7 – Animação

Animações é um dos quesitos muito importantes em jogos, você não acha?
Para isso o JPlay tem uma classe específica para lidar com animações.
Essa classe se chama Animation. Bem sugestivo o nome não?

7.1 – Como criar uma animação?

Para criar uma animação precisamos de uma imagem e que ela contenha alguns frames. O número de frames é uma escolha sua.

Um frame é um pedaço da imagem responsável por um movimento da animação.

Exemplo: temos a imagem abaixo:

Repare que existem quatro desenhos do Megaman em uma única imagem. Sendo que cada um desses desenhos pode ser chamado de frame. Logo, essa imagem possui 4 frames.

O conceito de frame é muito importante em animações, tenha-o sempre em mente.

7.2 – Instanciando a classe Animation

A classe a Animation tem dois construtores. Por enquanto, o único que nos interessa é o seguinte:

Animation(nome da imagem, número de frames).

Para criar um objeto dessa classe procedemos do seguinte modo:

```
Animation animacao = new Animation("animacao01.png", 4);
```

nome da imagem = animacao01.png
número de frames = 4

7.3 – Setando o tempo entre a mudança de frames

Em animações os frames devem mudar depois de um certo tempo. Para informar o tempo de mudança entre os frames ou o tempo em que cada frame será apresentado na tela, usamos o método **void setTimeChangeFrame(long time)**.

```
animação.setTimeChangeFrame(125);
```

Seta o tempo de mudança entre cada um dos 4 frames, isto é, a cada 125 milissegundos o frame apresentado na tela irá mudar, ou falando de outro modo, cada frame será apresentando na tela por 125 milissegundos.

O tempo a ser setado, pelo método apresentado, deve estar em milissegundos.
Lembre-se que 1 segundo é igual a 1000 milissegundos, 1s = 1ms.

7.4 – Executando a animação

Para fazer a animação ser executada o método **void runAnimation()** deve ser chamado, ele é o responsável pela troca de frames, respeitando o tempo estipulado pelo método void setTimeChangeFrame(long);

Até agora temos:

```
Animation animacao = new Animation("animacao01.png", 4);
animação.setTimeChangeFrame(125);
animacao.runAnimation();
```

Estamos prontos para criar a nossa primeira animação.

Exemplo 06: Roda uma animação

```
public class Exemplo06
{
   public static void main(String[] args)
   {
      Window janela = new Window(800,600);
      Keyboard keyboard = janela.getKeyboard();
      Mouse mouse = janela.getMouse();

      GameImage backGround = new GameImage("fundo.png");
      Animation animacao = new Animation("animacao01.png", 4);

      animacao.setPosition(300, 300);
      animacao.setTimeChangeFrame(125);

      boolean executando = true;
      while(executando)
      {
            backGround.draw();
            animacao.draw();
            janela.display();

            animacao.runAnimation();

            if ( keyboard.keyDown(Keyboard.ESCAPE_KEY) == true)
               executando = false;
      }
      janela.exit();
   }
}
```

7.5 – Executando a animação somente uma vez

Para executar a animação somente uma vez, use o método **void setRepeatAnimation(*boolean*)** da classe Animation. O valor passado por parâmetro deve ser *false*.

```
animação.setRepeatAnimation(false);
```

Quando uma animação é criada, ela será executada indefinidamente, para que isso não ocorra deve-se o usar o método mostrado acima passando o parâmetro *false*.

Se durante o jogo houver a necessidade de que a animação volte a ser executada use o método void setRepeatAnimation(*boolean*), passando como parâmetro o valor *true*. Isso fará com que a animação volte a ser executada.

```
animação.setRepeatAnimation(true);
```

Exemplo 07: Pausa uma animação

```
public class Exemplo07
{
   public static void main(String[] args)
   {
      Window janela = new Window(800,600);
      Keyboard keyboard = janela.getKeyboard();

      GameImage backGround = new GameImage("fundo.png");
      Animation animacao = new Animation("animacao01.png", 4);

      animacao.setPosition(300, 300);
      animacao.setTimeChangeFrame(125);
      animacao.setRepeatAnimation(true);

      long time = 0;

      boolean executando = true;
      while(executando)
      {
            backGround.draw();
            animacao.draw();
            janela.display();

            animacao.runAnimation();

            time += janela.timeElapsed();

            if (time > 4000 && time < 5000)
               animacao.setRepeatAnimation(false);
            else
                  if (time > 10000)
                     animacao.setRepeatAnimation(true);

            if ( keyboard.keyDown(Keyboard.ESCAPE_KEY) == true)
               executando = false;
      }
      janela.exit();
   }
}
```

A variável time serve para armazenar a quantidade de tempo.

O comando **long janela.timeElapsed()** retorna a quantidade de tempo em milissegundos passados deste a última atualização da tela e o momento de chamada do método.

Assim o comando time += janela.timeElapsed() é usado para contar o tempo.

Se o tempo passado for maior do que 4000 milissegundos (4 segundos) e menor do que 5000 milissegundos (5 segundos), faz a animação parar de rodar.

```
if (time > 4000 && time < 5000)
   animacao.setRepeatAnimation(false);
```

Se o tempo passado for maior do que 10 segundos volta a rodar a animação.

```
if (time > 10000)
   animacao.setRepeatAnimation(true);
```

7.6 – Trocando os frames manualmente

Exemplo 08: Trocando os frames manualmente

```
public class Exemplo08
{
  public static void main(String[] args)
  {
    Window janela = new Window(800,600);
    Keyboard keyboard = janela.getKeyboard();

    GameImage backGround = new GameImage("fundo.png");
    Animation animacao = new Animation("animacao02.png", 28);

    animacao.setPosition(300, 300);
    animacao.setTimeChangeFrame(80);
    animacao.setRangeOfFrames(0, 0);

    boolean executando = true;
    while(executando)
    {
        backGround.draw();
        animacao.draw();
        janela.display();

        animacao.runAnimation();

        if(keyboard.keyDown(Keyboard.LEFT_KEY))
          animacao.setRangeOfFrames(0, 13);
        else
          if(keyboard.keyDown(Keyboard.RIGHT_KEY))
            animacao.setRangeOfFrames(14, 27);

        if ( keyboard.keyDown(Keyboard.ESCAPE_KEY) == true)
          executando = false;
    }
    janela.exit();
  }
}
```

Para setar quais são os frames a serem usados na animação uso o método
void setRangeOfFrames(int frameInicial, int frameFinal).
Como se está setando os frames manualmente deve-se tomar o cuidado de fazer o seguinte:
animacao.setRangeOfFrames(0, 0),
isso garante que não haverá a troca de frames antes que elas realmente tenham que acontecer. Para saber o que aconteceria se isso não fosse feito experimente apagar esse comando.

No trecho de código abaixo temos que se a seta para a esquerda for apertada trocamos os frames que devem ser utilizados.

```
if(keyboard.keyDown(Keyboard.LEFT_KEY))
    animacao.setRangeOfFrames(0, 13);
```

Para entender melhor o que foi dito, pense no seguinte, quando apertar a seta para a esquerda os frames a serem utilizados na animação serão os 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12 e 13.

Já para o trecho de código abaixo:

```
if(keyboard.keyDown(Keyboard.RIGHT_KEY))
      animacao.setRangeOfFrames(14, 27);
```

os frames a serem utilizados na animação serão os 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 25, 25, 26, 27.

7.7 – Setando um tempo para cada frame

Em certas animações certa parte da animação deve ser executada mais rápida ou mais lenta do que outra parte. Para isso existe o método

```
public void setTimeOfFrame(int frame, long time);
```

Os parâmetros são: o número do frame e o valor do tempo em que o frame deve ser mostrado.

Exemplo:

```
animação. setTimeOfFrame(0, 100); // frame 0 será mostrado por 100 milissegundos
animação. setTimeOfFrame(1, 150); // frame 1 será mostrado por 150 milissegundos
animação. setTimeOfFrame(2, 120); // frame 2 será mostrado por 120 milissegundos
animação. setTimeOfFrame(3, 50);  // frame 3 será mostrado por 50 milissegundos
animação. setTimeOfFrame(4, 50);  // frame 4 será mostrado por 50 milissegundos
animação. setTimeOfFrame(5, 50);  // frame 5 será mostrado por 50 milissegundos
animação. setTimeOfFrame(6, 50);  // frame 6 será mostrado por 50 milissegundos
```

7.8 – Escondendo a animação

Para tornar a animação invisível use o método **public void hide().**

Para torna a animação novamente visível use o método **public void unhide();**

Obs.: Mesmo que animação não seja mostrada na tela, se o método runAnimation() estiver sendo chamado os frames continuarão a ser trocados.

7.9 – Últimas considerações

Se o que foi apresentando não suprir as necessidades, pode-se utilizar o método

public void setCurrFrame(int frame) -) este método seta o frame que deve ser desenhado.
Ao usar este método não é aconselhável que se use o método runAnimation() , pois isso poderia gerar alguns efeitos indesejáveis na troca de frames.
public boolean isAnimationFinished() – retorna *true* se houve a troca de todos os frames utilizados na animação.

public int getCurrFrame() – retorna o número do frame corrente a ser desenhado.

public boolean getRepeatAnimation() – retorna *true* se a animação será repetida, caso contrário, *false*.

public long getTimeOfFrame(int frame) – retorna o tempo em milissegundos que o frame será mostrado na tela.

public void reset() - reseta animação ao seu estado inicial, ou seja, seja o primeiro frame como o frame a será apresentando na tela.

public void setInitialFrame(int frame) – seta o frame que será usado para começar a animação.

public int getInitalFrame() – retorna o número do frame usado para começar a animação.

public void setFinalFrame(int frame) – seta o número do frame que será apresentado por último.

public int getFinalFrame() – retorna o número do frame final que é usado na animação.

public boolean isAnimationFinished() – retorna *true* se o último frame já foi apresentado, caso contrário, retorna false.

public long getTimeChangeFrame() – se não houve o uso do método setTimeOfFrame(int, long), todos os frames terão o mesmo tempo de apresentação na tela, o método getTimeChangeFrame retorna esse tempo.

public boolean getRepeatAnimation() – retorna true se a animação irá se repetir indefinidamente.

public long getTimeOfFrame(int frame) – retorna o tempo que o frame será mostrado na tela.

A classe Animation não possui métodos que façam a animação se mover pela tela. Para isso você pode usar as variáveis públicas ‘x’ e ‘y’.