Arcabouço para Jogos e Jogo Educacional em Ruby

Aluno: Victor David Santos Orientador: Prof. Dr. Marco Dimas Gubitoso

O Arcabouço MiniGL

Biblioteca em Ruby para desenvolvimento de jogos com alta produtividade, que fornece uma sintaxe sucinta e encapsula muitos dos elementos recorrentes em jogos, como animações, física (checagem de colisões), gerenciamento do campo de visão, interface de usuário, entre outras.

Um jogo construído com MiniGL tem como principal componente uma janela, que deve ser uma instância de uma classe derivada de Gosu::Window (Gosu é o nome da biblioteca sobre a qual a MiniGl foi construída):

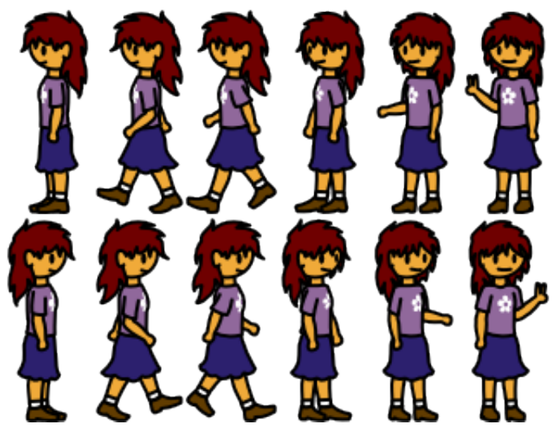
```
class MyGame < Gosu::Window
  def initialize
    super 800, 600, false # criando a janela
    Game.initialize self  # inicializando a MiniGL
  end
  def update
    # lógica de atualização a cada quadro
  end
  def draw
    # instruções de desenho de elementos
  end
end

MyGame.new.show
```

Após a chamada de 'show', a janela começa a executar um laço que chama as funções 'update' e 'draw' aproximadamente sessenta vezes por segundo, ou seja, o jogo começa a rodar a 60 *FPS*.

Abaixo, uma descrição mais detalhada dos principais “módulos” que compõem o arcabouço.

Controle de Animações

Utilização do conceito de *spritesheets* para fornecer uma interface de fácil utilização para animações, através da classe Sprite e suas derivadas.



Utilizando a imagem acima e o trecho de código abaixo, pode-se obter o efeito da menina caminhando para a direita:

```
# dentro do 'initialize' da janela
s = Sprite.new 0, 0, :sprite_Milena, 6, 2

# dentro do 'update' da janela
s.animate [6, 7, 6, 8], 10

# dentro do 'draw' da janela
s.draw
```

Os parâmetros de 'animate' são uma lista de índices (os índices das *sprites* são contados de cima para baixo e da esquerda para a direita) e o intervalo entre cada troca de índice, em quadros (ou *frames*).

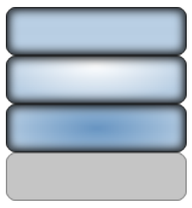
Eventos de teclado e mouse

A biblioteca oferece um jeito fácil de detectar diversos tipos de eventos do teclado e do mouse, conforme exemplos a seguir:

```
KB.key_down? Gosu::KbLeft # tecla direcional para a
                          esquerda está pressionada?
KB.key_pressed? Gosu::KbReturn # tecla Enter foi
                               pressionada neste exato
                               quadro?
KB.key_released? Gosu::KbSpace # barra de espaço foi
                               solta neste exato
                               quadro?
KB.key_held? Gosu::KbA # tecla A está sendo mantida
                       pressionada há mais de x
                       quadros (onde x é
                       configurável)?

Mouse.button_down? :middle   # mesmo conceito que serve
Mouse.button_pressed? :right # para o teclado, mas com
Mouse.button_released? :left # os botões do mouse

Mouse.double_click? :left # o usuário acaba de fazer um
                          # duplo clique do botão
                          # esquerdo?
```

Interface de Usuário

O arcabouço oferece prontas classes que representam elementos comuns de interface de usuário como botões e campos de texto.



A imagem acima é uma *spritesheet* a ser usada para representar os diversos estados de um botão (normal, quando o mouse passa por cima, quando é clicado e quando está desabilitado).

Com o uso das *closures* de Ruby, a definição da ação do botão é feita de maneira bastante intuitiva, passando-se o bloco de código direto para o construtor:

```
b = Button.new 0, 0, :ui_btn1, font1, 'Texto' {
  # ação de clique aqui
}
```

Os parâmetros do exemplo acima definem a posição, a imagem, a fonte e o texto do botão, mas é possível definir também, a cor da fonte, o posicionamento do texto em relação à imagem e muito mais.

Outro elemento interessante é o campo de texto, que pode ser instanciado como a seguir:

```
t = TextField.new 0, 0, font2, :ui_txtField {
  # ação de texto alterado aqui (opcional)
}
```

Física básica

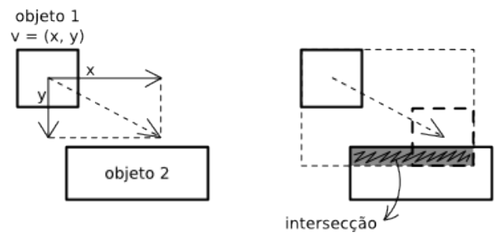
O módulo de física básica do arcabouço fornece uma sintaxe simples para movimentação de objetos baseada em forças e também checagem e resposta a colisões.

A qualquer objeto que inclua o módulo 'Movement', é acrescentado o método 'move', usado como a seguir:

```
obj.move forces, obstacles, ramps
```

'forces' deve ser um vetor com as componentes horizontal e vertical da força resultante sobre o objeto (a força da gravidade é incluída por padrão); 'obstacles' é uma lista de objetos que respondam aos métodos 'x', 'y', 'w', e 'h', indicando sua posição e tamanho, e 'ramps' deve ser uma lista de objetos da classe Ramp, definida na biblioteca MiniGL.

A checagem e resposta a colisões é baseada em caixas retangulares, e funciona como ilustrado abaixo:



Alinhamento de texto e quebras de linha

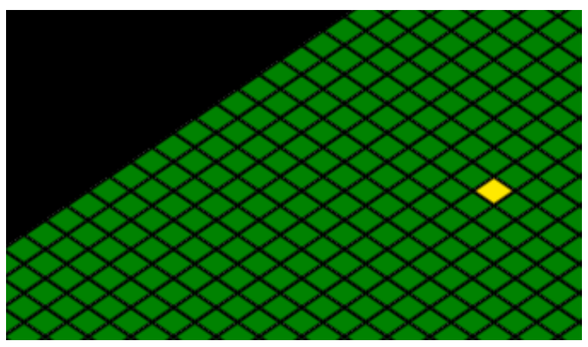
A biblioteca fornece a classe TextHelper, que facilita a escrita de texto com múltiplas linhas e diversos tipos de alinhamento, além de poder definir a cor e a opacidade.

O trecho de código abaixo, por exemplo, resultará num trecho de texto justificado a largura de 400 pixels, escrito em vermelho escuro (indicado pelo hexadecimal 0x800000):

```
th = TextHelper.new font1, 10
th.write_breaking
  'Este texto deve ser exibido com quebras de linha e\
   justificado.\nQuebras de linha explícitas devem\
   ser respeitadas.', x, y, 400, :justified, 0x800000
```

Campo de visão (*viewport*) e *tiles*

O arcabouço encapsula na classe Map a lógica de utilização de tiles (isto é, a representação do cenário como uma grade de casas retangulares), além do controle de campo de visão (quando o cenário inteiro não cabe na tela, é necessário movimentar o campo de visão do usuário).



A classe pode ser usada tanto para mapas de *tiles* ortogonais (isto é, paralelos aos eixos da janela) quanto para isométricos, como no exemplo acima.

Gerenciamento de recursos

Este módulo oferece uma sintaxe bastante resumida para carregamento de recursos como imagens, sons e fontes. Além disso, seu uso, em comparação com o método fornecido originalmente pela biblioteca Gosu, garante menos consumo de CPU e de memória.

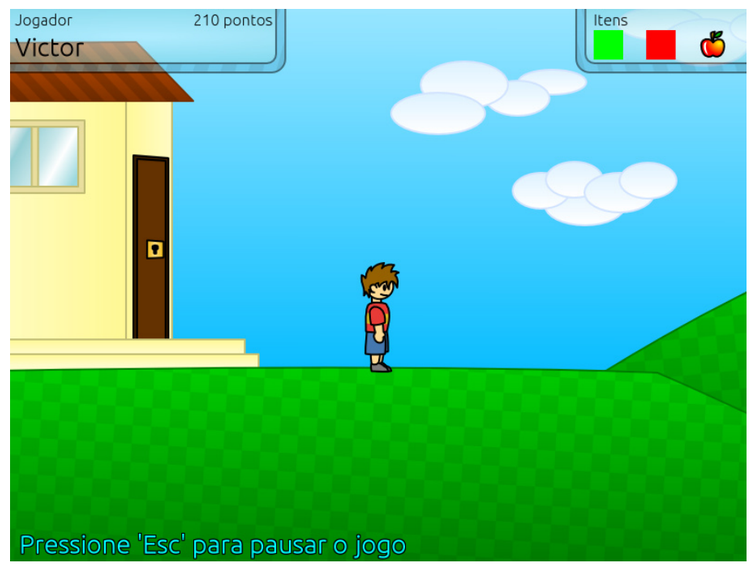
```
img1 = Res.img :path_to_img1
sound1 = Res.sound path_to_sound1
font1 = Res.font :path_to_font
```

Os caminhos dos arquivos indicados pelos símbolos acima são relativos a pastas padronizadas por tipo de recurso (por exemplo, 'data/img' para as imagens, 'data/sound' para os sons, etc.). As extensões também podem ser omitidas quando correspondem à extensão padrão (por exemplo, '.png' para imagens).

O Jogo Aventura do Saber

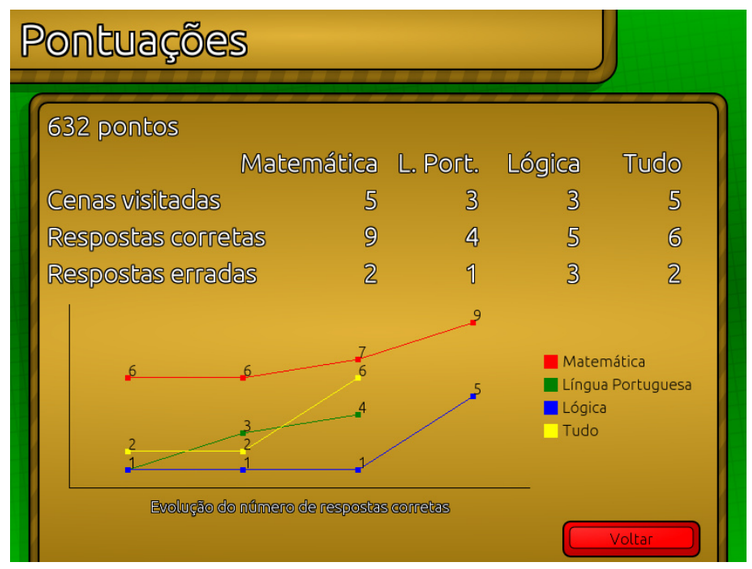
Jogo educacional de gênero aventura/RPG, foi concebido para demonstrar a possibilidade de utilização da linguagem Ruby para jogos, a funcionalidade do arcabouço MiniGL e, é claro, também devido à contribuição social trazida por um jogo educacional.

É focado em matemática, língua portuguesa e lógica. A jogabilidade consiste do personagem movendo-se pelo cenário e conversando com outros personagens, interagindo com objetos e coletando itens.



O jogo possui um sistema de pontuação que permite a posterior avaliação, por exemplo por um educador, do desempenho dos estudantes ao responder aos desafios do jogo – os outros personagens fazem perguntas relacionadas às áreas de conhecimento mencionadas, às quais o usuário deve responder corretamente para prosseguir na aventura.

Há uma tela com um resumo dos números para cada jogador, que pode ser usada pelos educadores para ter uma visualização rápida do perfil de cada aluno:



Os jogadores devem identificar-se por seu nome, podem escolher entre dois personagens para protagonizar a aventura, e também escolher entre 'missões' relacionadas a cada uma das áreas de conhecimento (ou uma missão que engloba as três simultaneamente).

O progresso de jogo é salvo automaticamente ao sair, e para retomar o jogo salvo o usuário só precisa clicar em seu nome na tela de identificação:

O jogo foi desenvolvido de modo que os elementos interativos (outros personagens, objetos e itens) são todos definidos por arquivos de texto, permitindo que ele seja estendido ou modificado conforme a necessidade, sem demandar o uso de programação.

Links

O projeto do arcabouço e do jogo estão ambos disponíveis no GitHub, nos links abaixo:

- https://github.com/victords/minigl
- https://github.com/victords/aventura-do-saber

Adicionalmente, o arcabouço está disponível como uma *gem* (minigl) no repositório RubyGems.