INSTITUTO FEDERAL DE
EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
RIO GRANDE DO NORTE

Programação de Computadores

Um pouco de diversão - libGOSU

O exemplo desenvolvido nesta apresentação é o mesmo do tutorial GOSU, que pode ser acessado em: https://github.com/jlnr/gosu/wiki/Ruby-Tutorial

O que veremos hoje?

* Introdução
* Biblioteca GOSU
  * A classe principal do jogo
    * Construtor, draw e update
    * Imagens
    * Movimentação
* Construção de jogo
  * Exemplo passo a passo

Introdução

* libgosu
  * Desenvolvimento de jogos
  * 2D
  * C++/Ruby
* Onde encontrar?
  * http://www.libgosu.org
* Instalação
  * sudo gem install gosu


```
jorgiano — bash — 69×7
bash | bash

ari-pc:~ jorgiano$ sudo gem install gosu
Successfully installed gosu-0.7.45-universal-darwin
1 gem installed
Installing ri documentation for gosu-0.7.45-universal-darwin...
Installing RDoc documentation for gosu-0.7.45-universal-darwin...
ari-pc:~ jorgiano$
```

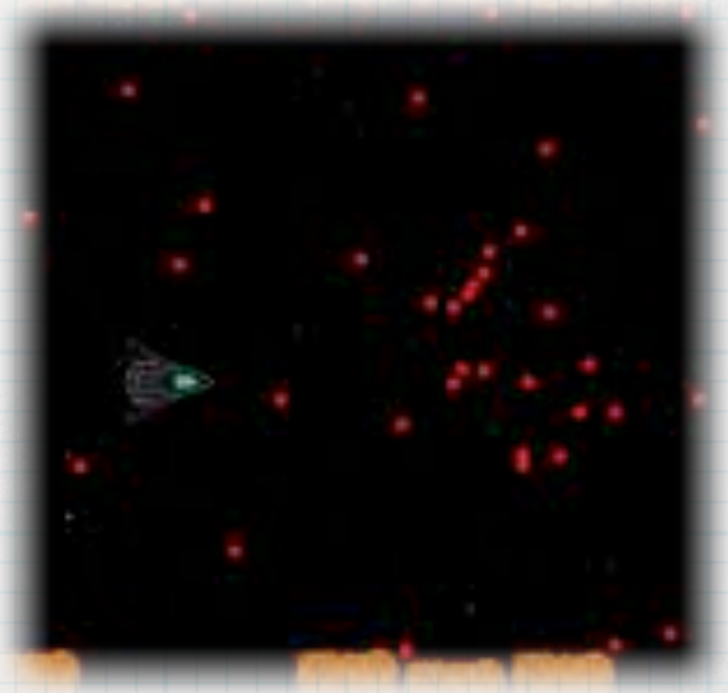
Introdução

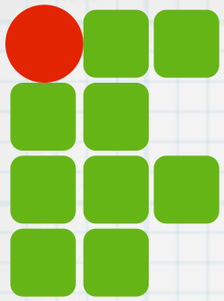
* libgosu
  * Desenvolvimento de jogos
  * 2D
  * C++/Ruby
* Onde encontrar?
  * http://www.libgosu.org
* Instalação
  * sudo gem install gosu


```
jorgiano — bash — 69×7
bash | bash

ari-pc:~ jorgiano$ sudo gem install gosu
Successfully installed gosu-0.7.45-universal-darwin
1 gem installed
Installing ri documentation for gosu-0.7.45-universal-darwin...
Installing RDoc documentation for gosu-0.7.45-universal-darwin...
ari-pc:~ jorgiano$
```

Introdução

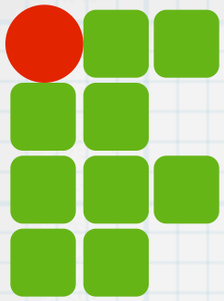
* Complementos
  * Adicionam funcionalidades avançadas
* Exemplos:
  * texplay (desenho)
  * chingu (lógica de jogos)
  * chipmunk (física - e.g. gravidade)





Gosu

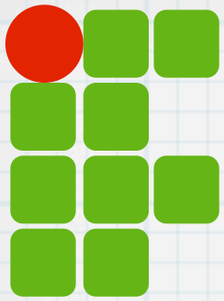
```
require 'gosu'

class CataEstrela < Gosu::Window

  def initialize
    super(640,480,false)
    self.caption = "Cata Estrelas"

  end

  def draw
  end

  def update
  end

end
```



Gosu

Usar biblioteca Gosu

```
require 'gosu'

class CataEstrela < Gosu::Window

  def initialize
    super(640,480,false)
    self.caption = "Cata Estrelas"

  end

  def draw
  end

  def update
  end

end
```

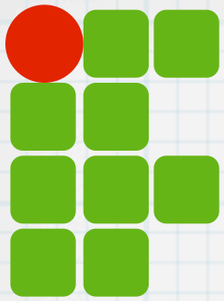


Gosu

Usar biblioteca Gosu

Classe que representa o jogo

```
require 'gosu'

class CataEstrela < Gosu::Window

  def initialize
    super(640,480,false)
    self.caption = "Cata Estrelas"

  end

  def draw
  end

  def update
  end

end
```



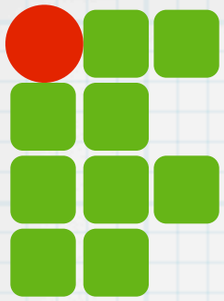
Gosu

Usar biblioteca Gosu

Classe que representa o jogo

Construtor

```
require 'gosu'

class CataEstrela < Gosu::Window

  def initialize
    super(640,480,false)
    self.caption = "Cata Estrelas"

  end

  def draw
  end

  def update
  end

end
```



Gosu

Usar biblioteca Gosu

Classe que representa o jogo

Construtor

Desenha a janela

```
require 'gosu'

class CataEstrela < Gosu::Window

  def initialize
    super(640,480,false)
    self.caption = "Cata Estrelas"

  end

  def draw
  end

  def update
  end

end
```

Gosu

Usar biblioteca Gosu

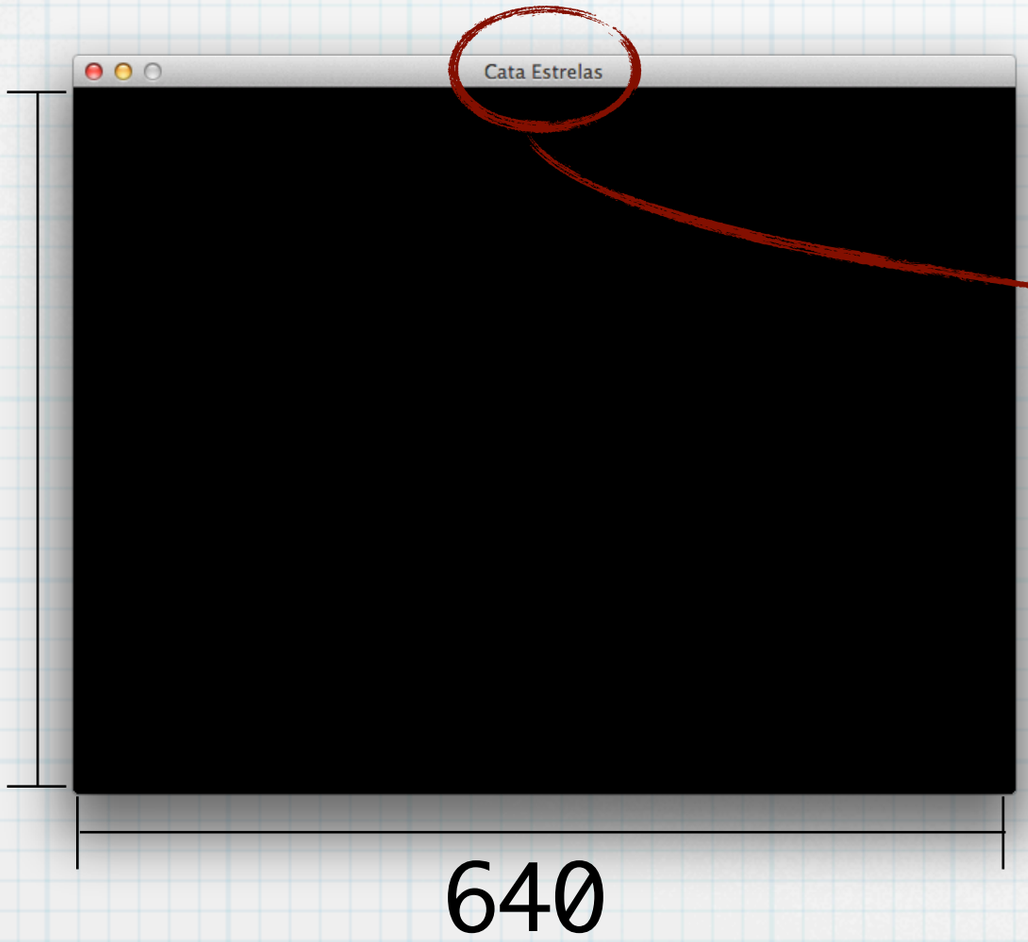
Classe que representa o jogo

Construtor

Desenha a janela

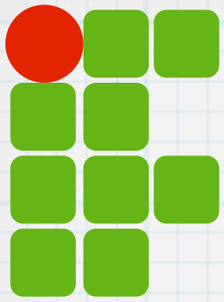
Lógica do jogo

```
require 'gosu'

class CataEstrela < Gosu::Window

  def initialize
    super(640,480,false)
    self.caption = "Cata Estrelas"

  end

  def draw
  end

  def update
  end

end
```



Gosu

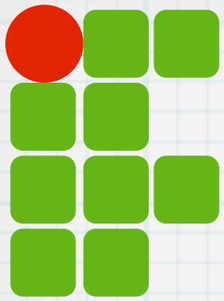
```
require 'gosu'

class CataEstrela < Gosu::Window

  def initialize
    super(640,480,false)
    self.caption= "Cata Estrelas"

  end

  def draw
  end

  def update
  end

end
```



Draw

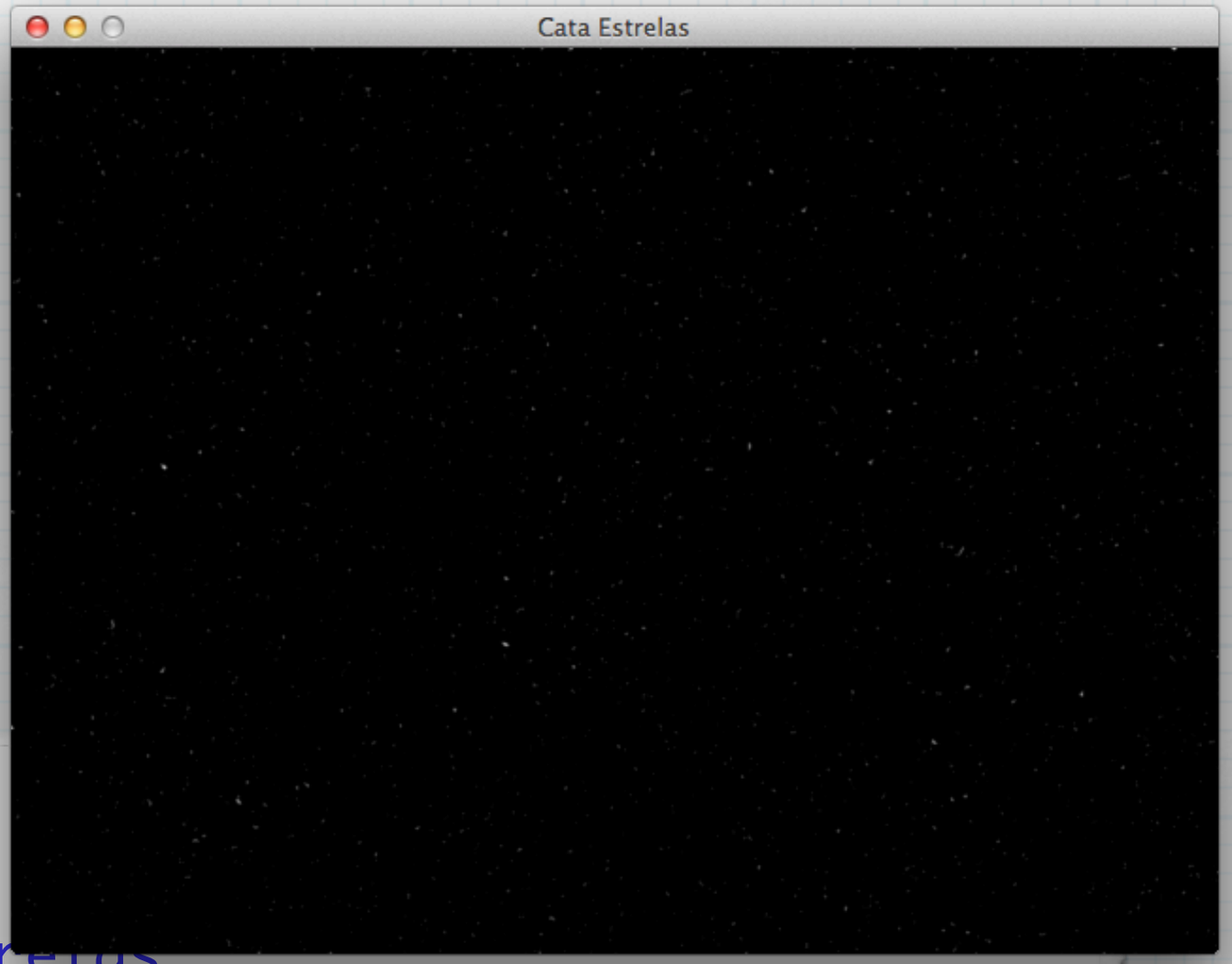
* Método que desenha a janela
  * Chamado quando for necessário desenhar
  * **TUDO** deve ser redesenhado
* Imagens em arquivos

```
def initialize
  super(640,480,false)
  self.caption = "Cata Estrelas"
  @imagem_fundo = Gosu::Image.new(self, "Space.png", true)
end

def draw
  @imagem_fundo.draw(0,0,0)
end
```

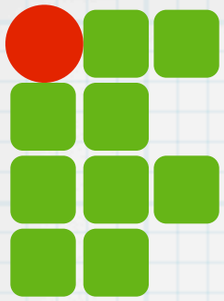


Draw

* Método que desenha a janela
  * Chamado quando for necessário desenhar
  * **TUDO** deve ser redesenhado
* Imagens em arquivos




```
def initialize
  super(640,480,false)
  self.caption = "Cata Estrelas"
  @imagem_fundo = Gosu::Image.new(self, "Space.png", true)
end

def draw
  @imagem_fundo.draw(0,0,0)
end
```

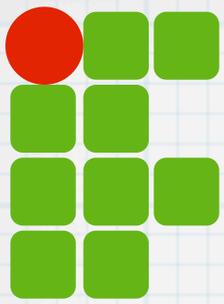


Draw

* Um jogador
  * Imagem em arquivo

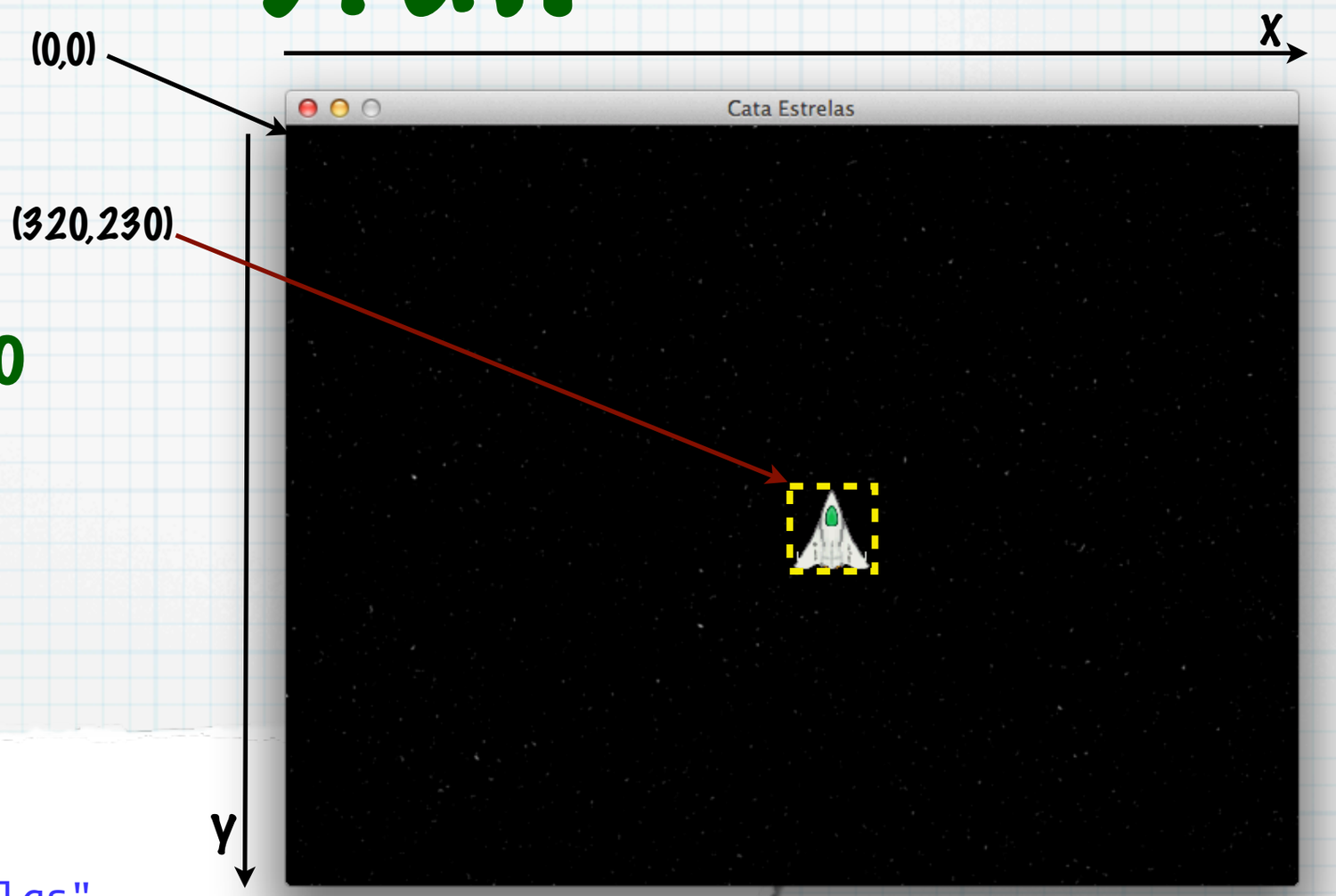
```
def initialize
  super(640,480,false)
  self.caption = "Cata Estrelas"
  @imagem_fundo = Gosu::Image.new(self, "Space.png", true)
  @jogador = Gosu::Image.new(self,"Nave.bmp",true)
end

def draw
  @imagem_fundo.draw(0,0,0)
  @jogador.draw(320,230,1)
end
```

draw (x,y,z):
- x é o valor, em pixel, no eixo x
- y é o valor, em pixel, no eixo y
- z é a profundidade



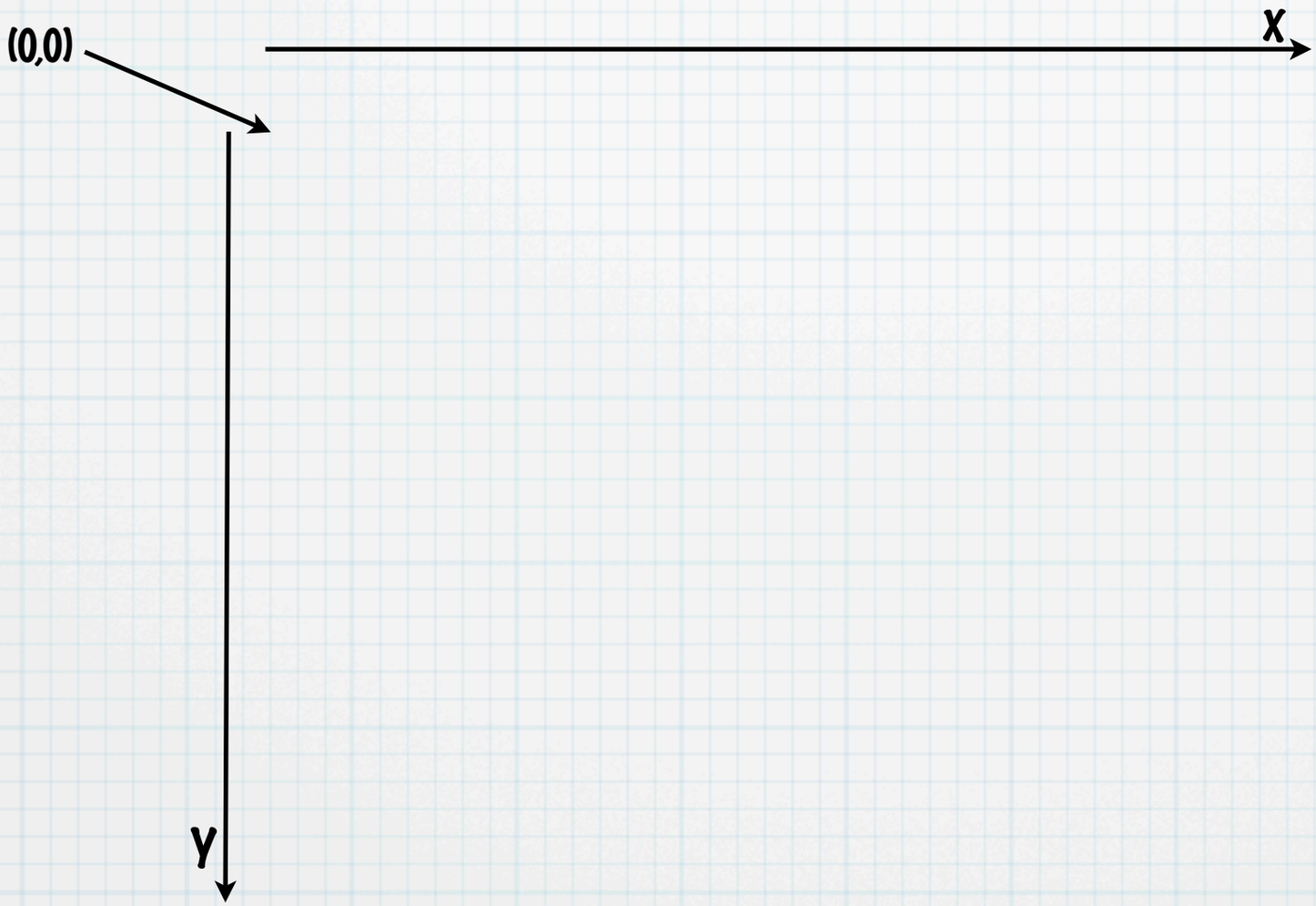
Draw

* Um jogador
  * Imagem em arquivo

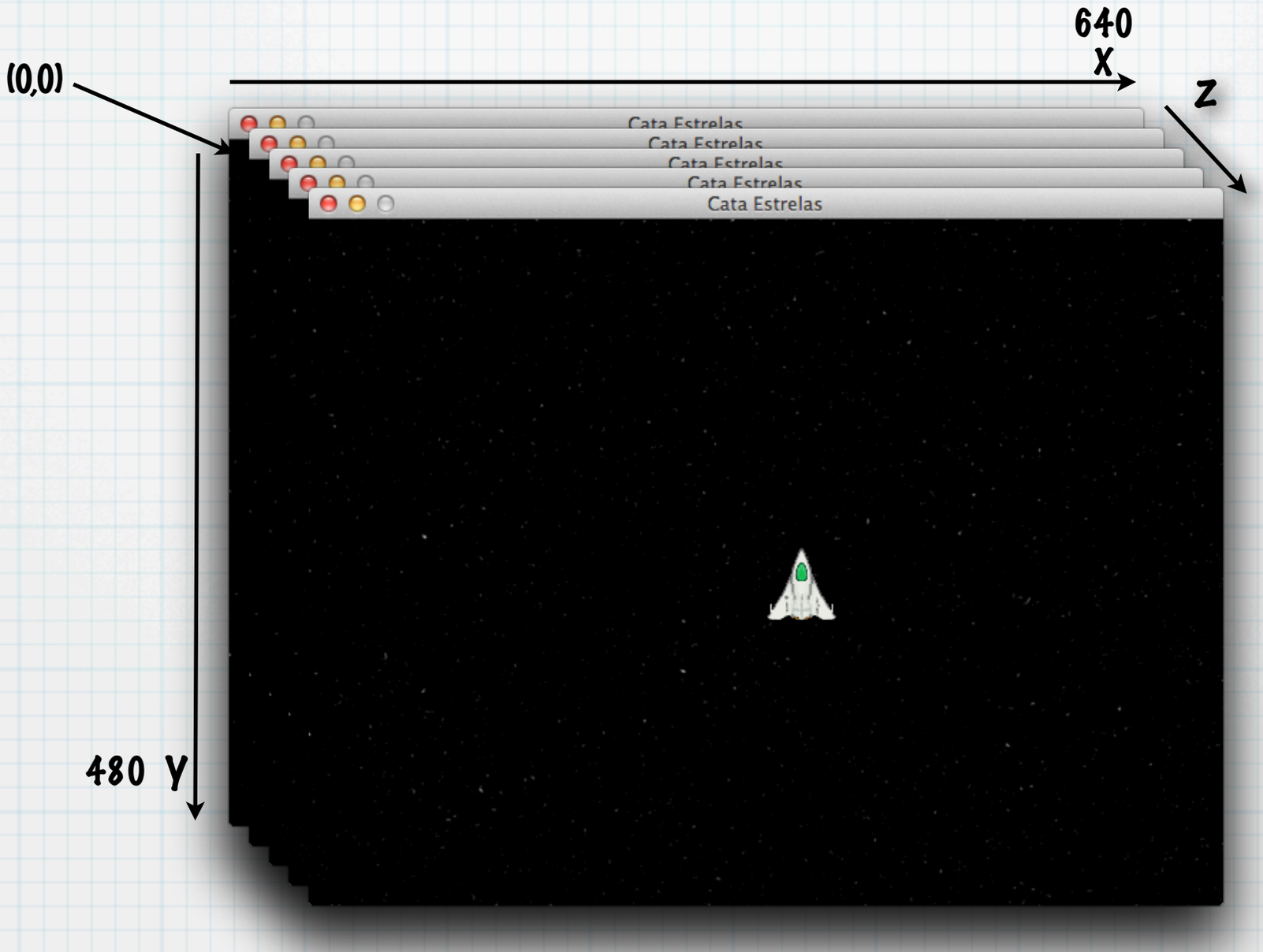



```
def initialize
  super(640,480,false)
  self.caption = "Cata Estrelas"
  @imagem_fundo = Gosu::Image.new(self, "Space.png", true)
  @jogador = Gosu::Image.new(self,"Nave.bmp",true)
end

def draw
  @imagem_fundo.draw(0,0,0)
  @jogador.draw(320,230,1)
end
```

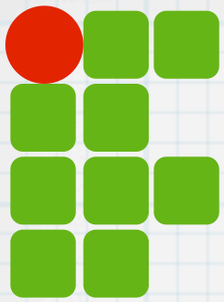
draw (x,y,z):
- x é o valor, em pixel, no eixo x
- y é o valor, em pixel, no eixo y
- z é a profundidade



Coordenadas



Coordenadas



eixo z

eixo z

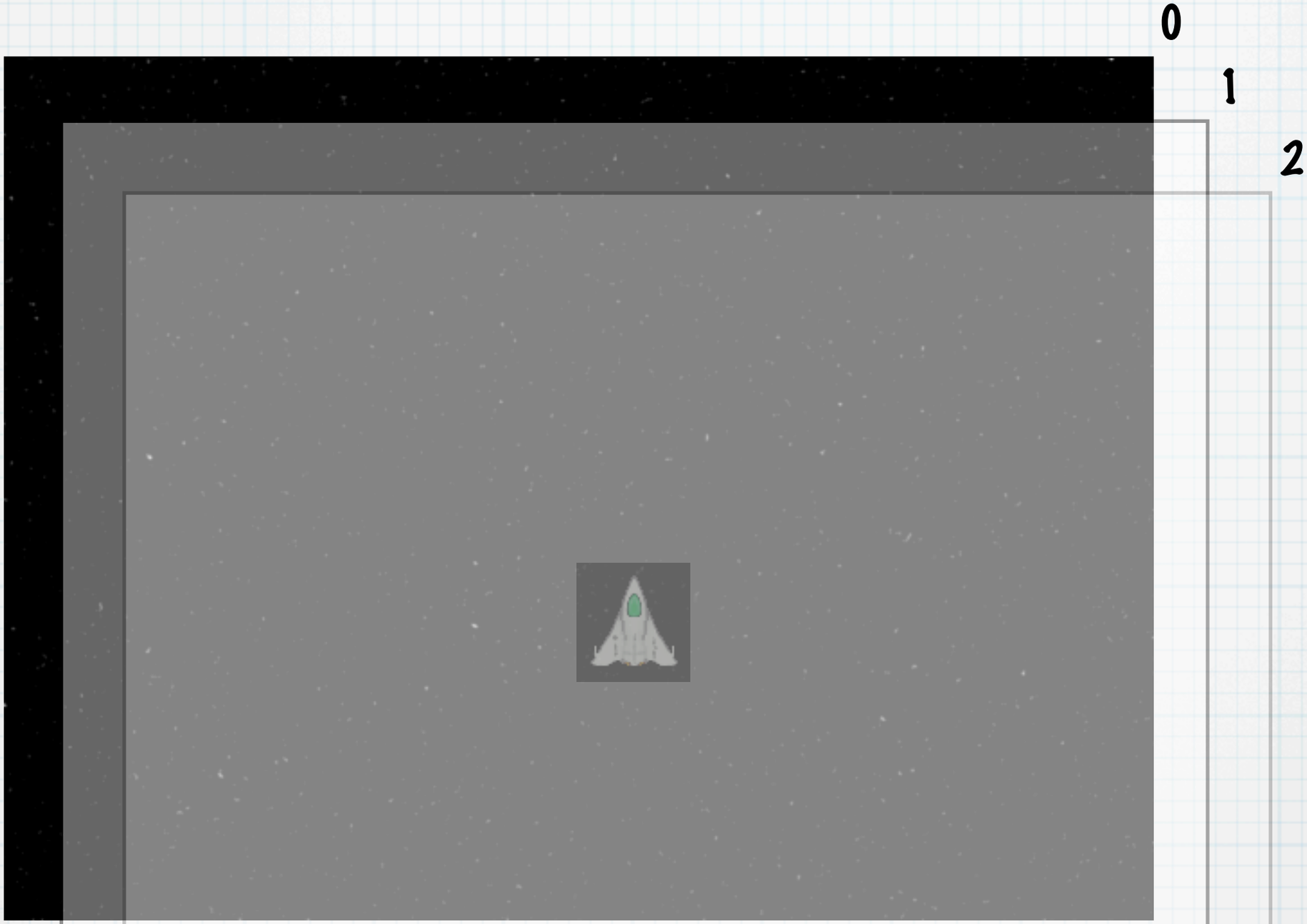
0

eixo z

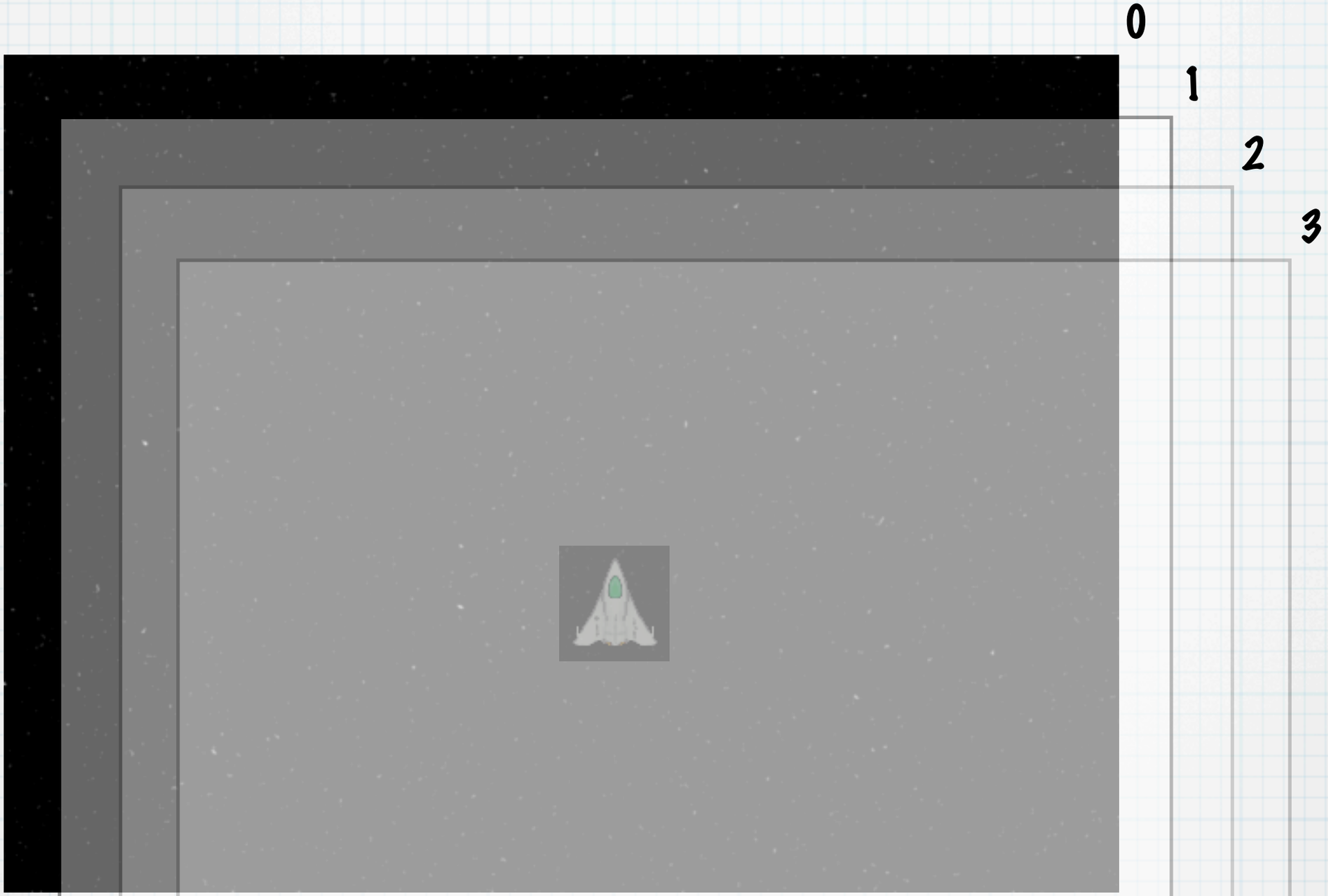
0

eixo z

eixo z

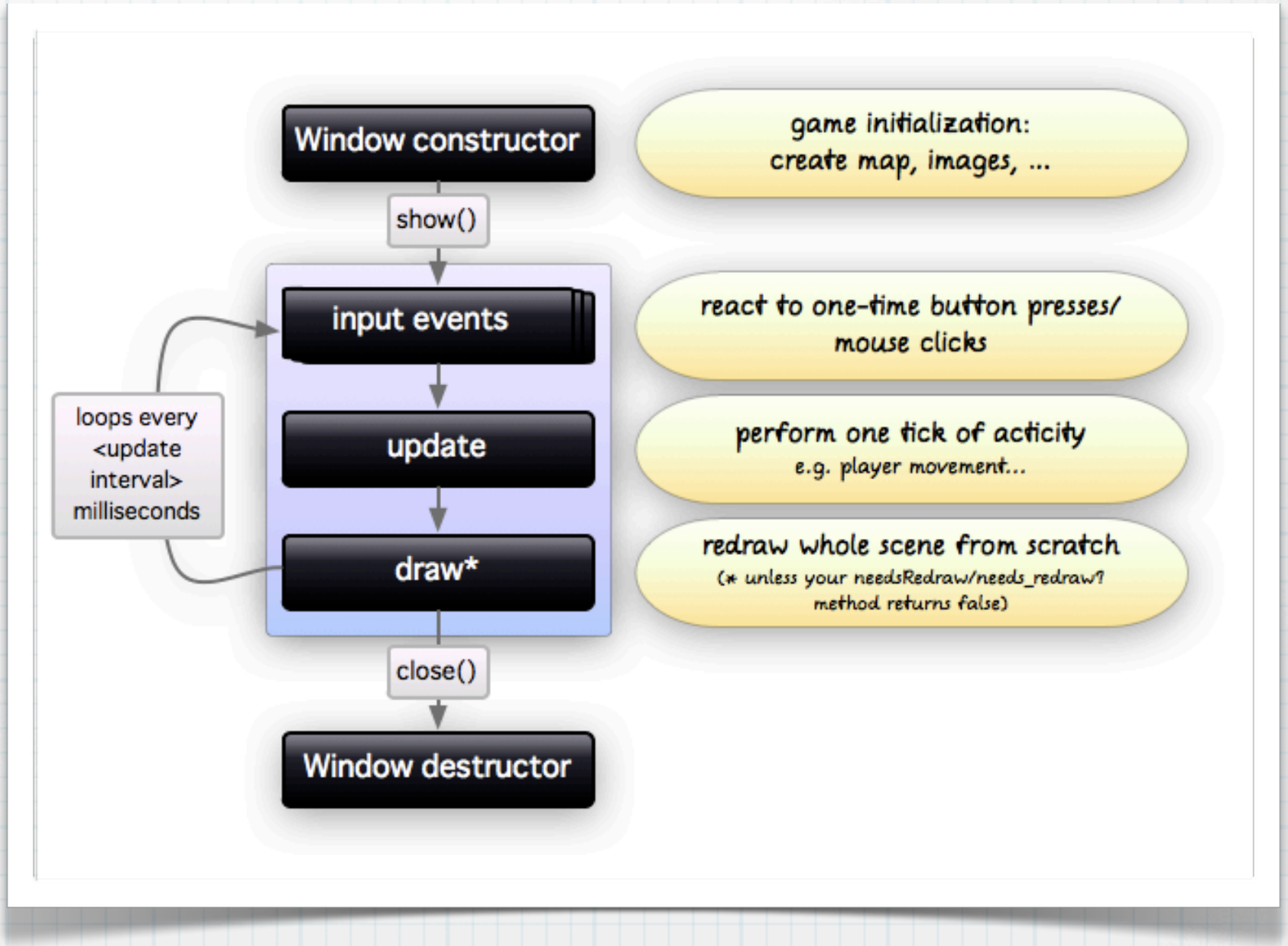


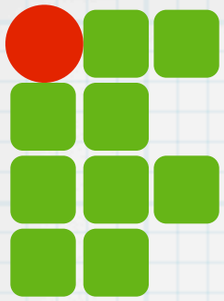
eixo z



Update

* Dinâmica do jogo
  * Chamado 60 vezes por segundo
  * Configurável
* o método `draw` é sempre chamado após o `update`

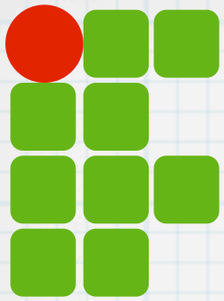




Animação

* Movendo o jogador com o update
  * @pos_x: posição no eixo x

```
class CataEstrela < Gosu::Window
  def initialize
    super(640,480,false)
    self.caption = "Cata Estrela"
    @imagem_fundo = Gosu::Image.new(self, "Space.png", true)
    @jogador = Gosu::Image.new(self,"Nave.bmp",true)
    @pos_x = 320
  end
  def draw
    @imagem_fundo.draw(0,0,0)
    @jogador.draw(@pos_x,230,1)
  end
  def update
    if ( button_down?(Gosu::Button::KbRight) ) then
      @pos_x = @pos_x + 10
      if (@pos_x > 630) then @pos_x = 630 end
    end
    if ( button_down?(Gosu::Button::KbLeft) ) then
      @pos_x = @pos_x - 10
      if (@pos_x < 10) then @pos_x = 10 end
    end
  end
end
```



Animação

* Movendo o jogador com o update
  * @pos_x: posição no eixo x

```
class CataEstrela < Gosu::Window
  def initialize
    super(640,480,false)
    self.caption = "Cata Estrela"
    @imagem_fundo = Gosu::Image.new(self, "Space.png", true)
    @jogador = Gosu::Image.new(self,"Nave.bmp",true)
    @pos_x = 320
  end
  def draw
    @imagem_fundo.draw(0,0,0)
    @jogador.draw(@pos_x,230,1)
  end
  def update
    if ( button_down?(Gosu::Button::KbRight) ) then
      @pos_x = @pos_x + 10
      if (@pos_x > 630) then @pos_x = 630 end
    end
    if ( button_down?(Gosu::Button::KbLeft) ) then
      @pos_x = @pos_x - 10
      if (@pos_x < 10) then @pos_x = 10 end
    end
  end
end
```

desenha na posição @pos_x

Animação

* Movendo o jogador com o update
  * @pos_x: posição no eixo x

```
class CataEstrela < Gosu::Window
  def initialize
    super(640,480,false)
    self.caption = "Cata Estrela"
    @imagem_fundo = Gosu::Image.new(self, "Space.png", true)
    @jogador = Gosu::Image.new(self,"Nave.bmp",true)
    @pos_x = 320
  end
  def draw
    @imagem_fundo.draw(0,0,0)
    @jogador.draw(@pos_x,230,1)
  end
  def update
    if ( button_down?(Gosu::Button::KbRight) ) then
      @pos_x = @pos_x + 10
      if (@pos_x > 630) then @pos_x = 630 end
    end
    if ( button_down?(Gosu::Button::KbLeft) ) then
      @pos_x = @pos_x - 10
      if (@pos_x < 10) then @pos_x = 10 end
    end
  end
end
```

desenha na posição @pos_x

Verifica se seta para direita está pressionada

Animação

* Movendo o jogador com o update
  * @pos_x: posição no eixo x

```
class CataEstrela < Gosu::Window
  def initialize
    super(640,480,false)
    self.caption = "Cata Estrela"
    @imagem_fundo = Gosu::Image.new(self, "Space.png", true)
    @jogador = Gosu::Image.new(self,"Nave.bmp",true)
    @pos_x = 320
  end
  def draw
    @imagem_fundo.draw(0,0,0)
    @jogador.draw(@pos_x,230,1)
  end
  def update
    if ( button_down?(Gosu::Button::KbRight) ) then
      @pos_x = @pos_x + 10
      if (@pos_x > 630) then @pos_x = 630 end
    end
    if ( button_down?(Gosu::Button::KbLeft) ) then
      @pos_x = @pos_x - 10
      if (@pos_x < 10) then @pos_x = 10 end
    end
  end
end
```

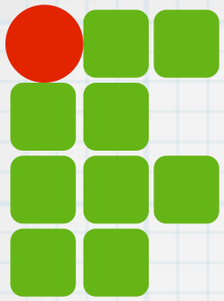
desenha na posição @pos_x

Verifica se seta para direita está pressionada

Verifica se seta para esquerda está pressionada

DEMO

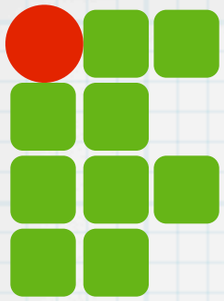
v0



Posição na tela

* Tamanho da janela
  * self.height
  * self.width
* Tamanho da imagem
  * imagem.height
    * @jogador.height
  * imagem.width
    * @jogador.width

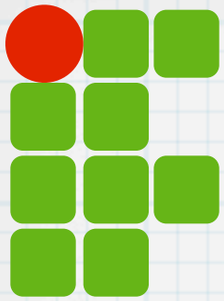
```
class CataEstrela < Gosu::Window
  def initialize
    super(640,480,false)
    self.caption = "Cata Estrelas"
    @imagem_fundo = Gosu::Image.new(self, "Space.png", true)
    @jogador = Gosu::Image.new(self,"Nave.bmp",true)
    @pos_x = 10
    @pos_y = self.height / 2
  end

  def draw
    @imagem_fundo.draw(0,0,0)
    @jogador.draw(@pos_x,@pos_y,1)
  end

  def update
    if ( button_down?(Gosu::Button::KbRight) ) then
      @pos_x = @pos_x + 10
      if (@pos_x > self.width-(@jogador.width+10) ) then
        @pos_x = self.width-(@jogador.width+10)
      end
    end
    if ( button_down?(Gosu::Button::KbLeft) ) then
      @pos_x = @pos_x - 10
      if (@pos_x < 10) then @pos_x = 10 end
    end
  end
end
```



Posição na tela

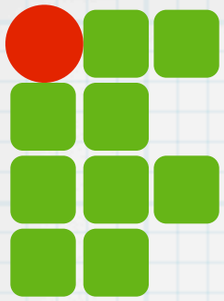
* Tamanho da janela
  * self.height
  * self.width
* Tamanho da imagem
  * imagem.height
    * @jogador.height
  * imagem.width
    * @jogador.width

```
class CataEstrela < Gosu::Window
  def initialize
    super(640,480,false)
    self.caption = "Cata Estrelas"
    @imagem_fundo = Gosu::Image.new(self, "Space.png", true)
    @jogador = Gosu::Image.new(self,"Nave.bmp",true)
    @pos_x = 10
    @pos_y = self.height / 2
  end

  def draw
    @imagem_fundo.draw(0,0,0)
    @jogador.draw(@pos_x,@pos_y,1)
  end

  def update
    if ( button_down?(Gosu::Button::KbRight) ) then
      @pos_x = @pos_x + 10
      if (@pos_x > self.width-(@jogador.width+10) ) then
        @pos_x = self.width-(@jogador.width+10)
      end
    end
    if ( button_down?(Gosu::Button::KbLeft) ) then
      @pos_x = @pos_x - 10
      if (@pos_x < 10) then @pos_x = 10 end
    end
  end
end
```



Movimento completo

```
class CataEstrela < Gosu::Window
  def initialize
    super(640,480,false)
    self.caption = "Cata Estrelas"
    @imagem_fundo = Gosu::Image.new(self, "Space.png", true)
    @jogador = Gosu::Image.new(self,"Nave.bmp",true)
    @pos_x = self.width / 2
    @pos_y = self.height / 2
  end

  def draw
    @imagem_fundo.draw(0,0,0)
    @jogador.draw(@pos_x,@pos_y,1)
  end

  ...
```

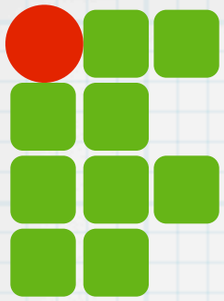


Movimento completo

```
...
def update
  if ( button_down?(Gosu::Button::KbRight) ) then
    @pos_x = @pos_x + 10
    if (@pos_x > self.width-(@jogador.width+10) ) then
      @pos_x = self.width-(@jogador.width+10)
    end
  end
  if ( button_down?(Gosu::Button::KbLeft) ) then
    @pos_x = @pos_x - 10
    if (@pos_x < 10) then @pos_x = 10 end
  end
  if ( button_down?(Gosu::Button::KbDown) ) then
    @pos_y = @pos_y + 10
    if (@pos_y > self.height-(@jogador.height+10) ) then
      @pos_y = self.height-(@jogador.height+10)
    end
  end
  if ( button_down?(Gosu::Button::KbUp) ) then
    @pos_y = @pos_y - 10
    if (@pos_y < 10) then @pos_y = 10 end
  end
 end
end
```

DEMO

v1



Organizando

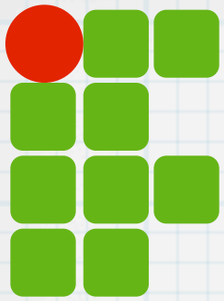
* Colocar o jogo e o jogador em classes separadas

jogador.rb

```
require 'gosu'
class Jogador
 def initialize (janela)
  @janela = janela
  @imagem = Gosu::Image.new(@janela,"Nave.bmp",true)
  @x = @janela.width/2
  @y = @janela.height/2
 end
 def draw
  @imagem.draw(@x,@y,1)
 end
 def mover_direita
  @x = @x + 5
  if (@x>@janela.width-(@imagem.width+10) ) then
   @x = @janela.width-(@imagem.width+10)
  end
 end
 def mover_esquerda
  @x = @x - 5
  if (@x < 10) then @x = 10 end
 end
 ...
end
```

Cata_estrela.rb

```
$LOAD_PATH << '.'

require 'gosu'
require 'jogador'
class CataEstrela < Gosu::Window
 def initialize
  super(640,480,false)
  self.caption = "Cata Estrelas"
  @imagem_fundo = Gosu::Image.new(self, "Space.png", true)
  @jogador = Jogador.new(self)
 end
  def draw
    @imagem_fundo.draw(0,0,0)
    @jogador.draw()
  end
  def update
    if ( button_down?(Gosu::Button::KbRight) ) then
      @jogador.mover_direita
    end
    if ( button_down?(Gosu::Button::KbLeft) ) then
      @jogador.mover_esquerda
    end
   ...
  end
end
```



Organizando

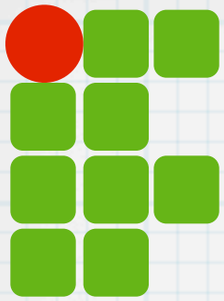
* Colocar o jogo e o jogador em classes separadas

jogador.rb

```
require 'gosu'
class Jogador
 def initialize (janela)
  @janela = janela
  @imagem = Gosu::Image.new(@janela,"Nave.bmp",true)
  @x = @janela.width/2
  @y = @janela.height/2
 end
 def draw
  @imagem.draw(@x,@y,1)
 end
 def mover_direita
  @x = @x + 5
  if (@x>@janela.width-(@imagem.width+10) ) then
   @x = @janela.width-(@imagem.width+10)
  end
 end
 def mover_esquerda
  @x = @x - 5
  if (@x < 10) then @x = 10 end
 end
 ...
end
```

Cata_estrela.rb

```
$LOAD_PATH << '.'

require 'gosu'
require 'jogador'
class CataEstrela < Gosu::Window
 def initialize
  super(640,480,false)
  self.caption = "Cata Estrelas"
  @imagem_fundo = Gosu::Image.new(self, "Space.png", true)
  @jogador = Jogador.new(self)
 end
  def draw
    @imagem_fundo.draw(0,0,0)
    @jogador.draw()
  end
  def update
    if ( button_down?(Gosu::Button::KbRight) ) then
      @jogador.mover_direita
    end
    if ( button_down?(Gosu::Button::KbLeft) ) then
      @jogador.mover_esquerda
    end
    ...
  end
end
```



Organizando

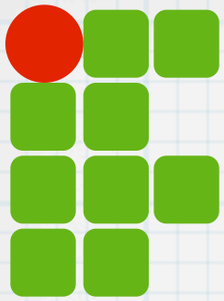
* Colocar o jogo e o jogador em classes separadas

jogador.rb

```
require 'gosu'
class Jogador
 def initialize (janela)
  @janela = janela
  @imagem = Gosu::Image.new(@janela,"Nave.bmp",true)
  @x = @janela.width/2
  @y = @janela.height/2
 end
 def draw
  @imagem.draw(@x,@y,1)
 end
 def mover_direita
  @x = @x + 5
  if (@x>@janela.width-(@imagem.width+10) ) then
   @x = @janela.width-(@imagem.width+10)
  end
 end
 def mover_esquerda
  @x = @x - 5
  if (@x < 10) then @x = 10 end
 end
 ...
end
```

Cata_estrela.rb

```
$LOAD_PATH << '.'

require 'gosu'
require 'jogador'
class CataEstrela < Gosu::Window
 def initialize
  super(640,480,false)
  self.caption = "Cata Estrelas"
  @imagem_fundo = Gosu::Image.new(self, "Space.png", true)
  @jogador = Jogador.new(self)
 end
  def draw
    @imagem_fundo.draw(0,0,0)
    @jogador.draw()
  end
  def update
    if ( button_down?(Gosu::Button::KbRight) ) then
      @jogador.mover_direita
    end
    if ( button_down?(Gosu::Button::KbLeft) ) then
      @jogador.mover_esquerda
    end
   ...
  end
end
```



Organizando

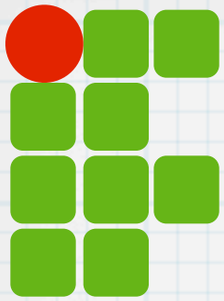
* Colocar o jogo e o jogador em classes separadas

jogador.rb

```
require 'gosu'
class Jogador
 def initialize (janela)
  @janela = janela
  @imagem = Gosu::Image.new(@janela,"Nave.bmp",true)
  @x = @janela.width/2
  @y = @janela.height/2
 end
 def draw
  @imagem.draw(@x,@y,1)
 end
 def mover_direita
  @x = @x + 5
  if (@x>@janela.width-(@imagem.width+10) ) then
   @x = @janela.width-(@imagem.width+10)
  end
 end
 def mover_esquerda
  @x = @x - 5
  if (@x < 10) then @x = 10 end
 end
 ...
end
```

Cata_estrela.rb

```
$LOAD_PATH << '.'

require 'gosu'
require 'jogador'
class CataEstrela < Gosu::Window
 def initialize
  super(640,480,false)
  self.caption = "Cata Estrelas"
  @imagem_fundo = Gosu::Image.new(self, "Space.png", true)
  @jogador = Jogador.new(self)
 end
  def draw
    @imagem_fundo.draw(0,0,0)
    @jogador.draw()
  end
  def update
    if ( button_down?(Gosu::Button::KbRight) ) then
      @jogador.mover_direita
    end
    if ( button_down?(Gosu::Button::KbLeft) ) then
      @jogador.mover_esquerda
    end
   ...
  end
end
```



Organizando

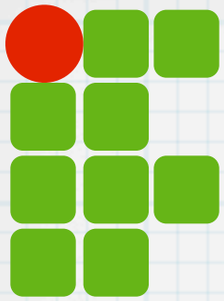
* Colocar o jogo e o jogador em classes separadas

jogador.rb

```
require 'gosu'
class Jogador
 def initialize (janela)
  @janela = janela
  @imagem = Gosu::Image.new(@janela,"Nave.bmp",true)
  @x = @janela.width/2
  @y = @janela.height/2
 end
 def draw
  @imagem.draw(@x,@y,1)
 end
 def mover_direita
  @x = @x + 5
  if (@x>@janela.width-(@imagem.width+10) ) then
   @x = @janela.width-(@imagem.width+10)
  end
 end
 def mover_esquerda
  @x = @x - 5
  if (@x < 10) then @x = 10 end
 end
 ...
end
```

Cata_estrela.rb

```
$LOAD_PATH << '.'

require 'gosu'
require 'jogador'
class CataEstrela < Gosu::Window
 def initialize
  super(640,480,false)
  self.caption = "Cata Estrelas"
  @imagem_fundo = Gosu::Image.new(self, "Space.png", true)
  @jogador = Jogador.new(self)
 end
  def draw
   @imagem_fundo.draw(0,0,0)
   @jogador.draw()
  end
  def update
    if ( button_down?(Gosu::Button::KbRight) ) then
      @jogador.mover_direita
    end
    if ( button_down?(Gosu::Button::KbLeft) ) then
      @jogador.mover_esquerda
    end
   ...
  end
end
```



Organizando

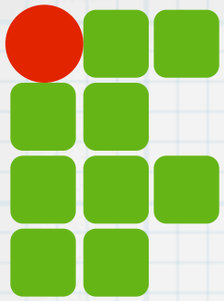
* Colocar o jogo e o jogador em classes separadas

jogador.rb

```
require 'gosu'
class Jogador
 def initialize (janela)
  @janela = janela
  @imagem = Gosu::Image.new(@janela,"Nave.bmp",true)
  @x = @janela.width/2
  @y = @janela.height/2
 end
 def draw
  @imagem.draw(@x,@y,1)
 end
 def mover_direita
  @x = @x + 5
  if (@x>@janela.width-(@imagem.width+10) ) then
   @x = @janela.width-(@imagem.width+10)
  end
 end
 def mover_esquerda
  @x = @x - 5
  if (@x < 10) then @x = 10 end
 end
 ...
end
```

Cata_estrela.rb

```
$LOAD_PATH << '.'

require 'gosu'
require 'jogador'
class CataEstrela < Gosu::Window
 def initialize
  super(640,480,false)
  self.caption = "Cata Estrelas"
  @imagem_fundo = Gosu::Image.new(self, "Space.png", true)
  @jogador = Jogador.new(self)
 end
  def draw
   @imagem_fundo.draw(0,0,0)
   @jogador.draw()
  end
  def update
    if ( button_down?(Gosu::Button::KbRight) ) then
     @jogador.mover_direita
    end
    if ( button_down?(Gosu::Button::KbLeft) ) then
      @jogador.mover_esquerda
    end
   ...
  end
end
```



Organizando

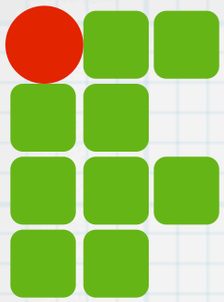
* Colocar o jogo e o jogador em classes separadas

jogador.rb

```
require 'gosu'
class Jogador
 def initialize (janela)
  @janela = janela
  @imagem = Gosu::Image.new(@janela,"Nave.bmp",true)
  @x = @janela.width/2
  @y = @janela.height/2
 end
 def draw
  @imagem.draw(@x,@y,1)
 end
 def mover_direita
  @x = @x + 5
  if (@x>@janela.width-(@imagem.width+10) ) then
   @x = @janela.width-(@imagem.width+10)
  end
 end
 def mover_esquerda
  @x = @x - 5
  if (@x < 10) then @x = 10 end
 end
 ...
end
```

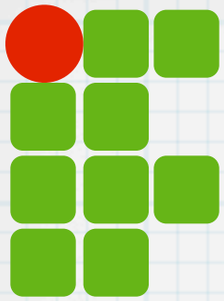
Cata_estrela.rb

```
$LOAD_PATH << '.'

require 'gosu'
require 'jogador'
class CataEstrela < Gosu::Window
 def initialize
  super(640,480,false)
  self.caption = "Cata Estrelas"
  @imagem_fundo = Gosu::Image.new(self, "Space.png", true)
  @jogador = Jogador.new(self)
 end
  def draw
   @imagem_fundo.draw(0,0,0)
   @jogador.draw()
  end
  def update
    if ( button_down?(Gosu::Button::KbRight) ) then
     @jogador.mover_direita
    end
    if ( button_down?(Gosu::Button::KbLeft) ) then
     @jogador.mover_esquerda
    end
    ...
  end
end
```



Organizando

* Cada objeto tem suas próprias responsabilidades
  * Características
  * Ações
* Objetos precisam conhecer outros objetos
  * Onde o jogador deve ser desenhado?
    * Na janela
  * O que deve acontecer quando apertar seta?
    * Mudar o jogador de posição



Arquivos

jogador.rb

```
require 'gosu'

class Jogador
  ...
end
```

principal.rb

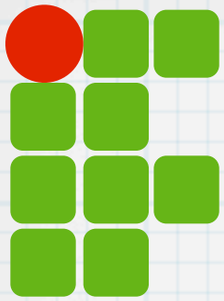
```
$LOAD_PATH << '.'

require 'CataEstrela'

jogo = CataEstrela.new
jogo.show
```

cata_estrela.rb

```
$LOAD_PATH << '.'
require 'gosu'
require 'jogador'

class CataEstrela < Gosu::Window
  ...
end
```

DEMO

v2

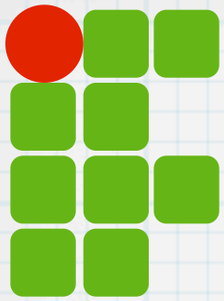


Mais Elementos

* Estrelas

estrela.rb

```
class Estrela

 attr_reader :x, :y

 def initialize(janela)
  @janela = janela
  @color = Gosu::Color.new(0xff000000)
  @color.red = rand(256 - 40) + 40
  @color.green = rand(256 - 40) + 40
  @color.blue = rand(256 - 40) + 40
  @x = rand * 640
  @y = rand * 480
  @imagem = Gosu::Image.new(@janela,"Estrela.png",true)
 end

 def draw
  @imagem.draw(@x - @imagem.width / 2.0, @y - @imagem.height / 2.0, 1, 1, 1, @color, :add)
 end
end
```



Mais Elementos

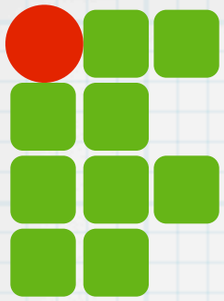
* Estrelas

criar métodos para leitura dos atributos:

```
def x
  return @x
end
```

estrela.rb

```
class Estrela

 attr_reader :x, :y

 def initialize(janela)
  @janela = janela
  @color = Gosu::Color.new(0xff000000)
  @color.red = rand(256 - 40) + 40
  @color.green = rand(256 - 40) + 40
  @color.blue = rand(256 - 40) + 40
  @x = rand * 640
  @y = rand * 480
  @imagem = Gosu::Image.new(@janela,"Estrela.png",true)
 end

 def draw
  @imagem.draw(@x - @imagem.width / 2.0, @y - @imagem.height / 2.0, 1, 1, 1, @color, :add)
 end
end
```



Mais Elementos

* Estrelas

criar métodos para leitura dos atributos:

```
def x
  return @x
end
```

estrela.rb

```
class Estrela

 attr_reader :x, :y

 def initialize(janela)
  @janela = janela
  @color = Gosu::Color.new(0xff000000)
  @color.red = rand(256 - 40) + 40
  @color.green = rand(256 - 40) + 40
  @color.blue = rand(256 - 40) + 40
  @x = rand * 640
  @y = rand * 480
  @imagem = Gosu::Image.new(@janela,"Estrela.png",true)
 end

 def draw
  @imagem.draw(@x - @imagem.width / 2.0, @y - @imagem.height / 2.0, 1, 1, 1, @color, :add)
 end
end
```

Arquivo de imagem Estrela.png

Mais Elementos

* Estrelas

criar métodos para leitura dos atributos:

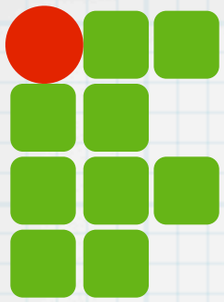
```
def x
  return @x
end
```

estrela.rb

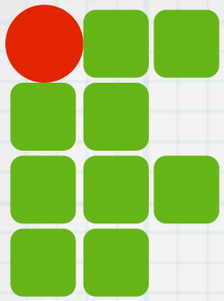
```
class Estrela

 attr_reader :x, :y

 def initialize(janela)
  @janela = janela
  @color = Gosu::Color.new(0xff000000)
  @color.red = rand(256 - 40) + 40
  @color.green = rand(256 - 40) + 40
  @color.blue = rand(256 - 40) + 40
  @x = rand * 640
  @y = rand * 480
  @imagem = Gosu::Image.new(@janela,"Estrela.png",true)
 end

 def draw
  @imagem.draw(@x - @imagem.width / 2.0, @y - @imagem.height / 2.0, 1, 1, 1, @color, :add)
 end
end
```

Criar cor aleatoreamente (RGB)

Arquivo de imagem Estrela.png



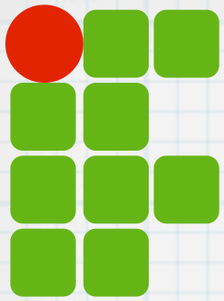
Criando estrelas



Criando estrelas

cata_estrela.rb

```
...
require 'estrela'

class CataEstrela < Gosu::Window
  def initialize
    ...
    @estrelas = []
    for i in 1..25 do
      @estrelas << Estrela.new(self)
    end
  end

  def draw
    ...
    for estrela in @estrelas do
      estrela.draw
    end
  end

  def update
    ...
  end
end
```



Criando estrelas

Array para armazenar as estrelas

cata_estrela.rb

```
...
require 'estrela'

class CataEstrela < Gosu::Window
  def initialize
    @estrelas = []
    for i in 1..25 do
      @estrelas << Estrela.new(self)
    end
  end

  def draw
    ...
    for estrela in @estrelas do
      estrela.draw
    end
  end

  def update
    ...
  end
end
```

Criando estrelas

Array para armazenar as estrelas

Laço para criar 25 estrelas

cata_estrela.rb

```
...
require 'estrela'

class CataEstrela < Gosu::Window
  def initialize
    @estrelas = []
    for i in 1..25 do
      @estrelas << Estrela.new(self)
    end
  end

  def draw
    ...
    for estrela in @estrelas do
      estrela.draw
    end
  end

  def update
    ...
  end
end
```

Criando estrelas

Array para armazenar as estrelas

Laço para criar 25 estrelas

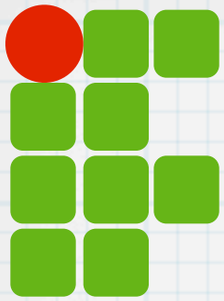
No método draw deve desenhar todas as estrelas contidas no array estrelas

cata_estrela.rb

```
...
require 'estrela'

class CataEstrela < Gosu::Window
  def initialize
    @estrelas = []
    for i in 1..25 do
      @estrelas << Estrela.new(self)
    end
  end

  def draw
    ...
    for estrela in @estrelas do
      estrela.draw
    end
  end

  def update
    ...
  end
end
```

DEMO

v3



Nave cata estrelas

* Quando passar por uma estrela a nave deve capturá-la

cata_estrela.rb

```
class CataEstrela < Gosu::Window
 def initialize
  ...
  @estrelas = Array.new
  for i in 1..25 do
   @estrelas << Estrela.new(self)
  end
 end

 def draw
  ...
 end

 def update
  ...
  if (rand(100) < 10 and @estrelas.size < 25) then
   @estrelas.push(Estrela.new(self))
  end
  @jogador.cata_estrelas(@estrelas)
 end
end
```

Nave cata estrelas

* Quando passar por uma estrela a nave deve capturá-la

O jogo começa sem estrelas

cata_estrela.rb

```
class CataEstrela < Gosu::Window
 def initialize
  ...
  @estrelas = Array.new
  for i in 1..25 do
   @estrelas << Estrela.new(self)
  end
 end

 def draw
  ...
 end

 def update
  ...
  if (rand(100) < 10 and @estrelas.size < 25) then
   @estrelas.push(Estrela.new(self))
  end
  @jogador.cata_estrelas(@estrelas)
 end
end
```

Nave cata estrelas

* Quando passar por uma estrela a nave deve capturá-la

O jogo começa sem estrelas

Novas estrelas são criadas no decorrer do jogo

cata_estrela.rb

```
class CataEstrela < Gosu::Window
 def initialize
  ...
  @estrelas = Array.new
  for i in 1..25 do
   @estrelas << Estrela.new(self)
  end
 end

 def draw
  ...
 end

 def update
  ...
  if (rand(100) < 10 and @estrelas.size < 25) then
   @estrelas.push(Estrela.new(self))
  end
  @jogador.cata_estrelas(@estrelas)
 end
end
```

Nave cata estrelas

* Quando passar por uma estrela a nave deve capturá-la

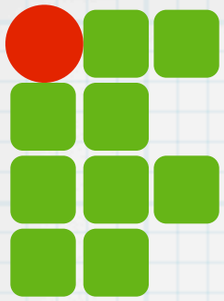
O jogo começa sem estrelas

Novas estrelas são criadas no decorrer do jogo

O jogador deve verificar se está "sobre" uma ou mais estrelas

cata_estrela.rb

```
class CataEstrela < Gosu::Window
 def initialize
  ...
  @estrelas = Array.new
  for i in 1..25 do
   @estrelas << Estrela.new(self)
  end
 end

 def draw
  ...
 end

 def update
  ...
  if (rand(100) < 10 and @estrelas.size < 25) then
   @estrelas.push(Estrela.new(self))
  end
  @jogador.cata_estrelas(@estrelas)
 end
end
```



Nave cata estrelas

* Método no jogador que verifica posição da nave e de cada estrela.

jogador.rb

```
class Jogador
 def initialize (janela)
 def draw
 def mover_direita
 def mover_esquerda
 def mover_cima
 def mover_baixo
 def cata_estrelas(estrelas)
  estrelas.reject! do |estrela|
     Gosu::distance(@x, @y, estrela.x, estrela.y) < 35
   end
 end
end
```

Nave cata estrelas

* Método no jogador que verifica posição da nave e de cada estrela.

jogador.rb

```
class Jogador
 def initialize (janela)
 def draw
 def mover_direita
 def mover_esquerda
 def mover_cima
 def mover_baixo
 def cata_estrelas(estrelas)
  estrelas.reject! do |estrela|
     Gosu::distance(@x, @y, estrela.x, estrela.y) < 35
   end
 end
end
```

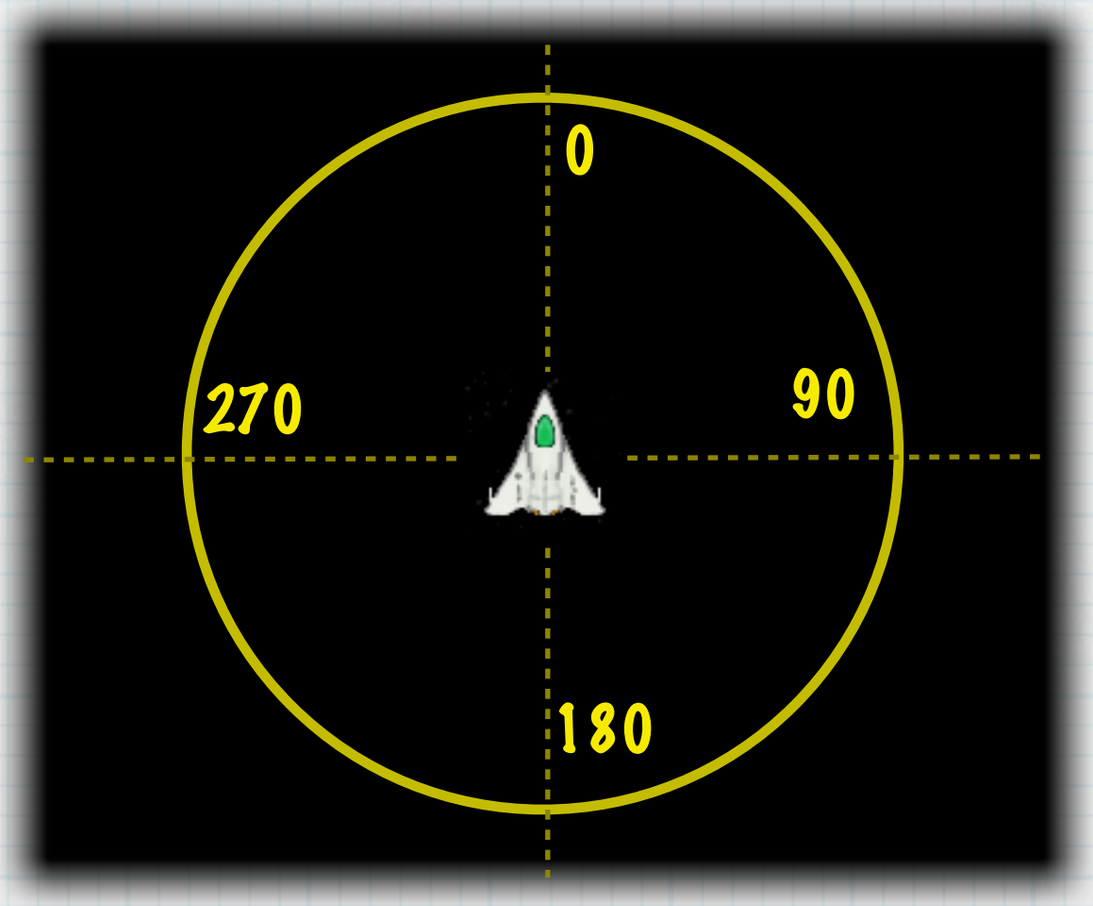
elimina do array as estrelas cuja distância da nave seja menor que 35.

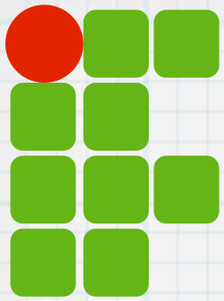
DEMO

v4

Movimento da nave

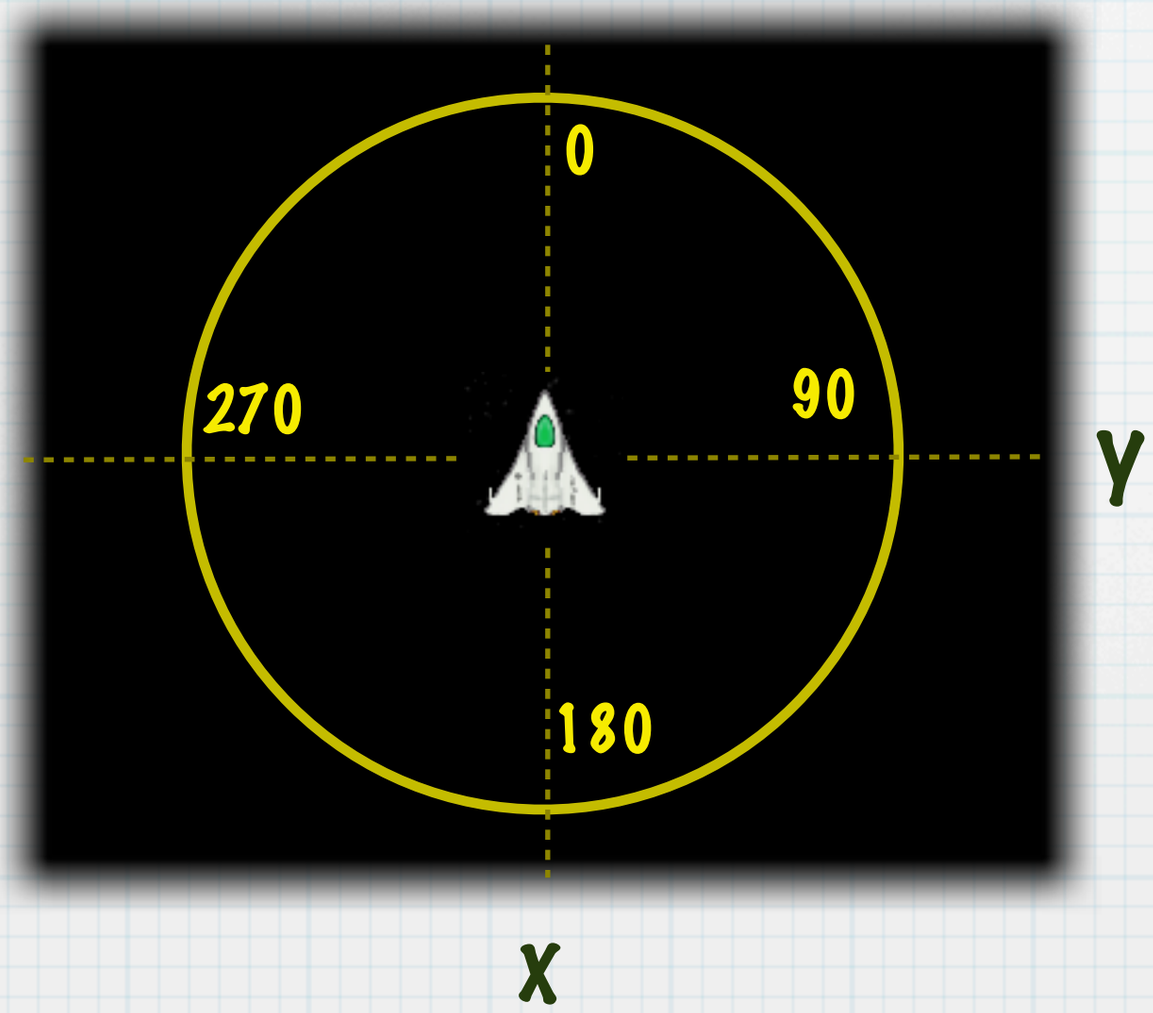
* Nave deve acelerar para frente
* Giro em torno do eixo

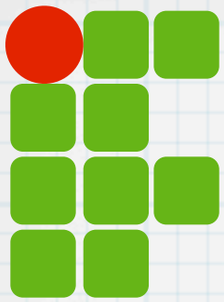




Desenho da nave

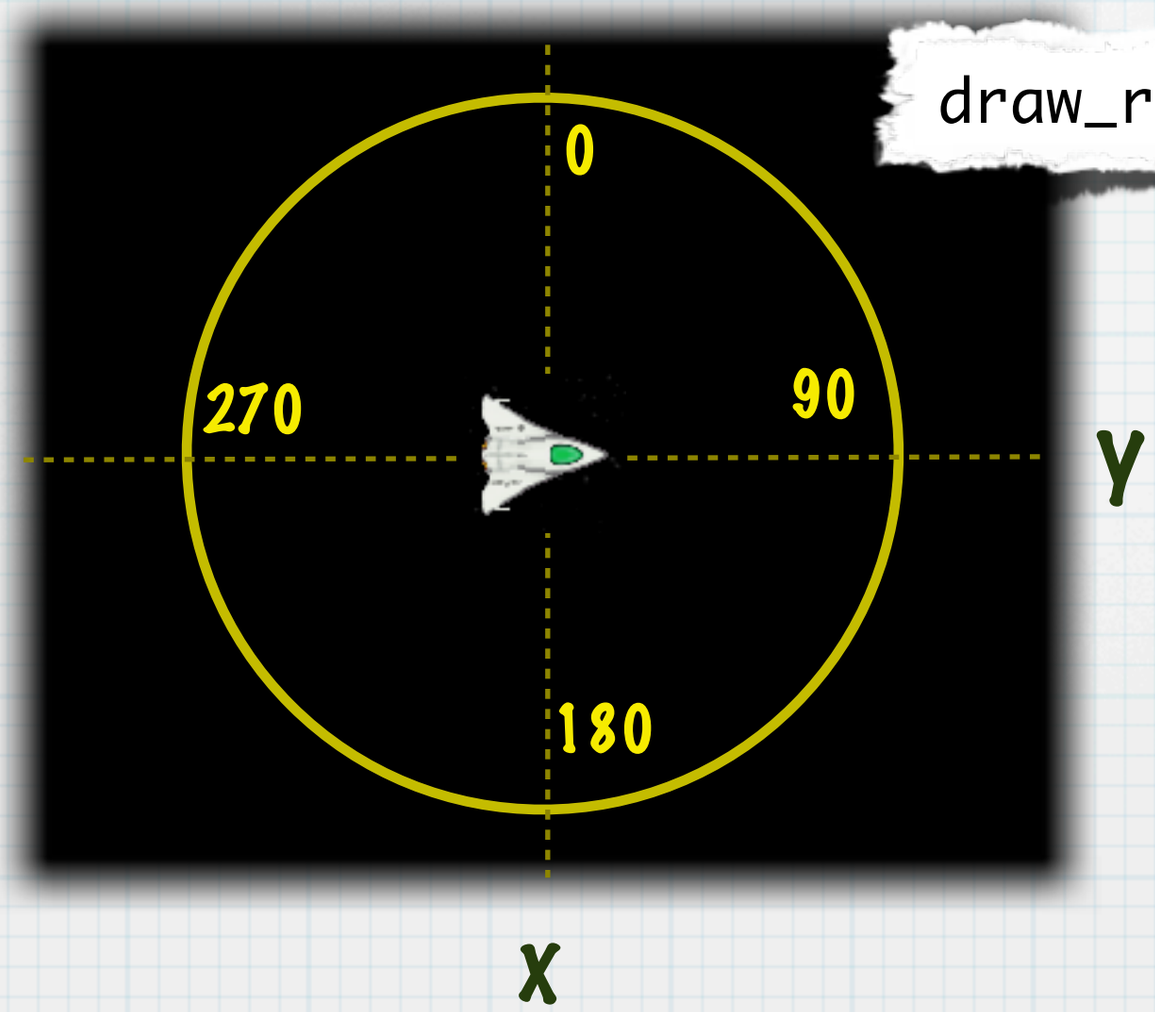
* draw_rot
  * @imagem.draw_rot(@x, @y, 2, @angulo)
    * x,y é no centro da imagem

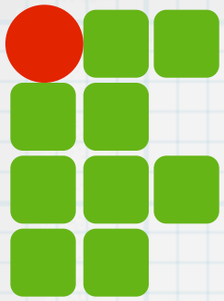




Desenho da nave

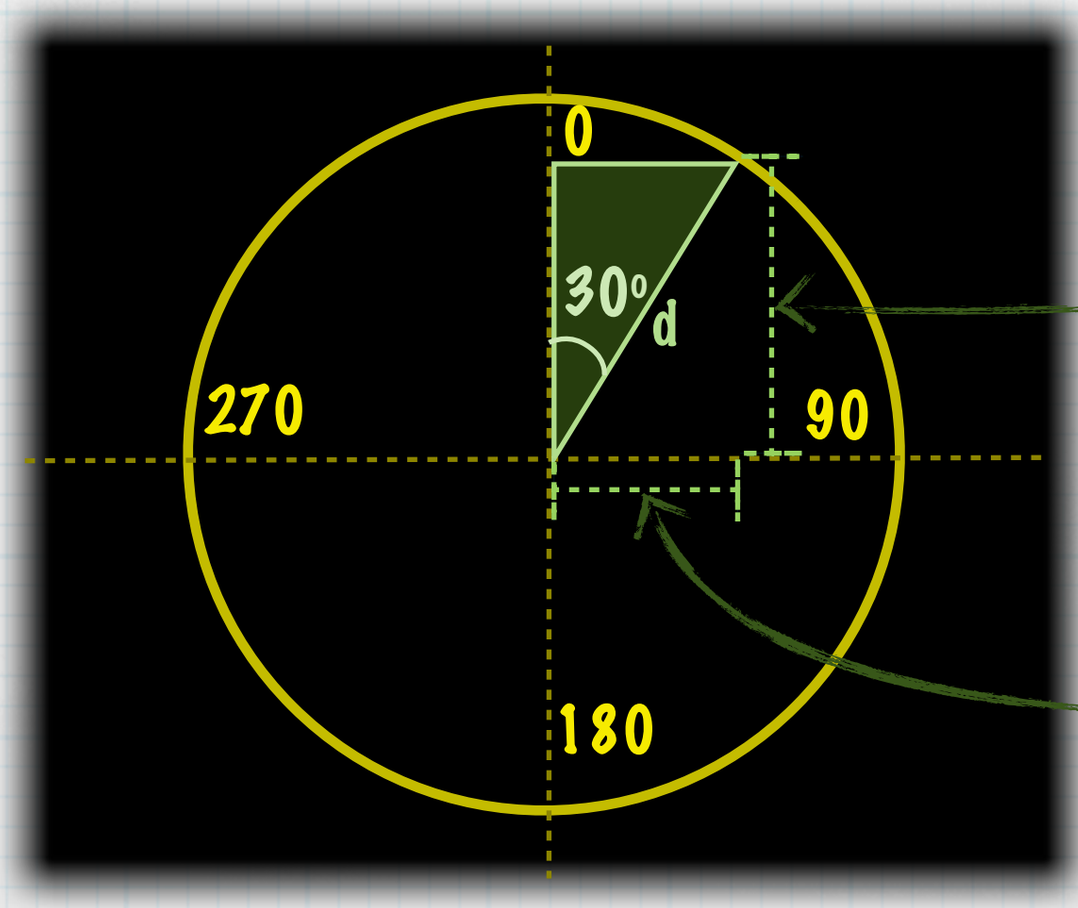
* draw_rot
  * @imagem.draw_rot(@x, @y, 2, @angulo)
    * x,y é no centro da imagem





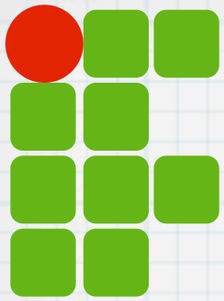
Movimento da nave

* Gosu::offset_x e Gosu::offset_y
  * x = Gosu::offset_x(30.0,d)
  * x = Gosu::offset_x(30.0,d)



offset x

offset y

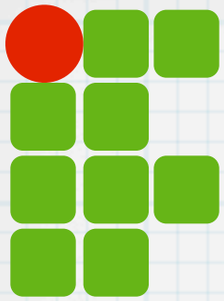


Movimento da nave

- Mudanças em jogador
  - método draw
  - Adicionar velocidades
    - do eixo x
    - do eixo y
  - substituir os métodos mover_??? pelos:
    - girar_esquerda
    - girar_direita
    - acelerar
    - mover
- Se atingir bordas do espaço muda para o lado inverso

jogador.rb

```
class Jogador
 def initialize (janela)
  ...
  @vel_x = 0
  @vel_y = 0
  @angulo = 0.0
 end
 def draw
  @imagem.draw_rot(@x, @y, 2, @angulo)
 end
 def girar_direita
  @angulo += 5.0
 end
 def girar_esquerda
  @angulo -= 5.0
 end
 def acelerar
  @vel_x += Gosu::offset_x(@angulo, 0.5)
  @vel_y += Gosu::offset_y(@angulo, 0.5)
 end
 def mover
  @x += @vel_x
  @y += @vel_y
  @x %= 640
  @y %= 480
  @vel_x *= 0.95
  @vel_y *= 0.95
 end
end
```



Movimento da nave

* Mudanças em CataEstrela
  * Apenas método update
    * Muda botões e chamadas aos métodos do jogador

cata_estrela.rb

```
class CataEstrela < Gosu::Window
 ...

 def update
  if ( button_down?(Gosu::Button::KbRight) ) then
   @jogador.girar_direita
  end
  if ( button_down?(Gosu::Button::KbLeft) ) then
   @jogador.girar_esquerda
  end
  if ( button_down?(Gosu::Button::KbUp) ) then
   @jogador.acelerar
  end
  if (rand(100) < 10 and @estrelas.size < 25) then
   @estrelas.push(Estrela.new(self))
  end
  @jogador.cata_estrelas(@estrelas)
  @jogador.mover
 end
end
```

DEMO

v5

Animação das estrelas

* Efeito 3D
  * Estrelas giram em torno do próprio eixo
  * Várias imagens
    * No arquivo uma imagem com as várias estrelas lado a lado

* Método load_tiles
  * `@imagens = Gosu::Image::load_tiles(@janela, "Estrela.png", 25, 25, false)`

Animação das estrelas

* Efeito 3D
  * Estrelas giram em torno do próprio eixo
  * Várias imagens
    * No arquivo uma imagem com as várias estrelas lado a lado

* Método load_tiles
  * `@imagens = Gosu::Image::load_tiles(@janela, "Estrela.png", 25, 25, false)`

Array com as imagens

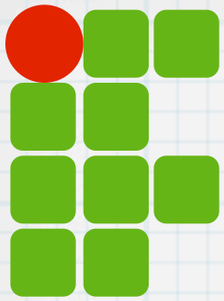
Animação das estrelas

* Efeito 3D
  * Estrelas giram em torno do próprio eixo
  * Várias imagens
    * No arquivo uma imagem com as várias estrelas lado a lado

* Método load_tiles
  * `@imagens = Gosu::Image::load_tiles(@janela, "Estrela.png", 25, 25, false)`

Array com as imagens

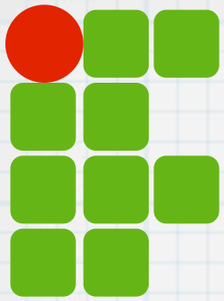
A imagem é dividida em blocos de 25x25 pixels



Animação das estrelas

estrela.rb

```
class Estrela
 attr_reader :x, :y
 def initialize(janela)
  @janela = janela
  @color = Gosu::Color.new(0xff000000)
  @color.red = rand(256 - 40) + 40
  @color.green = rand(256 - 40) + 40
  @color.blue = rand(256 - 40) + 40
  @x = rand * 640
  @y = rand * 480
  @imagens = Gosu::Image::load_tiles(@janela, "Estrela.png", 25, 25, false)
 end

 def draw
  imagem= @imagens[Gosu::milliseconds / 100 % @imagens.size]
  imagem.draw(@x - imagem.width / 2.0, @y - imagem.height / 2.0,1, 1, 1, @color, :add)
 end
end
```



Animação das estrelas

estrela.rb

```
class Estrela
 attr_reader :x, :y
 def initialize(janela)
  @janela = janela
  @color = Gosu::Color.new(0xff000000)
  @color.red = rand(256 - 40) + 40
  @color.green = rand(256 - 40) + 40
  @color.blue = rand(256 - 40) + 40
  @x = rand * 640
  @y = rand * 480
  @imagens = Gosu::Image::load_tiles(@janela, "Estrela.png", 25, 25, false)
 end

 def draw
  imagem= @imagens[Gosu::milliseconds / 100 % @imagens.size]
  imagem.draw(@x - imagem.width / 2.0, @y - imagem.height / 2.0,1, 1, 1, @color, :add)
 end
end
```

Define qual imagem mostrar

Animação das estrelas

estrela.rb

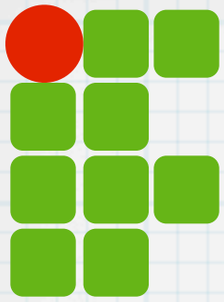
```
class Estrela
 attr_reader :x, :y
 def initialize(janela)
  @janela = janela
  @color = Gosu::Color.new(0xff000000)
  @color.red = rand(256 - 40) + 40
  @color.green = rand(256 - 40) + 40
  @color.blue = rand(256 - 40) + 40
  @x = rand * 640
  @y = rand * 480
  @imagens = Gosu::Image::load_tiles(@janela, "Estrela.png", 25, 25, false)
 end

 def draw
  imagem= @imagens[Gosu::milliseconds / 100 % @imagens.size]
  imagem.draw(@x - imagem.width / 2.0, @y - imagem.height / 2.0,1, 1, 1, @color, :add)
 end
end
```

Define qual imagem mostrar

Desenha imagem

DEMO

v6

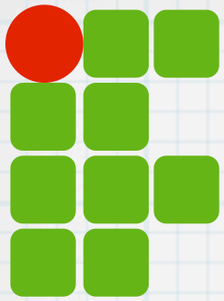


Som ao catar estrela

* Arquivo de som "Beep.wav"
* Carrega no initialize
* Toca ao capturar uma estrela

jogador.rb

```
class Jogador
 def initialize (janela)
  ...
  @beep = Gosu::Sample.new(@janela, "Beep.wav")
  ...
 end
 ...
 def cata_estrelas(estrelas)
  estrelas.reject! do |estrela|
   if Gosu::distance(@x, @y, estrela.x, estrela.y) < 35 then
    @beep.play
    true
   else
    false
   end
  end
 end
end
```

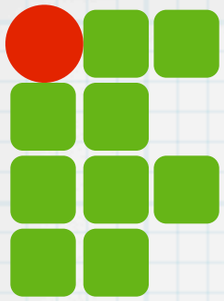


Som ao catar estrela

* Arquivo de som "Beep.wav"
* Carrega no initialize
* Toca ao capturar uma estrela

jogador.rb

```
class Jogador
 def initialize (janela)
  ...
  @beep = Gosu::Sample.new(@janela, "Beep.wav")
  ...
 end
 ...
 def cata_estrelas(estrelas)
  estrelas.reject! do |estrela|
   if Gosu::distance(@x, @y, estrela.x, estrela.y) < 35 then
    @beep.play
    true
   else
    false
   end
  end
 end
end
```

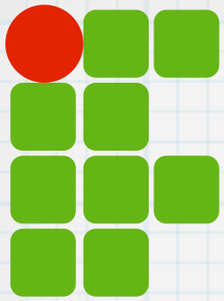


Placar

* 10 pontos por estrela capturada
  * Jogador conta seus pontos

estrela.rb

```
class Jogador
 attr_reader :placar
 def initialize (janela)
  ...
  @placar = 0
 end
 ...
 def cata_estrelas(estrelas)
  estrelas.reject! do |estrela|
   if Gosu::distance(@x, @y, estrela.x, estrela.y) < 35
then
    @beep.play
    @placar+=10
    true
   else
    false
   end
  end
 end
end
```

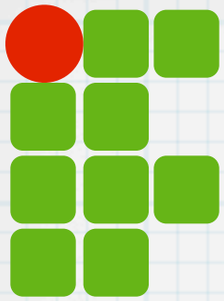


Placar

* 10 pontos por estrela capturada
  * Jogador conta seus pontos

estrela.rb

```
class Jogador
 attr_reader :placar
 def initialize (janela)
  ...
  @placar = 0
 end
 ...
 def cata_estrelas(estrelas)
  estrelas.reject! do |estrela|
   if Gosu::distance(@x, @y, estrela.x, estrela.y) < 35
then
    @beep.play
    @placar+=10
    true
   else
    false
   end
  end
 end
end
```

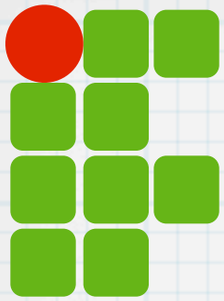


Placar

* 10 pontos por estrela capturada
  * Jogador conta seus pontos

estrela.rb

```
class Jogador
 attr_reader :placar
 def initialize (janela)
  ...
  @placar = 0
 end
 ...
 def cata_estrelas(estrelas)
  estrelas.reject! do |estrela|
   if Gosu::distance(@x, @y, estrela.x, estrela.y) < 35
then
    @beep.play
    @placar+=10
    true
   else
    false
   end
  end
 end
end
```

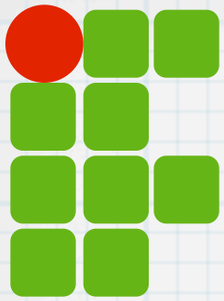


Placar

* 10 pontos por estrela capturada
  * Jogador conta seus pontos

estrela.rb

```
class Jogador
 attr_reader :placar
 def initialize (janela)

 @placar = 0
 end
 ...
 def cata_estrelas(estrelas)
  estrelas.reject! do |estrela|
   if Gosu::distance(@x, @y, estrela.x, estrela.y) < 35
then
    @beep.play
    @placar+=10
     true
    else
     false
    end
  end
 end
end
```

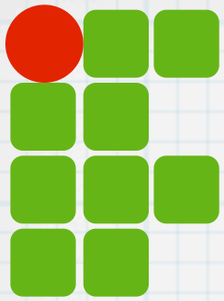


Mostrar placar

* CataEstrela deve mostrar os pontos

cata_estrela.rb

```
$LOAD_PATH << '.'

require 'gosu'
require 'jogador'
require 'estrela'

class CataEstrela < Gosu::Window
 def initialize
  ...
  @font = Gosu::Font.new(self, Gosu::default_font_name, 20)
 end

 def draw
  ...
  @font.draw("Placar: #{@jogador.placar}", 10, 10, 3, 1.0, 1.0, 0xffffff00)
 end
 ...
end
```



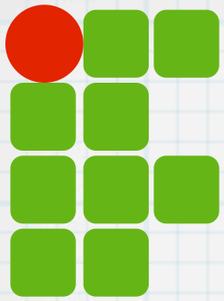
Mostrar placar

* CataEstrela deve mostrar os pontos

Cria o tipo da fonte

cata_estrela.rb

```
$LOAD_PATH << '.'

require 'gosu'
require 'jogador'
require 'estrela'

class CataEstrela < Gosu::Window
 def initialize
  ...
  @font = Gosu::Font.new(self, Gosu::default_font_name, 20)
 end

 def draw
  ...
  @font.draw("Placar: #{@jogador.placar}", 10, 10, 3, 1.0, 1.0, 0xffffff00)
 end
 ...
end
```



Mostrar placar

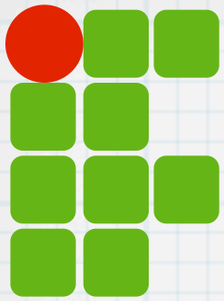
* CataEstrela deve mostrar os pontos

Cria o tipo da fonte

Desenha o texto nas cordenadas x=10 e y=10. Eixo z=3.

cata_estrela.rb

```
$LOAD_PATH << '.'

require 'gosu'
require 'jogador'
require 'estrela'

class CataEstrela < Gosu::Window
 def initialize
 ...
  @font = Gosu::Font.new(self, Gosu::default_font_name, 20)
 end

 def draw
 ...
  @font.draw("Placar: #{@jogador.placar}", 10, 10, 3, 1.0, 1.0, 0xffffff00)
 end
 ...
end
```



Mostrar placar

* CataEstrela deve mostrar os pontos

cata_estrela.rb

```
$LOAD_PATH << '.'

require 'gosu'
require 'jogador'
require 'estrela'

class CataEstrela < Gosu::Window
 def initialize
  ...
  @font = Gosu::Font.new(self, Gosu::default_font_name, 20)
 end

 def draw
  ...
  @font.draw("Placar: #{@jogador.placar}", 10, 10, 3, 1.0, 1.0, 0xffffff00)
 end
 ...
end
```

Cria o tipo da fonte

Desenha o texto nas cordenadas x=10 e y=10. Eixo z=3.

Fator de proporcionalidade de x e y

Mostrar placar

* CataEstrela deve mostrar os pontos

cata_estrela.rb

```
$LOAD_PATH << '.'

require 'gosu'
require 'jogador'
require 'estrela'

class CataEstrela < Gosu::Window
 def initialize
  ...
  @font = Gosu::Font.new(self, Gosu::default_font_name, 20)
 end

 def draw
  ...
  @font.draw("Placar: #{@jogador.placar}", 10, 10, 3, 1.0, 1.0, 0xffffff00)
 end
 ...
end
```

Cria o tipo da fonte

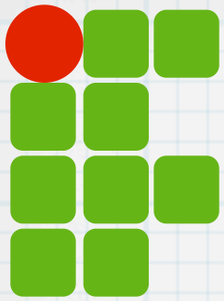
Desenha o texto nas cordenadas x=10 e y=10. Eixo z=3.

Fator de proporcionalidade de x e y

Cor do texto

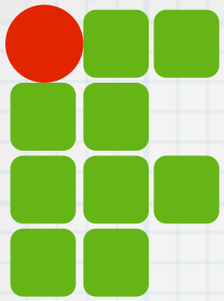
DEMO

v7



Jogo

* Tela inicial com menu
  * Tecla [i] para iniciar Jogo
* Temporizador para finalizar jogo
* Tela final com o placar
* Estado determina o que fazer

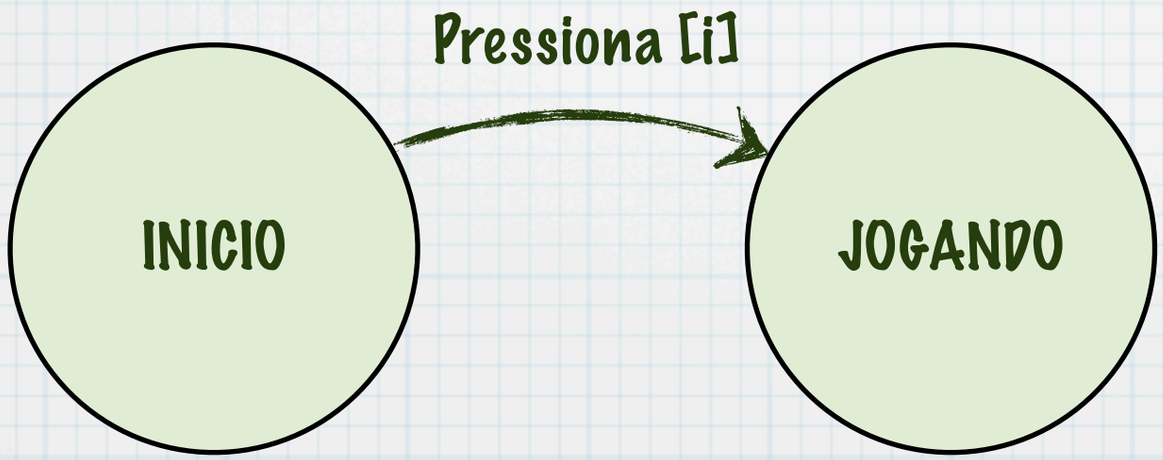


Jogo

* Tela inicial com menu
  * Tecla [i] para iniciar Jogo
* Temporizador para finalizar jogo
* Tela final com o placar
* Estado determina o que fazer

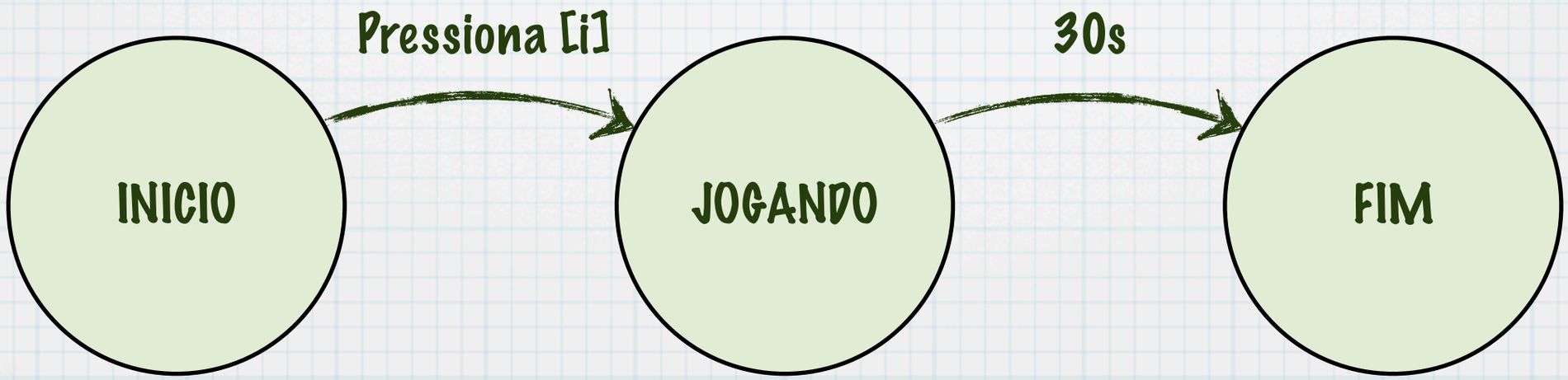
Jogo

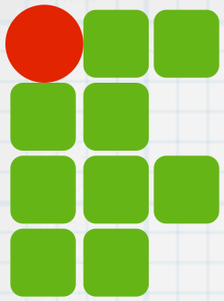
* Tela inicial com menu
  * Tecla [i] para iniciar Jogo
* Temporizador para finalizar jogo
* Tela final com o placar
* Estado determina o que fazer



Jogo

* Tela inicial com menu
  * Tecla [i] para iniciar Jogo
* Temporizador para finalizar jogo
* Tela final com o placar
* Estado determina o que fazer



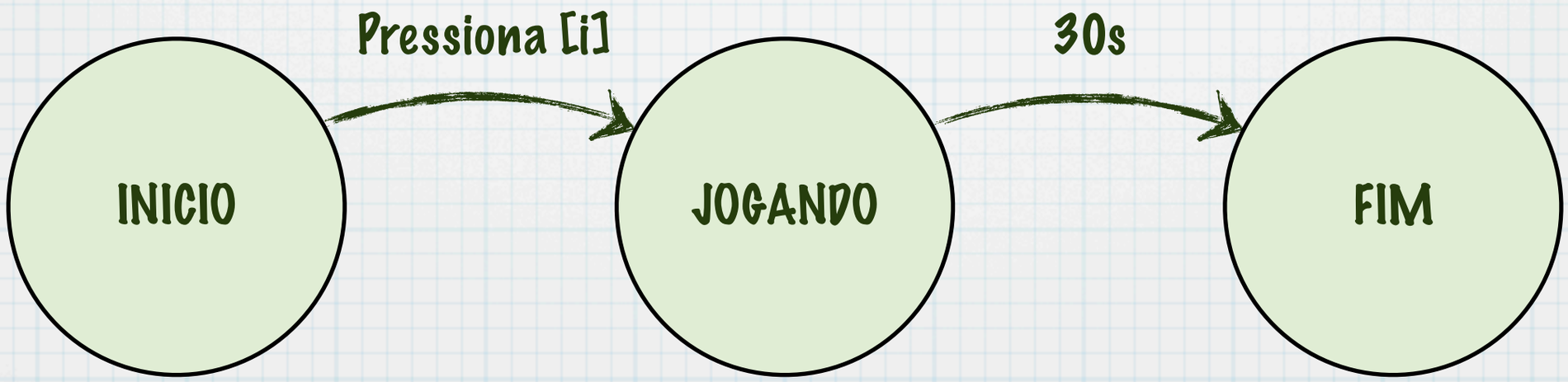


Jogo

* Tela inicial com menu
  * Tecla [i] para iniciar Jogo
* Temporizador para finalizar jogo
* Tela final com o placar
* Estado determina o que fazer

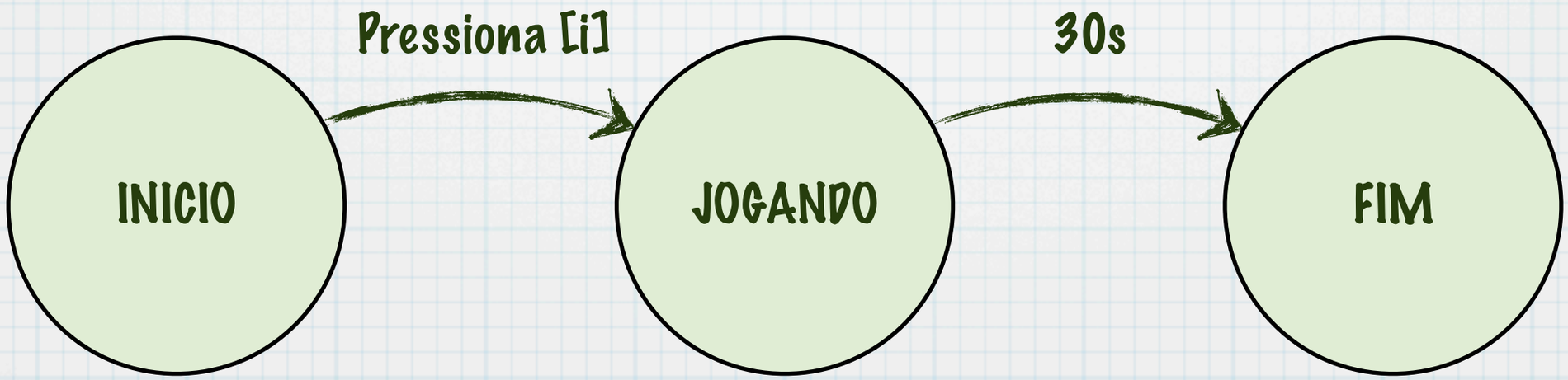
cata_estrela.rb

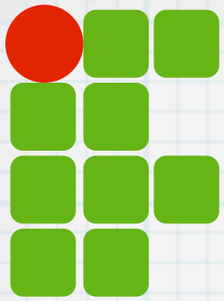
```
class CataEstrela < Gosu::Window
 def initialize
  ...
  @tempo = 0.0
  @estado = "INICIO"
 end
```



Jogo

* Métodos update e draw



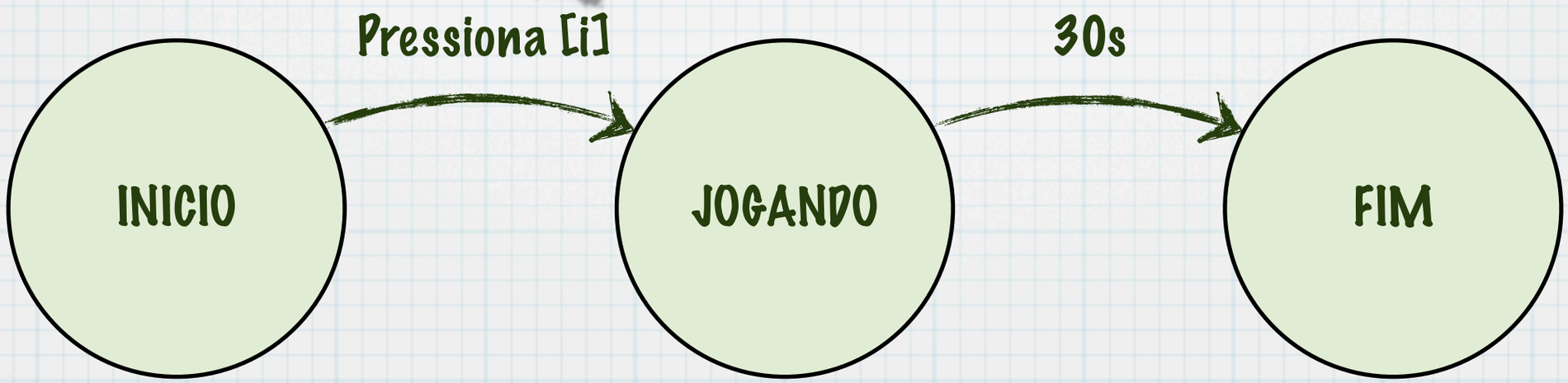


Jogo

* Métodos update e draw

cata_estrela.rb

```
def draw
  @imagem_fundo.draw(0,0,0)
  if (@estado == "INICIO") then
   ...
  elsif (@estado == "JOGANDO") then
   ...
  elsif(@estado == "FIM") then
   ...
  end
end
```



Jogo

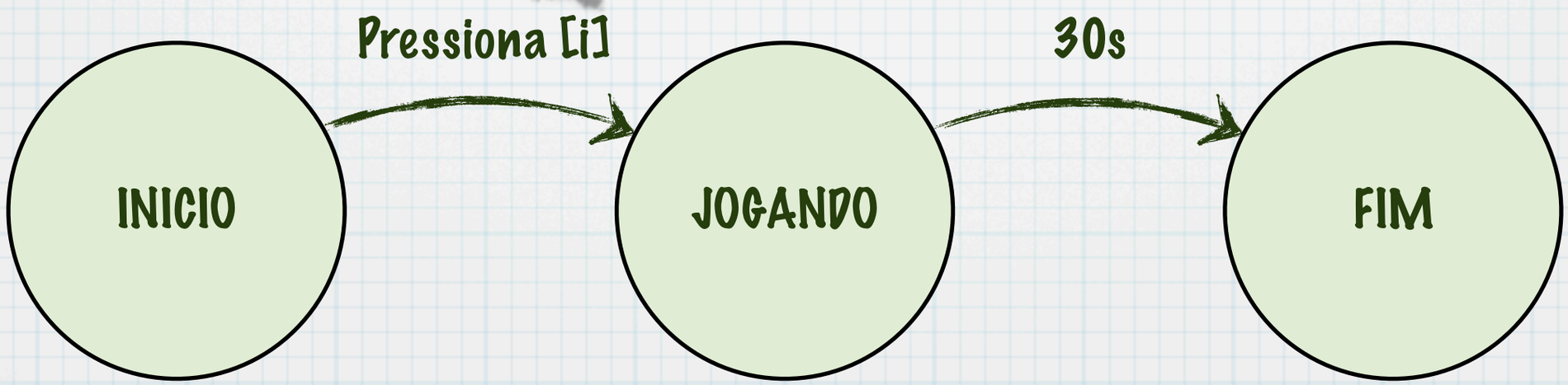
* Métodos update e draw

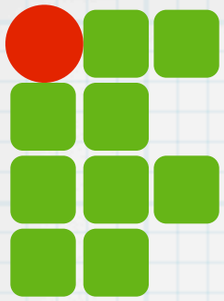
cata_estrela.rb

```
def draw
  @imagem_fundo.draw(0,0,0)
  if (@estado == "INICIO") then
   ...
  elsif (@estado == "JOGANDO") then
   ...
  elsif(@estado == "FIM") then
   ...
 end
end
```

cata_estrela.rb

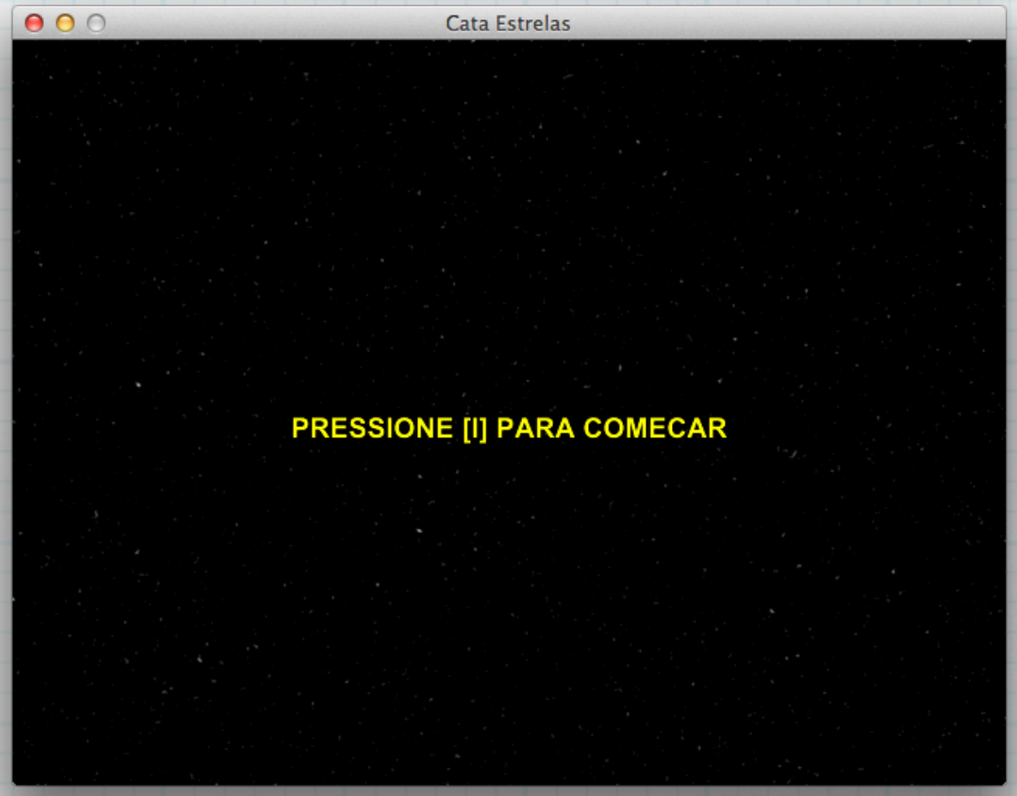
```
def update
 if (@estado == "INICIO" ) then
  ...
 elsif (@estado =="JOGANDO" ) then
  ...
 elsif (@estado == "FIM") then
 end
end
```





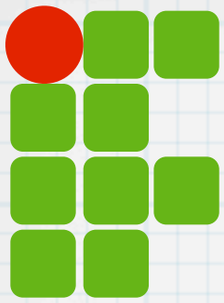
Menu inicial

* Tela inicial com mensagem
* Jogo começa
* O método draw desenha o menu



cata_estrela.rb

```
def draw
 @imagem_fundo.draw(0,0,0)
 if (@estado == "INICIO") then
  msg = "PRESSIONE [I] PARA COMECAR"
  x=(self.width)/2-((@fonte.text_width(msg,1)/2))
  @fonte.draw(msg, x, self.height/2, 3, 1.0, 1.0, 0xffffff00)
  ...
```

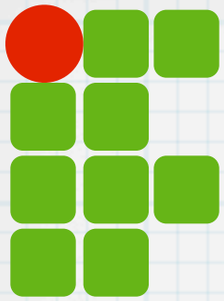


Menu inicial

* Jogo muda de estado quando usuário digita a tecla i

cata_estrela.rb

```
def update
 if (@estado == "INICIO" ) then
  if (button_down?(Gosu::Button::KbI)) then
   @estado = "JOGANDO"
  end
...
```



Jogando

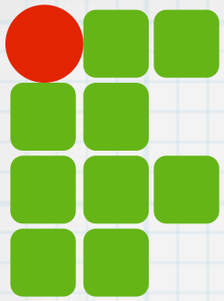
- Tempo de jogo
  - Variável para contar o tempo
    - Declarada em `initialize` com valor zero
    - Incrementada quando estiver no estado "JOGANDO"
    - Determina fim de jogo quando atinge tempo (30s
- Draw e update de acordo com o funcionamento do jogo descrito anteriormente

cata_estrela.rb

```
def draw
 ...
 elsif (@estado == "JOGANDO") then
  @jogador.draw()
  for estrela in @estrelas do
   estrela.draw()
  end
  @fonte.draw("Placar: #{@jogador.placar}", 10, 10, 3, 1.0, 1.0, 0xffffff00)
  @fonte.draw("Tempo: #{@tempo.to_i}s", 10, 30, 3, 1.0, 1.0, 0xffffff00)
 elsif(@estado == "FIM") then
  ...
```

cata_estrela.rb

```
def update
 ...
 elsif (@estado =="JOGANDO" ) then
  if ( button_down?(Gosu::Button::KbRight) ) then
   @jogador.girar_direita
  end
  if ( button_down?(Gosu::Button::KbLeft) ) then
   @jogador.girar_esquerda
  end
  if ( button_down?(Gosu::Button::KbUp) ) then
   @jogador.acelerar
  end
  if (rand(100) < 10 and @estrelas.size < 25) then
   @estrelas.push(Estrela.new(self))
  end
  @jogador.cata_estrelas(@estrelas)
  @jogador.mover
  @tempo+=1.0/60.0
  if (@tempo.to_i == 30) then
   @estado = "FIM"
  end
 ...
 end
end
```



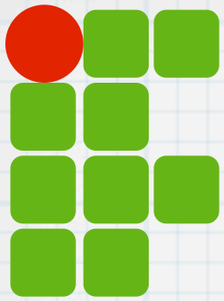
Fim de jogo

* Método draw mostra o placar
* O método update nada faz

```
class CataEstrela < Gosu::Window
  ...
 def draw
  ...
  elsif(@estado == "FIM") then
   msg = "FIM DE JOGO, VOCE FEZ #{@jogador.placar} PONTOS"
   x=(self.width)/2-((@fonte.text_width(msg,1)/2))
   @fonte.draw(msg, x, self.height/2, 3, 1.0, 1.0, 0xffffff00)
  end
 end
 def update
  ...
  elsif (@estado == "FIM") then
  end
 end
end
```

DEMO

V8



Conclusão

* Simplifica o desenvolvimento de jogos
* Bibliotecas extras especifica para jogos

* Tutorial GOSU para Ruby
  * https://github.com/jlnr/gosu/wiki/Ruby-Tutorial

* Exercício: Faça o tutorial abaixo
  * https://sites.google.com/a/ruby4kids.com/gosu/home

Dúvidas?