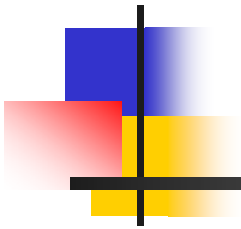
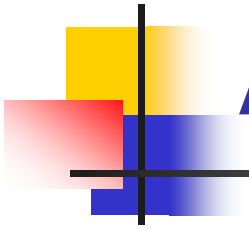


Programação Gráfica – Parte 3

Versão em Java – 2006 – PUCPR – Tutoria de Jogos – 1º Ano

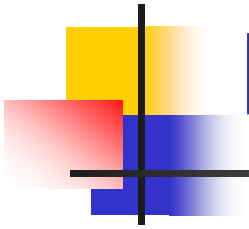
Paulo V. W. Radtke

pvwradtke@gmail.com
http://www.ppgia.pucpr.br/~radtke/jogos/



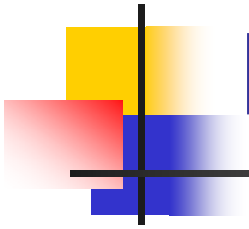
AVISO IMPORTANTE!!

- Esta versão é dedicada exclusivamente para o cursos de **Sistemas de Informação**.
- Para a versão de **Ciência da Computação** e **Engenharia da Computação**, utilizando **C**, pegue o arquivo correspondente e participe da aula no horário adequado.



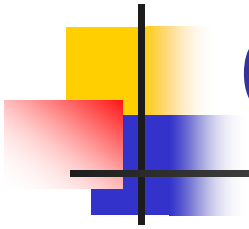
Entrega da 2ª Parcial

- Datas importantes:
  - 12 a 17 de Junho.
  - Relatório impresso contendo:
    - Código fonte do protótipo da interface.
    - Impressão no relatório dos recursos gráficos da fase/jogo (*tilemaps*, sprites, cenários de fundos, protótipo, etc).
    - Discussão do uso dos recursos com a lógica do jogo no terceiro bimestre.
  - Defesa em laboratório do protótipo e entrega do relatório com a equipe completa.



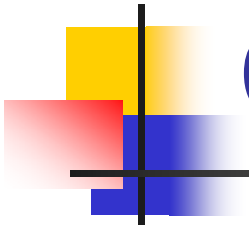
Entrega da 2ª Parcial

- Por recursos gráficos,entende-se que neste bimestre já teremos:
  1. TODOS os sprites necessários para o demo.
  2. TODAS as telas de fundo/tilemaps.
  3. TODAS as fontes.
- Logo, espera-se que o relatório inclua TODOS estes elementos.



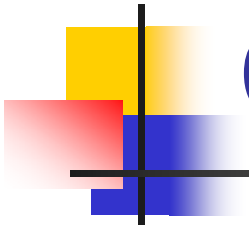
Conteúdo

- Carregando Imagens
- Desenhando Imagens
- Criando Sprites
- Desenhando um Sprite
- Animando o Sprite



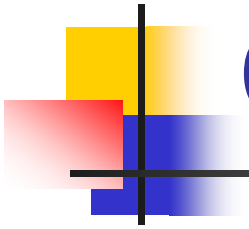
Carregando Imagens

- Imagens para Midlets J2ME devem estar no formato PNG.
- Este formato é escolhido por permitir transparência, ser aberto e por possuir boa compressão.
- Em geral, para reduzir o tamanho das imagens, utilizamos imagens com 256 cores.



Carregando Imagens

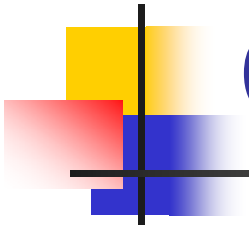
- Imagens são SEMPRE carregadas a partir do diretório raiz do Midlet.
- Durante a fase de desenvolvimento, as imagens ficam no diretório **res** da aplicação.
- Assim, **TODAS** imagens (sem exceção) utilizas pela aplicação devem ficar neste diretório.



Carregando Imagens

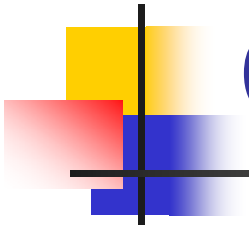
- Imagens são representadas em J2ME pela classe **Image**.
- Esta classe também nos fornece 6 (seis) métodos estáticos diferentes para criar imagens.
- Porém, nos interessa UM deles:

public static Image createImage(String name)
throws IOException



Carregando Imagens

- Este método carrega uma imagem a partir do arquivo JAR (ou diretório **res** do WTK).
- Para tal, basta fornecer o nome do arquivo (ex: “/nome.png”).
- Este método pode falhar e joga uma exceção de I/O caso isso ocorra.



Carregando Imagens

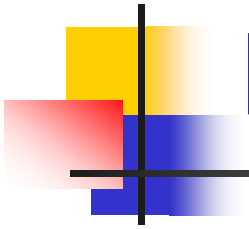
- A referência para **Image** normalmente é criada como um atributo do GameCanvas.
- Caso a imagem vá ser usada durante toda a aplicação, esta é carregada no construtor.

Carregando Imagens

- Exemplo de uso típico:

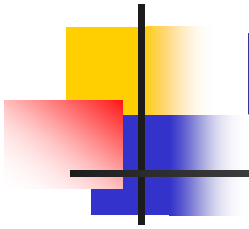
Deve ser um atributo da classe declarado previamente!

```
try
{
   imagem=Image.createImage("/fundo.png");
}
catch(java.io.IOException e){}
```



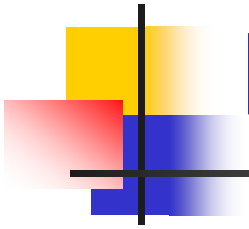
Desenhando Imagens

- Para desenhar a imagem, utilizamos o contexto gráfico (Graphics) e o método **drawImage**.
- O método desenha a imagem baseada numa coordenada indicada e um dado alinhamento.
- Os exemplos até agora desenhavam um texto. O desenho de uma imagem é bastante semelhante.



Desenhando Imagens

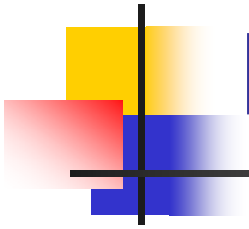
- public void drawImage(Image img, int x, int y, int anchor) :
  - *img*: referência para a imagem desenhada.
  - *x* e *y*: coordenada de referência.
  - *anchor*: indica a posição relativa da referência:
    - LEFT: à esquerda
    - RIGHT: à direita
    - TOP: em cima
    - BOTTOM: em baixo
    - HCENTER: centro na horizontal.
    - VCENTER: centro na vertical.



Desenhando Imagens

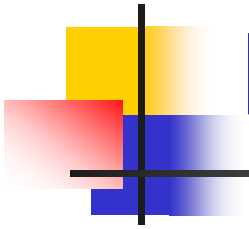
- Exemplo de uso:

```
Graphics g = getGraphics();
...
g.drawImage(imagem, 0, 0,
   Graphics.LEFT | Graphics.TOP);
```



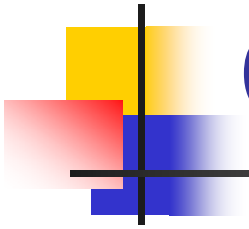
Exercício 01

- Crie um projeto com o Midlet chamado TestaSprite no KToolBar.
- Pegue o conteúdo do arquivo j2me-**exemplo04-TestaSprite.zip** e coloque os arquivos nos diretórios adequados.
- Modifique o código para carregar a imagem no arquivo **fundo.png** e mostre-a como fundo da tela.
- Rode e veja o resultado.



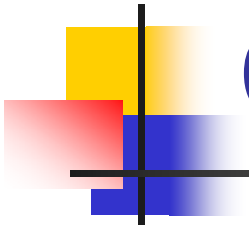
Desafio

- A imagem foi desenhada completa.
- Temos como desenhar apenas um pedaço dela? (*clipping*)
- Podemos determinar apenas uma área na tela para desenhar?
- Veja na documentação do J2ME na classe Image e Graphics.



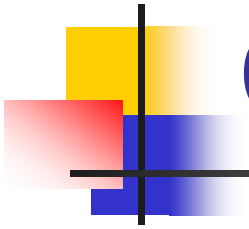
Criando Sprites

- Sprites em J2ME são criados a partir de imagens.
- Lembrando, *sprites* são os elementos animados que representam os personagens do jogo.
- A classe do sprite chama-se **Sprite**.



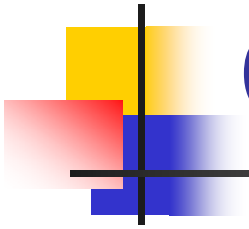
Criando Sprites

- Sprites podem ter pontos transparentes ou não.
- Para colocar pontos transparentes, temos que usar uma ferramenta de edição de imagens.
- Para imagens de 256 cores ou menos, a cor transparente é uma cor chave.
- Para imagens com mais cores usa-se o canal de alpha.



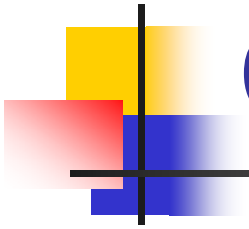
Criando Sprites

- Recomendações:
  - Para evitar problemas, use cores contrastantes.
  - Em geral, como serão usadas poucas cores, podemos utilizar imagens com 256 cores.
  - Outro motivo para usar arquivos com menos cores é o tamanho do arquivo.
  - Finalmente, certos *handsets* que não aceitam transparência por canal de alpha, só por cor chave (*color key*), que só é obtido com 256 cores.



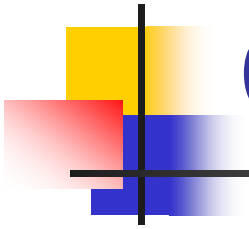
Criando Sprites

- Para criar um sprite a partir de uma imagem seguimos os seguintes passos:
  1. Carregamos a imagem do JAR, como no exemplo da tela de fundo.
  2. Em seguida, criamos o sprite usando como base a imagem recém carregada.



Criando Sprites

- Para criar o sprite, utilizamos basicamente dois métodos construtores:
  - public Sprite(Image image)
  - public Sprite(Image image, int frameWidth, int frameHeight)
- Como em geral criamos sprites animados, interessa o segundo método.

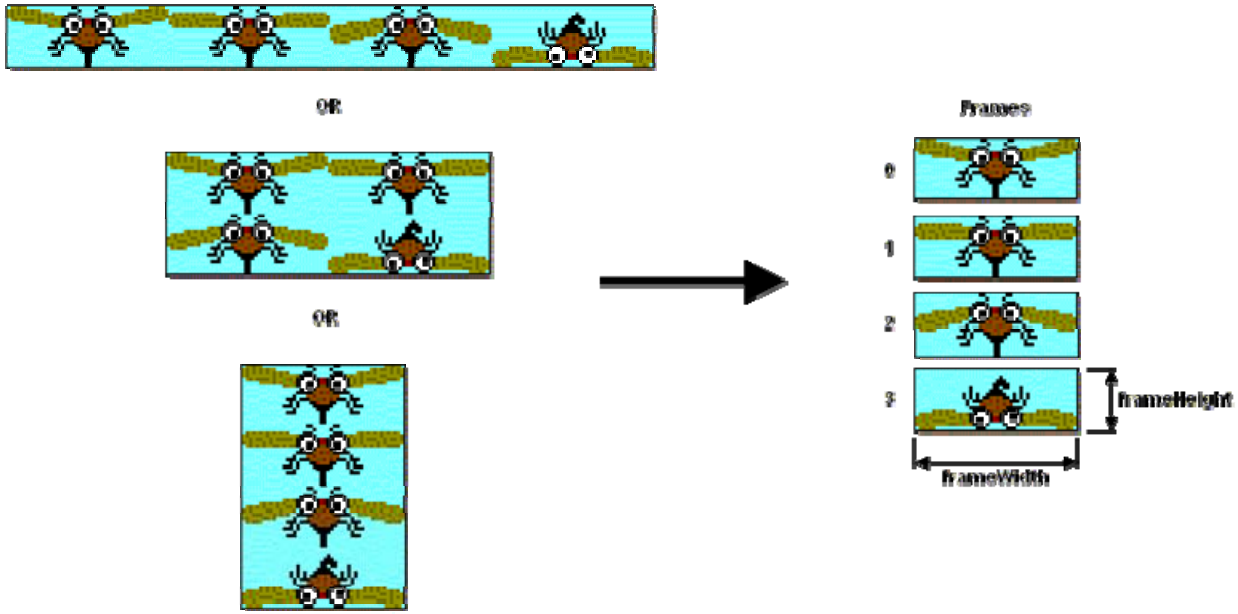


Criando Sprites

- Neste método, os parâmetros do frame significam:
  - frameWidth: largura do quadro de animação.
  - frameHeight: altura do quadro de animação.

Criando Sprites

- Exemplos de arquivos com mais de um quadro de sprite:

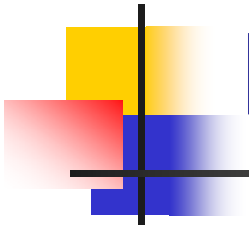


Criando Sprites

- Exemplo de criação de sprite no construtor do GameGavas:

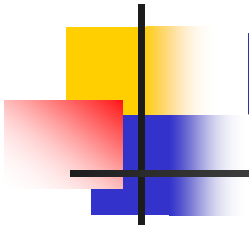
```
// Carrega o sprite
try
{
    Image temp = Image.createImage("/sprites16.png");
    spr = new Sprite(image, 16, 16);
}
catch(java.io.IOException e){}
```

Deve ser um atributo da classe declarado previamente!



Desenhando um Sprite

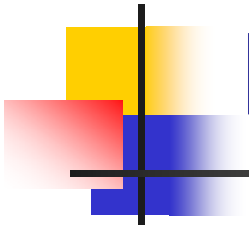
- Ao contrário de imagens, o sprite possui suas próprias funcionalidade de desenho.
- Esta funcionalidade recebe o contexto gráfico para desenhar.
- Como na função de redesenho nós capturamos o contexto, basta passá-lo ao sprite.



Desenhando um Sprite

- O método que utilizamos para desenhar um sprite é paint, que possui a seguinte assinatura:

public final void paint(Graphics g)

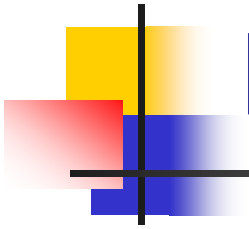


Desenhando um Sprite

- Para indicar em qual posição o sprite é desenhado utilizamos o seguinte método:

public void setPosition(int x, int y)

- Este método é herdado de **Layer**.

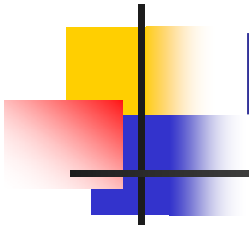


Desenhando um Sprite

- Assim, um código que desenha um *sprite* tem a seguinte aparência:

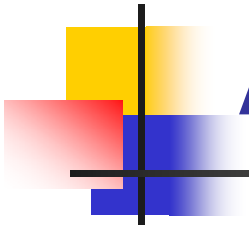
spr.setPosition(200,50);

spr.paint(g);



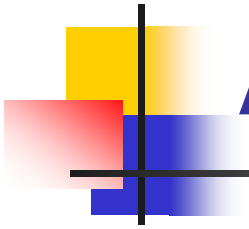
Exercício 02

- Modifique o exercício anterior para carregar o sprite no arquivo sprites16.png.
- Este arquivo contém um sprite com dois quadros de tamanho 16x16.
- Desenhe este sprite na tela.
- Utilize o estado das teclas para mover o sprite na tela.



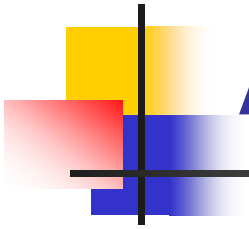
Animando o Sprite

- Sprites estáticos são interessante, porém em muitas situações precisamos de animações.
- Tais animações são obtidas com diversos quadros desenhados.
- O arquivo **sprites16.png** possui dois quadros.



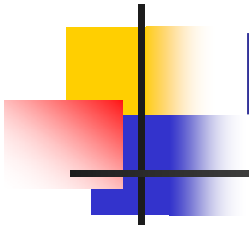
Animando o Sprite

- O programa anterior mostrava apenas um deles.
- Por default, o J2ME mostra o quadro 0.
- Para mudarmos o quadro selecionado, utilizamos o seguinte método da classe Sprite:
- *public void setFrame(int indice)*



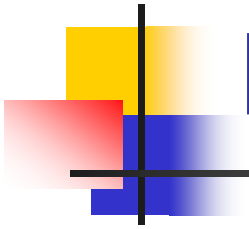
Animando o Sprite

- Este método recebe um número VÁLIDO da sequência.
- Assim, ele escolhe qual dos quadros será desenhado nos paints subsequentes.
- Se trocarmos os quadros de tempos em tempos, podemos ter a ilusão de animação.



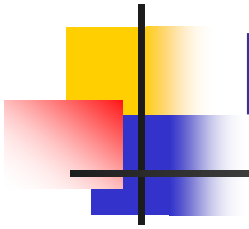
Exercício 03

- Modifique o exercício anterior para mudar o quadro exibido do sprite com o pressionamento do botão de tiro.



Exercício 04

- Além de mudarmos manualmente o quadro, podemos definir uma seqüência complexa com o método **setFrameSequence**, que recebe um vetor de inteiros.
- Depois, utilizamos os métodos **nextFrame** e **previousFrame** para rodar a animação.
- Estude a documentação do J2ME e experimente fazer uma animação com mais quadros.



Próxima Aula

- Desenhando um TileMap
- Colisão de Sprites