Entendendo o básicão

O que é Sprite?

É um objeto gráfico bi ou tridimensional que se move na tela sem deixar traços de sua passagem (como se fosse um "espírito"). No nosso caso até o fundo da tela será um sprite, só que esse não se moverá.

Que treco era aquele de classe de controle?

Ah... essa é simples. O framework javaPlay usa números inteiros para chamar alguma classe que é responsável por alguma coisa. Seria como o número da sua matrícula na UFF, se o professor quiser se referir a você pode simplesmente chamar o seu número e aí você responde.

Ao ser chamada a classe passa a ter o controle total nesse instante.

Depois de ler e não entender nada, vamos a um exemplo.

Classe	Explicação	Número de Chamada
TelaInicial	Apresentar um mensagem no COMEÇO do jogo.	0
Jogar	Classe responsável por COMANDAR o jogo.	1
TelaFinal	Apresentar um mensagem no FIM do jogo.	2

Esse comando adiciona uma nova classe ao jogo com o número X sendo o de chamada.
GameEngine.getInstance().addGameStateController(X, new Classe());

```
GameEngine.getInstance().addGameStateController( 0, new TelaInicial() );
GameEngine.getInstance().addGameStateController( 1, new Jogar ()       );
GameEngine.getInstance().addGameStateController( 2, new TelaFinal ()  );
```

Tá tá... Já adicionei as classes...

Calma pequeno mestre. Já adicionou as classes para controle e agora quer saber como dizer qual é a classe tem que ser chamada primeiro no jogo? A resposta está a sua frente gafanhoto:

GameEngine.getInstance().set**StartingGameStateController**(numero X);
Numero X = número da classe adicionada em addGameStateController.

Basta dizer qual o número da classe que você quer que seja inicializada e pronto. Facim, facim.

Mas e quando eu precisar passar o controle pra outra classe, tipo da TelaInicial para a Jogar?

Hum... pergunta inteligente. Já, esperava por essa!

GameEngine.getInstance().setNextGameStateController(numero X);
Diga qual o número X de chamada da próxima classe e pronto.

Entendi tudo até agora, mas e aquelas paradinhas de teclado?

Paradinhas de teclado? Ah... lembrei... você quer saber quando um tecla foi pressionada né?

```
Keyboard k = GameEngine.getInstance().getKeyboard();
if ( k.keyDown( Keyboard.ENTER_KEY) == true )
```

ou

```
if ( k.keyDown( Keyboard.ENTER_KEY) )
```

tanto faz.

Capturamos um instância da classe Keyboard

```
Keyboard k = GameEngine.getInstance().getKeyboard
```

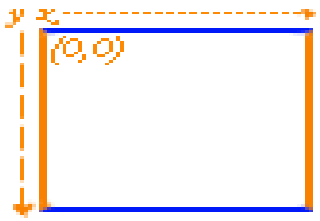
Retorna true se a tecla foi pressionada e false ao contrário.

```
k.keyDown( TECLA ESCOLHIDA )
```

O nosso Keyboard forcesse as seguintes teclas:
Todas maiúsculas: explicação.
up_key: tecla seta pra cima
left_key: tecla seta para esquerda
right_key: tecla seta para direita
down_key: tecla para baixo
escape_key: tecla scape de espaço
space_key: tecla barra de espaço
enter_key:tecla enter

Uhum... entendi essas coisas sobre teclado e pra mover o sprite?

Primeiro, todo Sprite tem suas próprias coordenadas X e Y.
A propósito java diz que a tela tem as seguintes coordenadas:



As coordenadas começam no canto esquerdo da tela
na parte de cima. Ali estão presentes o X=0 e o Y=0
e se estendem até o fim da tela que representando cada eixo.

Mover player para Cima

```
If ( k.keyDown(Keyboard.UP_KEY) == true )
    sprite.y--;
```

Mover player para Baixo

```
if ( k.keyDown(Keyboard.DOWN_KEY) == true )
    sprite.y++;
```

Mover player para Esquerda

```
if( k.keyDown(Keyboard.LEFT_KEY) == true )
     sprite.x--;
```

Mover player para Direita

```
if( k.keyDown(Keyboard.RIGHT_KEY) == true )
    sprite.x++;
```

Legal! Já sei o que é um Sprite, classe de controle, capturar teclas pressionadas e mover um Sprite pela tela, mas e pra carregar as informações na memória, processar as etapas do jogo e desenhar as informações na tela?

Fiquei emocionado agora! Meu pupilo fazendo este tipo de pergunta! Aí estão os métodos que fazem parte da classe GameStateController e são responsáveis por isso.

`public void load() { }` – carrega tudo na memória.
`public void unload() { }` - descarrega tudo da memória.
`public void start() { }` - chamado quando somente algumas coisas devem ser modificadas na memória.
`public void step( int ) { }` – responsável por cada etapa do jogo. Passando o tempo em millisegundos decorrente entre o frame atual e o frame anterior, permitindo assim animações baseadas em tempo.
`public void draw( ) { }` – responsável pelo desenho dos objetos do jogo na tela.

Já vimos várias coisas até agora. Têm algumas coisas da qual ainda não falei:

Classe Game Object

Seguinte, até agora falamos sobre controles, sprites, movimentação etc. mas e sobre os nossos bonecos aqueles nós vamos controlar? Todos esses vêm da classe GameObject, é ela que faz os sprites terem coordenadas X e Y.

Toda classe que usaremos como jogadores, inimigos ou até o fundo têm que estender o GameObject.

No jogo temos: as classes Player e Inimigo estendendo a classe GameObject.

Ah... tá. Então as minhas classes que não serão usadas para controlar o jogo mas sim controladas por outra classe estenderão a classe GameObject.

A classe GameObject terá também os métodos **step** e **draw**.

Sprites

Pra criarmos um sprite devemos fornecer o **caminho** onde o sprite se localiza. Se estiver na mesma pasta do jogo basta colocar o nome da imagem que será usada no sprite. A **quantidade de imagens** em que o sprite é dividido e as suas **dimensões** em pixels.

Podemos utilizar imagens do tipo jpg e png nas imagens dos sprites.

```
sprite = new Sprite ( caminho, quantidadeImagens, alturaPixels, larguraPixels );
ex.: sprite = new Sprite( "imagem.png", 1,  800,  600 );
```

Desenhando um sprite

```
Sprite.draw( g, coordenadaX, coordenadaY) ;
```

As coordenadas X e Y são pontos onde o sprite começara a ser desenhado. O g apresentado é fornecido pela Java, portando, não precisamos nos preocupar com ele.

Ex.: tamanho do sprite = 40x40, X=34, Y=89. O sprite será desenhado inicialmente em 34x89 e terminará em 74x129;

Terminando um jogo

Para terminar um jogo basta usar o seguinte comando em qualquer parte do código

```
GameEngine.getInstance().requestShutdown();
```