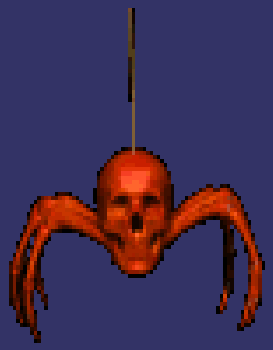
Desenvolvimento Básico de Jogos 2D utilizando a linguagem JAVA



Por

Alexandre Stürmer Wolf

Visão Geral

- Teoria;
  - Introdução;
  - Histórico;
  - Motivação;
  - Objetivos;
  - Requisitos
  - O que é um Jogo de Computador;
  - Ferramentas de desenvolvimento
  - Conclusão;

Visão Geral

- Prática (Desenvolvendo um jogo simples)
  - Requisitos exemplificados;
    - Utilizando a interface gráfica;
    - Captura de eventos;
    - Primitivas de Imagens;
    - Imagens prontas;
    - Aplicação de Som;
    - Programação Concorrente;
    - Problemas de Flic (cintilação);
    - Utilizando componentes SWING;

Histórico

- O primeiro jogo de computador que se tem notícia foi criado por A.S. Douglas em 1952, em seu doutorado na Universidade de Cambridge. Ele criou uma versão gráfica do jogo-da-velha, que rodava no computador de válvulas EDSAC.
- Os jogos de computador começaram com Spacewar, um jogo criado pelos estudantes do MIT em 1961, com um gigantesco computador PDP-1, utilizado para cálculos estatísticos.
- O primeiro jogo de aventura pra PC foi Colossal Cave Adventure, abreviado Adventure, em 1976. O jogo era todo baseado em texto. A maioria dos jogos permaneceu assim até o surgimento do mouse nos anos 80. Depois os jogos misturavam imagens estáticas e texto. A medida que os computadores pessoais ficavam mais poderosos, a jogabilidade e a capacidade de gerar objetos que se moviam aumentava.

Motivação

- Atualmente o desenvolvimento de jogos e entretenimento eletrônico representam um mercado de milhões de dolares;
- Nos últimos anos é que esse mercado começou a ser explorado aqui no Brasil;
- A proliferação de equipamentos de celular a baixo custo com possibilidade de instalar/desinstalar jogos com extrema facilidade;
- A distribuição através da internet;
- A possibilidade de se criar um bom produto “no fundo do quintal“ e mudar de vida de um dia para o outro...

Objetivos

- Apresentar os principais recursos da linguagem que podem ser utilizados em aplicações de multimídia;
- Noções e utilização de eventos de iteração;
- Programação concorrente usando Threads;
- Desenvolver uma aplicação de iteração com o usuário;
- Permitir ao programador criar um pequeno jogo de computador;
- Conhecer técnicas de movimento e lógica de interceptação;
- Demonstrar o funcionamento de teorias baseadas no “caos“ afim de tornar o jogo mais complexo e menos previsível;

Requisitos

- Um jogo de computador é composto de ENTRADA / PROCESSAMENTO e SAÍDA, assim como qualquer requisito em informática. Assim, devemos saber receber os dados via teclado, mouse e/ou joystick (entrada), atualizar os dados do jogo em função da entrada recebida (processamento) e por último mostrar as imagem e o som (saída).
- Contudo, estes requisitos devem ser vistos apenas como os conhecimentos básicos.
- Rotinas condicionais, laços de repetição, estruturas de dados são inevitáveis;
- Bom conhecimento matemático, auxilia muito...

- ACIMA DE TUDO: CRIATIVIDADE e uma ótima lógica...

O que é um Jogo de Computador

- Um jogo de computador, é um conjunto de várias artes diferentes formando uma só;
- Como a idéia de arte é altamente subjetiva, geralmente nunca irá agradar a todos.
- Deve poder ser manipulada pelo computador, onde através de um dispositivo de entrada, voce interage com algum “fenomeno“ que acontece no computador (processamento), obtendo alguma resposta em um dispositivo de saída;
- Um jogo de computador pode ser considerado um “Passa-Tempo“ que é jogado através de um computador.
- Outros consideram uma ferramenta para desenvolver capacidades de reflexo e atenção;

Ferramentas de Desenvolvimento

- Quando se desvenvolve aplicações de maoir vulto, sempre é interessante se utilizar de ferramentas que apoiam o desenvolvimento;
- Atualmente as ferramentas mais eficientes (Free) são:
  - Eclipse (www.eclipse.org);
  - NetBeans (www.java.sun.com);

  Outras ferramentas:
  - Dr. Java;
  - J++ (boa, mas não Free);

- Nota do Autor (Pessoal): com o atual lançamento da versão 5.0 do NetBeans, essa se tornou a melhor ferramenta do mercado.

Conclusão Inicial

- É necessário ter várias capacidades para se desenvolver um jogo de computador, acima de tudo CRIATIVIDADE;
- "SER UM ÓTIMO DESENVOLVEDOR DE PROGRAMAS COMERCIAS, NÃO QUER DIZER QUE CONSIGA DESENVOLVER JOGOS, MAS GARANTO QUE O INVERSO É VERDADEIRO".
- Desenvolver jogos é criar ARTE, sem inspiração, nada é possível;
- O ideal é desenvolver em equipe, onde cada um é responsável por uma área (cenário, personagens, engine, lógica, história);

Iniciando...

- Todos os recursos apresentados serão implementados;
- Após “n“ transparências, será exibido um código completo que poderá ser copiado e executado;
- Você pode escolher a ferramenta que bem desejar, para executar a aplicação;
- Por questões de pouco tempo e muito conteúdo, sugiro utilizar o Dr. Java;
- Quanto ao sistema operacional, fica a critério do usuário, no entanto os diretórios de recursos deverão ser adaptados de acordo com a necessidade;
- Caso esteja em dúvida ao escolher o S.O., sugiro o Linux, pois facilita o término de aplicações awt e Threads desgovernadas, sem maiores prejuízos, situação que não ocorre no Windows...
- Códigos disponíveis em http://ensino.univates.br/~awolf na pasta minicurso.

1 - Estrutura Gráfica Básica

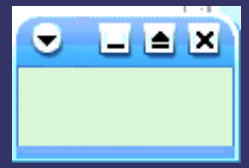
- Herdar as características da classe “Frame“(**awt**) ou “JFrame“ (swing).

ex1

```
import java.awt.*;
public class TelaJogo extends Frame{
        public TelaJogo(){
                super();
        }

        public static void main( String args[] ){
                TelaJogo jogo = new TelaJogo();
                jogo.setVisible( true );
        }
}
```

Código para a criação de uma janela de comunicação com o usuário

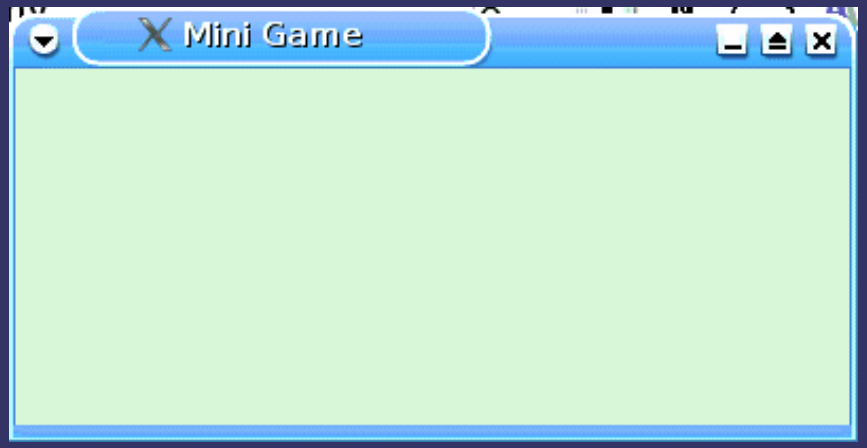


Definindo Características da Janela de Comunicação com o Usuário...

- Uma vez instanciado o “nosso“ construtor, o mesmo chama o contrutor da classe pai e cria uma janela;
- Entre os parâmetros que podem ser passados para o construtor da classe Frame, é o título da janela;
- Caso deseje, pode mudar o titulo da janela usando setTitle(<Titulo>);
- Outra característica importante é o dimensionamento da tela, para isso use setSize(<Largura>,<Altura>);

Ex:

```
public TelaJogo(){
        super("Mini Game");
        setSize( 400, 200 );
}
```



Definindo Características da Janela de Comunicação com o Usuário...

- Por “default“, a classe “awt“ não possui tratamento para o fechamento de janelas, já a swing possui;
- Uma solução para a “awt“ é criar um método para tratar esse evento .(Colocar a chamada/método dentro do construtor do jogo);

```
addWindowListener(new java.awt.event.WindowAdapter() {
        public void windowClosing(java.awt.event.WindowEvent evt) {
                exitForm(evt);
        }
});

private void exitForm(java.awt.event.WindowEvent evt) {
        System.exit(0);
}
```

Até o momento temos para a “awt“ ...

```
import java.awt.*;
public class TelaJogo1 extends Frame{
  public TelaJogo1() {
    super("Mini Game");
    setSize(400, 200);

    addWindowListener(new java.awt.event.WindowAdapter() {
      public void windowClosing(java.awt.event.WindowEvent evt) {
        exitForm(evt);
      }
    });
  }

  private void exitForm(java.awt.event.WindowEvent evt) {
    System.exit(0);
  }

  public static void main(String args[]){
    TelaJogo1 jogo = new TelaJogo1();
    jogo.setVisible(true);
  }
}
```


Até o momento temos para a “swing“ ...

```
import java.awt.*;
import javax.swing.*;
public class TelaJogo2 extends JFrame{
   public TelaJogo2() {
      super("Mini Game");
      setSize(400, 200);
   }

   public static void main(String args[]){
      TelaJogo2 jogo = new TelaJogo2();
      jogo.setVisible(true);
   }
}
```

Observe o “J“ antes do Frame

O tratador de eventos para fechamento de janelas é implementado na própria JFrame

Criando um Tratador de Eventos do Teclado

- Indiferente do uso da “awt“ ou “swing“, os tratadores de eventos devem ser obrigatoriamente desenvolvidos, seja manualmente ou com uma ferramenta IDE(Integrated development environment).

Ex:

```
addKeyListener(new java.awt.event.KeyAdapter() {
        public void keyPressed(java.awt.event.KeyEvent evt) {
                TrataTeclas(evt);
        }
});

private void TrataTeclas(java.awt.event.KeyEvent evt) {

}
```

Até o momento temos ...

```
public TelaJogo() {
   super("Mini Game");
   setSize(400, 200);

   addKeyListener(new java.awt.event.KeyAdapter() {
      public void keyPressed(java.awt.event.KeyEvent evt) {
         TrataTeclas(evt);
      }
   });

}

private void TrataTeclas(java.awt.event.KeyEvent evt) {

}
```

Adiciona um novo evento à lista de “escutadores“ dos eventos de teclado

Acrescentado ao nosso construtor uma chamada a um método tratador

Método para tratar os eventos do teclado

Utilizando os Eventos do Teclado

- Podemos obter o código das teclas do teclado;
- Imprimir os códigos para posteriormente utilizar-los em uma aplicação;
- Enquanto a aplicação estiver aberta, a mesma vai imprimir os códigos correspondentes de cada tecla pressionada (int) símbolo (char), localização (teclado numérico/alfanumérico) (int);

Ex:

```
private void TrataTeclas(java.awt.event.KeyEvent evt) {
        System.out.println(evt.getKeyCode());
        System.out.println(evt.getKeyChar());
        System.out.println(evt.getKeyLocation());
}
```

Código Completo até Agora...

```
import java.awt.*;
import javax.swing.*;
public class TelaJogo3 extends JFrame{
   public TelaJogo3() {
      super("Mini Game");
      setSize(400, 200);
      addKeyListener(new java.awt.event.KeyAdapter() {
            public void keyPressed(java.awt.event.KeyEvent evt) {
              TrataTeclas(evt);
         }
      });
   }

   private void TrataTeclas(java.awt.event.KeyEvent evt) {
      System.out.println(evt.getKeyCode());
      System.out.println(evt.getKeyChar());
      System.out.println(evt.getKeyLocation());
   }

   public static void main(String args[]) {
      TelaJogo3 jogo = new TelaJogo3();
      jogo.setVisible(true);
   }
}
```

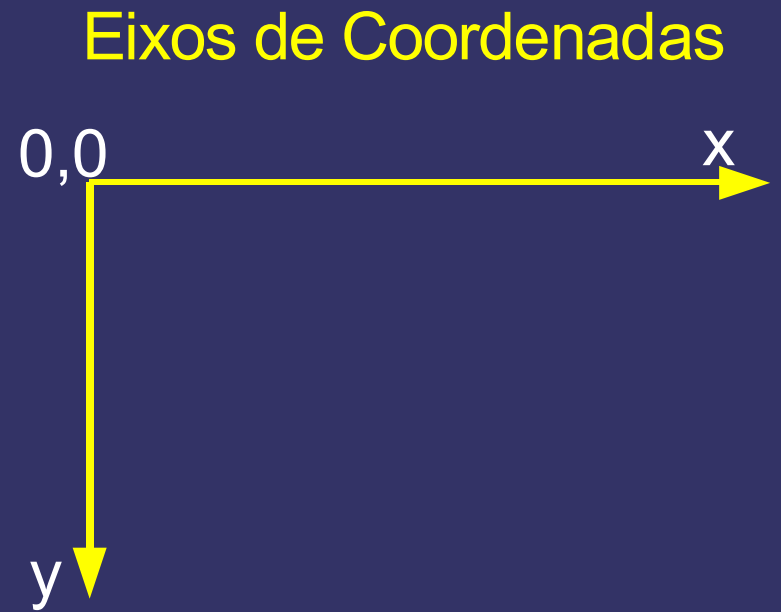

2 – Desenhando na Janela (Primitivas)

- A interface gráfica do java dispara automaticamente um método chamado “paint“ toda vez que precisa redesenhar a tela;
- Podemos usar o método “paint“ para “redesenhar“ na tela o que desejamos;

Ex:

```
public void paint(Graphics g)
{

}
```



Principais Métodos da Graphics

- Ver a API do Java para todos métodos nativos;

```
public void paint(Graphics g)
{
    g.setColor(Color.blue);
    g.setFont(new Font("Serif", Font.BOLD, 12));
    g.drawString("Fonte Serif aqui", 20, 70);
    g.drawLine(10, 10, 100, 100);


    g.setColor(Color.red);
    g.setFont(new Font("Times New Roman",
    Font.BOLD+Font.ITALIC, 14));
    g.drawString("Testando a fonte times new roman", 40, 90);
}
```

Mais exemplos...

```
public void paint(Graphics g)
{
      g.setColor(Color.blue);
      g.setFont(new Font("Ariel",Font.PLAIN, 14));
      g.drawString("Testando a fonte Ariel", 10, 120);

      g.drawRect(20, 20, 80, 80);
      g.drawOval(100, 80, 150, 180);

      g.fillOval(150, 100, 180, 150);
      g.drawRoundRect(200, 200, 250, 250, 50, 50);

      g.setColor(Color.green);
      g.drawArc( 300, 100, 40, 50, 10, 80);
}
```


Mais exemplos...

```
public void paint(Graphics g)
{
	g.setColor(Color.blue);
	g.setFont(new Font("Serif", Font.BOLD, 12));

	for(int n=0; n<40; n++)
	{
		g.drawString("Olha a impressão" + n, 20 + n * 10, 70 + n * 10);
	}

}
```


Usando Imagens (Toolkit)

```
Toolkit toolkit = Toolkit.getDefaultToolkit();
Image image = toolkit.getImage("img.gif");


public void paint( Graphics g)
{
    g.drawImage(image, 100, 100, null);
}
```

Esse tipo de procedimento causa muitos problemas de refresh

Exemplo

```
import java .applet.*;
import java.awt.*;
import javax.swing.*;
public class DesenhaTela extends JFrame{
  private Image imagem;

  public DesenhaTela() {
    super("Mini Game");
    setSize(400, 400);
    JPanel pa = new JPanel(true);
    getContentPane().add(pa);

    Toolkit toolkit = Toolkit.getDefaultToolkit();
    imagem = toolkit.getImage("/root/progs/java/Games/JogoDoido/gato.gif");
  }

  public void paint(Graphics g) {
    g.drawImage(imagem, 10, 10, null);
  }

  public static void main(String args[]) {
    DesenhaTela tela = new DesenhaTela();
    tela.setVisible(true);
  }
}
```

Aplicando Som

```
import java.applet.*
import java.net.*;
public class Som
{
   public static void main(String args[])
   {
      AudioClip Tiro = null;

      try{

         URL u;
         u = new URL("file:///root/progs/java/Jogos/Recursos/tiro.wav");
         Tiro = Applet.newAudioClip(u);


      }catch(Exception e){

         System.out.println("Erro ao carregar o som");

      }

      Tiro.stop();
      Tiro.play();
   }
}
```

Concorrência

```
public class Concorrencia1
{
   public static void main(String args[])
   {
      for(int i=0; i<10; i++)
      {
         new Concorrencia2(i).start();
      }
   }
}
```

```
public class Concorrencia2 extends Thread{
   private int Processo;
   private int Pausa;

   public Concorrencia2(int Processo) {
      this.Processo = Processo;
      Pausa = (int)(Math.random()*500)+500;
   }

   public void run(){
      for(int i=0; i<10; i++)
      {
         System.out.println(" Concorrencia " + Processo);
         try{sleep(Pausa); }catch(Exception e){}
      }
   }
}
```

Problemas de Flic

- A cintilação é um problema grave que ocorre durante a animação;
- Para eliminar a cintilação, devemos utilizar a técnica Double Buffer, ou seja, desenhar a tela toda em memória e depois imprimir no vídeo;
- Existem várias receitas de “bolo“ na internet, cada uma com vantagens e desvantagens;
- O Double Buffer ideal deve atualizar na tela somente o que foi modificado no Buffer, no entanto esse procedimento de analise, muitas vezes, pode ser muito complexo, tornando o procedimento dispendioso e mais lento do que imprimir todo o Buffer.

Como desenvolver então?

- A classe Swing, apesar de ser mais lenta do que a awt, possui para quase todos os componentes a possibilidade de utilizar Double Buffer, isso consome mais memoria e torna o processo um pouco mais lento, mas torna a animação mais suave;
- Os programadores profissionais de jogos não utilizam o Double Buffer da Swing por ser o “Bubble Sort“ da programação, ou seja, é eficiente mas não eficaz (atende mas não é a melhor solução);

Utilizando os Componentes Swing

- Muitas pessoas solicitam um componente específico para colocar imagens na tela usando componentes Swing;
- Isso não é necessário, pois quase todos componentes permitem colocar imagens em seu interior;
- A grande maioria dos programadores utilizam o componente Label (etiqueta/texto), mais precisamento JLabel para colocar imagens em seu interior;
- Utiliza-se da capacidade de “Iconificação“, ou seja, imprimir um ícone no componente
- Compatível com:
  - Gif (gif animado);
  - JPG (jpeg);

Utilizando...

```
import javax.swing.*;
public class Principal extends JFrame {
   private JLabel Explosao;

   private Principal() {
      Explosao = new JLabel();
      Explosao.setIcon(new ImageIcon("/root/progs/java/Jogos/Recursos/explosao.gif"));

      getContentPane().add(Explosao);
      pack();
   }

   public static void main(String args[]) {

      new Principal().setVisible(true);

   }
}
```


Juntando Todos Recursos

- Para poder criar um jogo, devemos juntar todos os recursos anteriormente apresentados;
- Não adianta conhecer os recursos se não se sabe o que se quer ou como utilizar os mesmos;

- Idéia proposta:
  - Fugir dos monstros, se te pegarem você morre;
  - Você pode atirar???

- Apresentação de um protótipo jogável...

- Em anexo o parcial