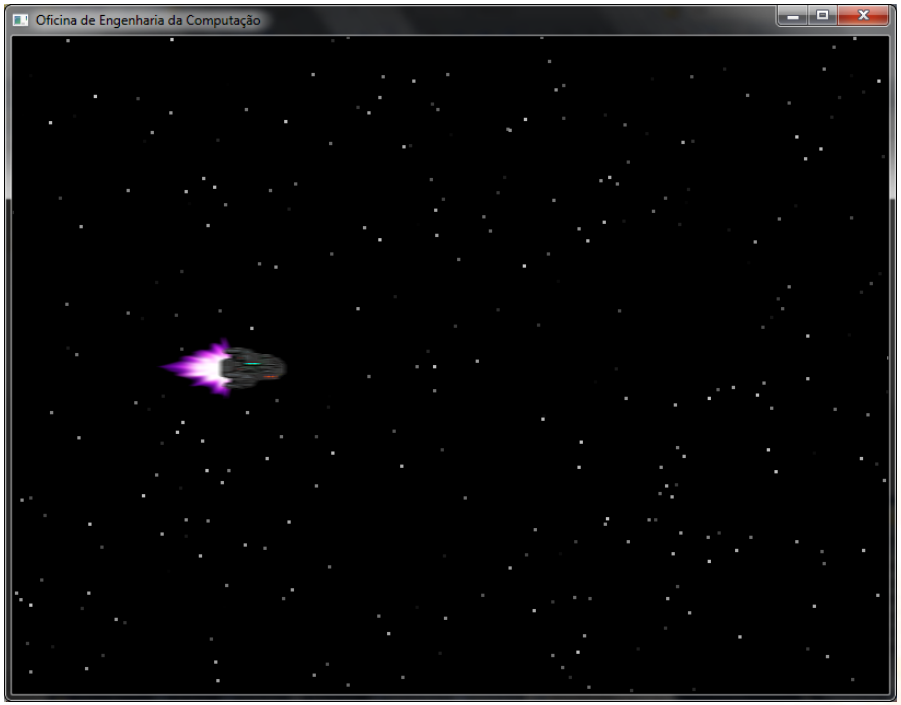
```
private void load()
{
  addKeyListener(new KeyboardAdapter());
}

private void update(double dt)
{
    if (key_states[KeyEvent.VK_RIGHT])
        px = px + (100 * dt);
    if (key_states[KeyEvent.VK_UP])
        py = py - (100 * dt);
    ...
}

private void draw(Graphics g)
{
    Graphics2D g2d = (Graphics2D) g;
    g2d.fill(new Ellipse2D.Double(px, py, 100, 100));
}
```

Projeto: Space Shooter

- **Jogo simples:** Space Shooter
  1. Controle da nave pelo teclado;
  2. Mantendo a nave na tela;
  3. Geração de estrelas;
  4. Turbo para a nave;



Projeto: Space Shooter

- **Continuação?** Trabalho 2!
  1. Implementar a estrutura do jogo usando classes;
  2. Atirar com a nave (usando array de tiros);
  3. Gerar inimigos (usando array de inimigos);
  4. Detectar colisão (nave/inimigos);
  5. Detectar colisão (tiros/inimigos);
  6. Controle de vidas e pontuação;
  7. Etc...

Exercícios

Lista de Exercícios 06 – Java 2D

http://uniriodb2.uniriotec.br