Tópicos Especiais em Linguagens de Programação

Aula 07 – Estruturas de Repetição e Imagens

Edirlei Soares de Lima
<edirlei@iprj.uerj.br>

Estruturas de Repetição

- Diversos problemas somente podem ser resolvidos numericamente por um computador se o resultado de pequenas computações forem acumulados.
  - **Exemplo:** calcular o fatorial de um número.

- Precisamos de mecanismos que nos permitam requisitar que um conjunto de instruções seja repetidamente executado, até que uma determinada condição seja alcançada.

- **Repetições são programadas através da construção de laços (ou ciclos).**

Estruturas de Repetição (`while`)

- **Estruturas de repetição** são utilizadas para indicar que um determinado conjunto de instruções deve ser executado um número definido ou indefinido de vezes, ou enquanto uma condição não for satisfeita.

- Em Lua, uma das formas de se trabalhar com repetições é através do comando **while**:

```
...
while expressão_lógica do
   -- Bloco de comandos
end
...
```

Enquanto a "**expressão_lógica**" for verdadeira, o "bloco de comandos" é executado.

Depois, a execução procede nos comandos subsequentes ao bloco while.

Estruturas de Repetição – Exemplo 1

- **Exemplo 1:**

"Crie um programa em Lua que escreva na tela todos os números entre 0 e 100"

```
local x = 0

while x <= 100 do
  io.write(x, "\n")
  x = x + 1
end
```

Estruturas de Repetição – Exemplo 2

- **Exemplo 2:**

"Fatorial de um número não-negativo"

$$n! = \prod_{i=1}^{n} i = n \times (n-1) \times (n-2) \times \cdots \times 3 \times 2 \times 1$$

```
function fatorial(n)
  local f = 1
  while n > 1 do
    f = f * n
    n = n - 1
  end
  return f
end
```

Estruturas de Repetição (`for`)

- Outra forma de se trabalhar com repetições é através do comando **for**:

```
...
for var = valor_inicial, valor_final, incremento do
   -- Bloco de comandos
end
...
```

- O bloco de comandos será executado para cada valor de `var` partindo de `valor_inicial` e indo até `valor_final`, usando o `incremento` para incrementar o valor de var

Estruturas de Repetição – Exemplo 1

- **Exemplo 1:** “Escrever na tela os números entre 0 e 100”

```
local x = 0

while x <= 100 do
  io.write(x, "\n")
  x = x + 1
end
```

```
for x = 0, 100, 1 do
  io.write(x, "\n")
end
```

- **Importante:**
  - Na estrutura **for** a variável de controle x é uma variável local;
  - Nunca modifique o valor da variável de controle dentro da estrutura **for**;

Estruturas de Repetição – Exemplo 2

- **Exemplo 2:**

"Fatorial de um número não-negativo"

$$n! = \prod_{i=1}^{n} i = n \times (n-1) \times (n-2) \times \cdots \times 3 \times 2 \times 1$$

```
function fatorial(n)
  local res = 1
  for f = n, 1, -1 do
    res = res * f
  end
  return res
end
```

Estruturas de Repetição (`repeat`)

- A estrutura **while** avalia a expressão booleana que controla a execução do bloco de comandos no **início do laço**.

- A linguagem Lua oferece uma terceira construção de laços através do comando **repeat**:
  - A expressão booleana é avaliada no final do laço.
  - Isso significa que o bloco de comandos é executado pelo menos uma vez.

```
...
repeat
   -- Bloco de comandos
until expressão_lógica
...
```

De Volta ao “Hello World”

- Na ultima implementação do “Hello World” fizemos o texto se mover na tela e retornar ao inicio quando ele atingir o limite da tela.

- E se nós precisássemos fazer o mesmo com 20 “Hello World” ao mesmo tempo?
  - Duplicar código nunca é uma opção viável!

- Com uma estrutura de repetição podemos fazer isso sem duplicar linhas de código.

- Como podemos fazer isso?

```
local px    -- posição x do texto

function love.load()
  love.graphics.setColor(0, 0, 0)
  love.graphics.setBackgroundColor(255, 255, 255)
  px = 0
end

function love.update(dt)
  px = px + (100 * dt)

  if px > love.graphics.getWidth() then
    px = 0
  end

end

function love.draw()
    love.graphics.print("Hello World", px, 300)
end
```

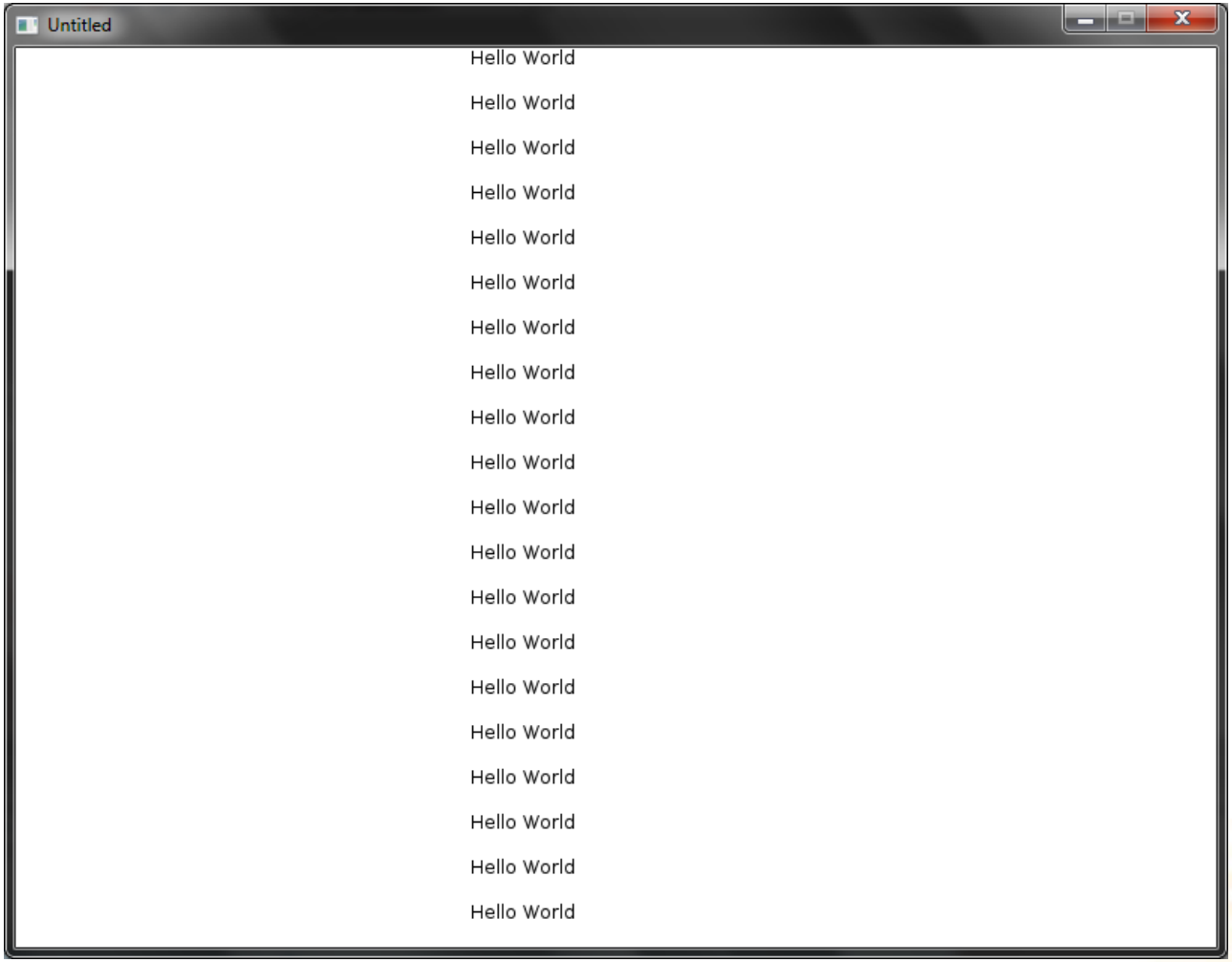
```
local px    -- posição x do texto

function love.load()
  love.graphics.setColor(0, 0, 0)
  love.graphics.setBackgroundColor(255, 255, 255)
  px = 0
end

function love.update(dt)
  px = px + (100 * dt)
  if px > love.graphics.getWidth() then
    px = 0
  end
end

function love.draw()
  for y = 0, 20, 1 do
    love.graphics.print("Hello World", px, y * 30)
  end
end
```

De Volta ao “Hello World”



Tipo `image`

- Jogos não são criados somente com formas geométricas básicas. Normalmente a arte do jogo é definida por um **conjunto de imagens**.

- O Löve oferece um **tipo de dados** especial para armazenar imagens chamado `image`.

- Podemos **carregar uma nova imagem** através do comando:

```
image = love.graphics.newImage(filename)
```

- Podemos **desenhar uma imagem** através do comando:

```
love.graphics.draw(drawable, x, y, r, sx, sy, ox, oy, kx, ky)
```

Tipo `image`

- Para desenhar uma imagem na tela é necessário **duas etapas**:
  - Carregar a imagem com o comando `love.graphics.newImage`
  - Desenhar a imagem com o comando `love.graphics.draw`

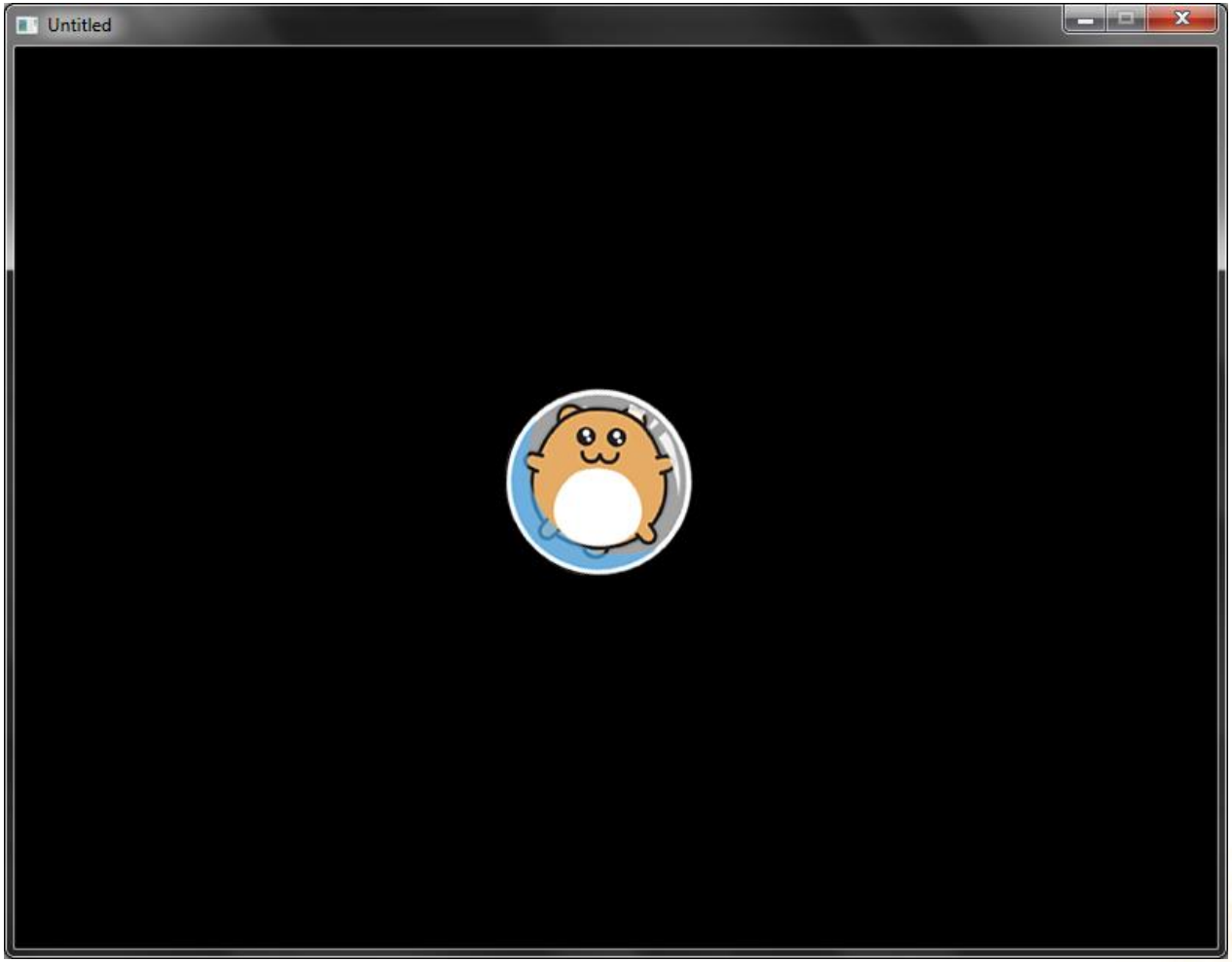
- **Exemplo:**

```
function love.load()
   hamster = love.graphics.newImage("hamster.png")
end

function love.draw()
   love.graphics.draw(hamster, 325, 225)
end
```

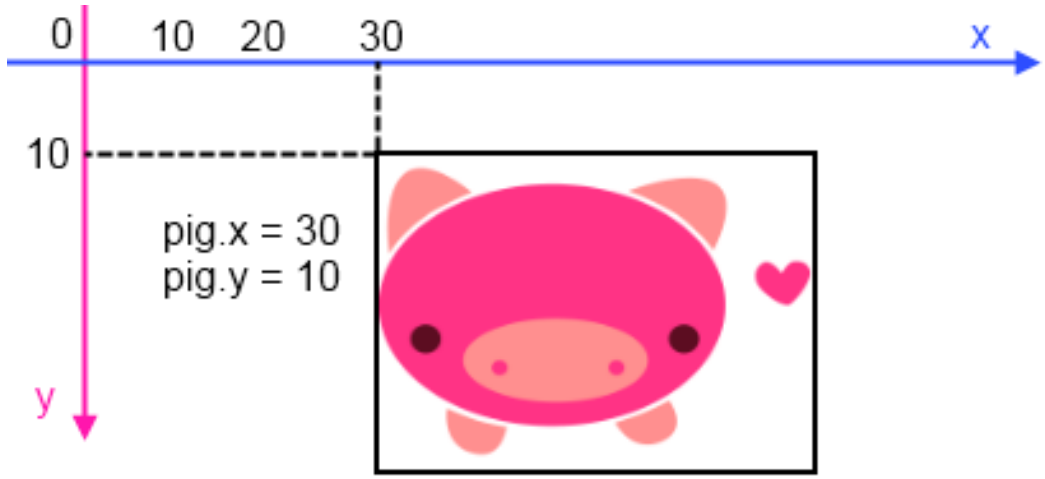
http://www.inf.puc-rio.br/~elima/intro-eng/hamster.png

Tipo `image`



Tipo `image`

- Por padrão, as imagens são desenhadas com o ponto de origem no canto superior esquerdo:



- É possível modificar o ponto de origem através dos outros parâmetros da função `love.graphics.draw`

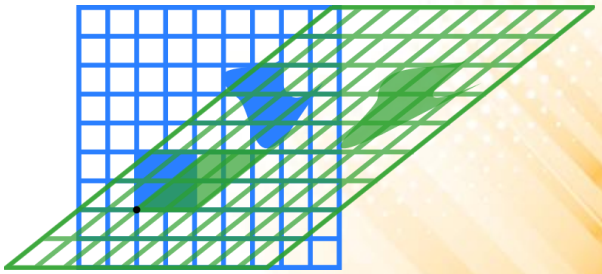
Tipo `image`

- Note que a função `love.graphics.draw` recebe vários outros parâmetros que podem ser utilizados:

```
love.graphics.draw(drawable, x, y, r, sx, sy, ox, oy, kx, ky)
```

  - **drawable**: imagem ou outros objetos que podem ser desenhados;
  - **x**: posição onde o objeto será desenhado (eixo x);
  - **y**: posição onde o objeto será desenhado (eixo y) ;
  - **r**: orientação do objeto (radiano);
  - **sx**: fator de escala do objeto (eixo x);
  - **sy**: fator de escala do objeto (eixo y);
  - **ox**: ponto de origem do objeto (eixo x);
  - **oy**: ponto de origem do objeto (eixo y);
  - **kx**: fator de distorção (eixo x);
  - **ky**: fator de distorção (eixo y);

fator de distorção:



Tipo `image`

- **Exemplo:**

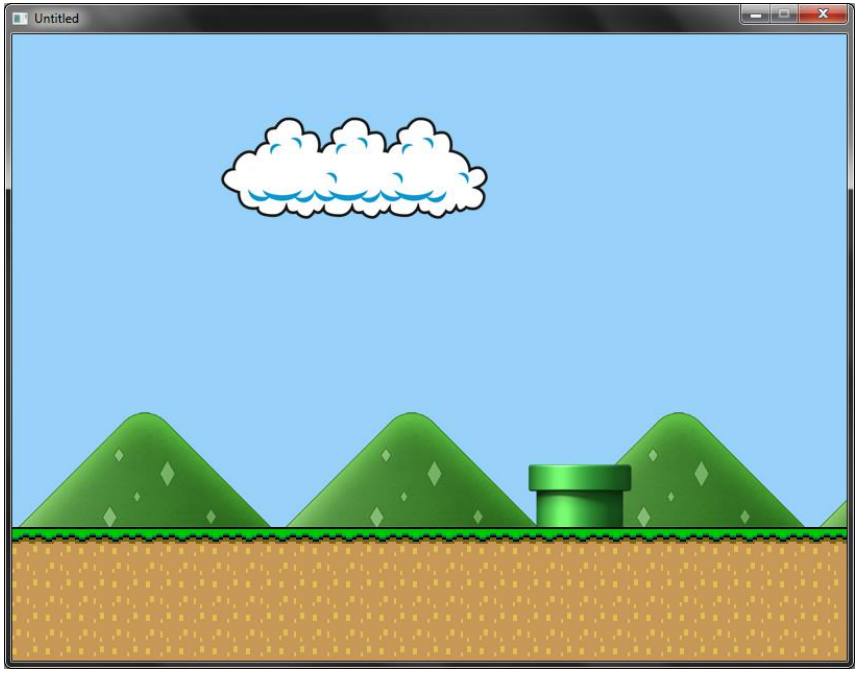
```
function love.load()
   hamster = love.graphics.newImage("hamster.png")
end

function love.draw()
   love.graphics.draw(hamster, 400, 300, math.rad(90), 1, 1,
                      hamster:getWidth()/2, hamster:getHeight()/2)
end
```

Note que é possível acessar a largura e altura da imagem através dos comandos `hamster:getWidth()` e `hamster:getHeight()`

Exercício 1

1) Faça um programa que desenhe na tela um cenário semelhante ao mostrado na imagem abaixo:



Importante: você deve utilizar estruturas de repetição para desenhar as imagens que se repetem mais de uma vez (chão e montanhas)

Imagens: http://www.inf.puc-rio.br/~elima/intro-eng/imagens_cenario.zip