



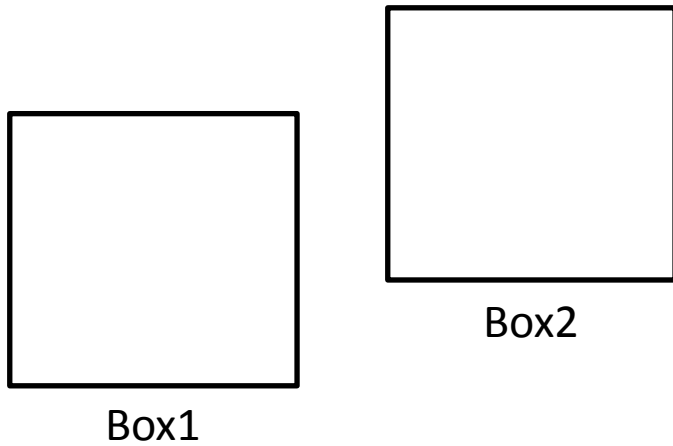
ENG1000 – Introdução à Engenharia

Aula 10 – Outras Funcionalidades

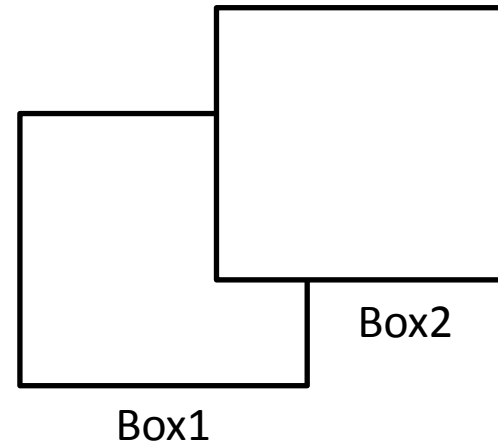
Edirlei Soares de Lima
<elima@inf.puc-rio.br>

Detecção de Colisão

Sem Colisão

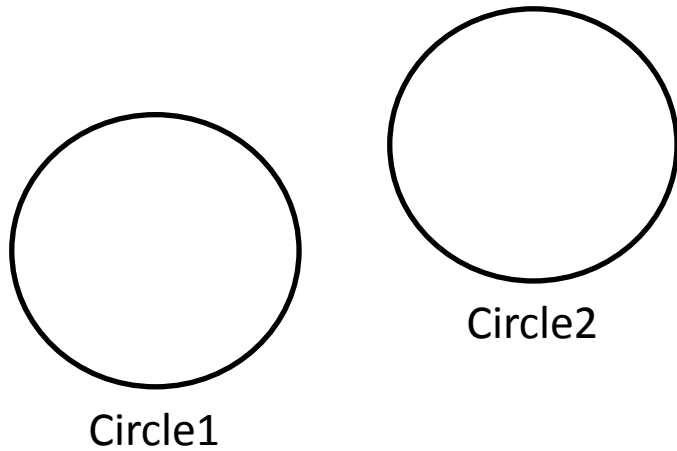


Com Colisão

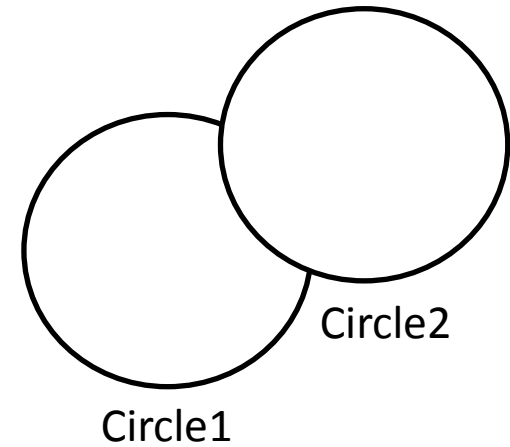


Detecção de Colisão

Sem Colisão



Com Colisão



Detecção de Colisão (Box)

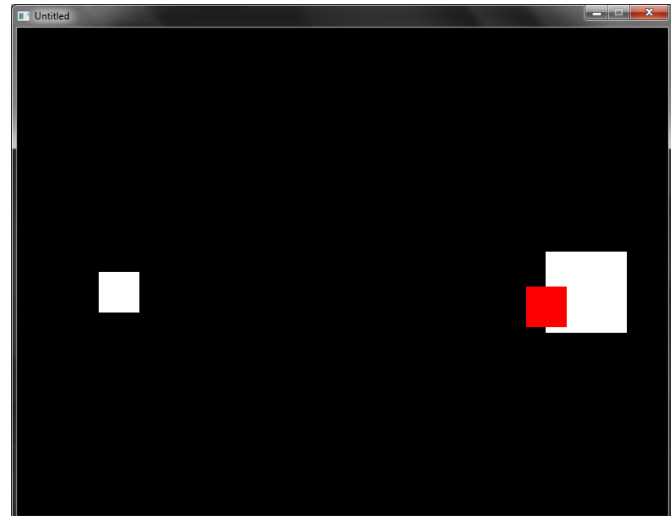
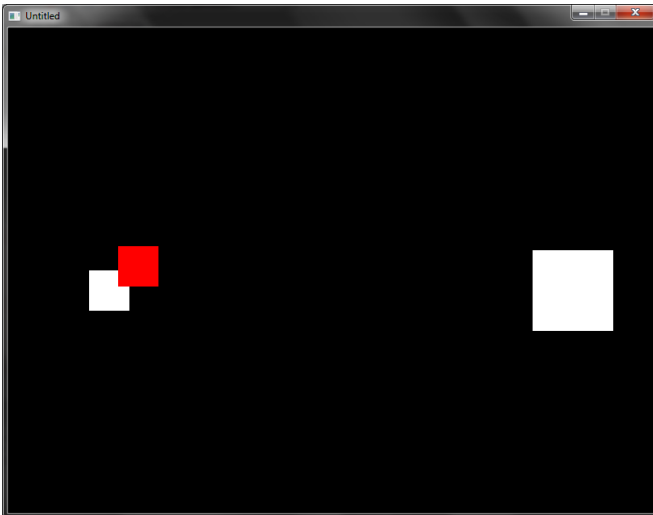
```
function love.load()
  player1 = {
    x = 390,
    y = 300,
    width = 50,
    height = 50,
    collided = false
  }
  box1 = {
    x = 100,
    y = 300,
    width = 50,
    height = 50
  }
  box2 = {
    x = 650,
    y = 275,
    width = 100,
    height = 100
  }
end
```

```
function CheckBoxCollision(x1,y1,w1,h1,x2,y2,w2,h2)
    return x1 < x2+w2 and x2 < x1+w1 and y1 < y2+h2 and y2 < y1+h1
end
```

```
function love.update(dt)
    if love.keyboard.isDown("left") then
        player1.x = player1.x - (120 * dt)
    end
    if love.keyboard.isDown("right") then
        player1.x = player1.x + (120 * dt)
    end
    if love.keyboard.isDown("up") then
        player1.y = player1.y - (120 * dt)
    end
    if love.keyboard.isDown("down") then
        player1.y = player1.y + (120 * dt)
    end

    if CheckBoxCollision(player1.x, player1.y, player1.width,
        player1.height, box1.x, box1.y, box1.width, box1.height) or
        CheckBoxCollision(player1.x, player1.y, player1.width,
        player1.height, box2.x, box2.y, box2.width, box2.height) then
        player1.collided = true
    else
        player1.collided = false
    end
end
```

```
function love.draw()  
  love.graphics.setColor(255,255,255)  
  love.graphics.rectangle("fill", box1.x, box1.y,  
                           box1.width, box1.height)  
  love.graphics.rectangle("fill", box2.x, box2.y,  
                           box2.width, box2.height)  
  
  if player1.collided == true then  
    love.graphics.setColor(255,0,0)  
  end  
  
  love.graphics.rectangle("fill", player1.x, player1.y,  
                           player1.width, player1.height)  
end
```



Detecção de Colisão (Circle)

```
function love.load()
  player1 = {
    x = 390,
    y = 300,
    raio = 25,
    collided = false
  }
  circle1 = {
    x = 100,
    y = 300,
    raio = 25
  }
  circle2 = {
    x = 650,
    y = 300,
    raio = 50
  }
end
```

```

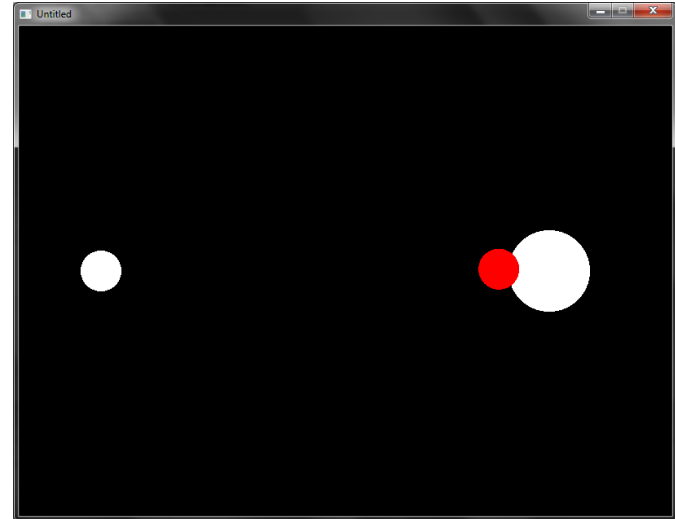
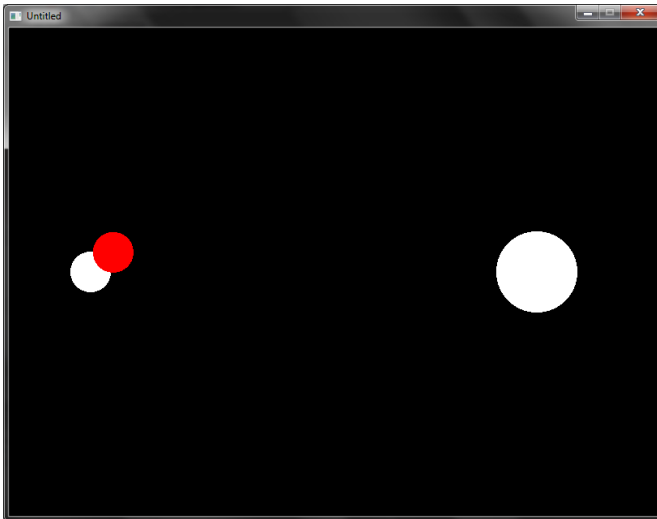
function CheckCircularCollision(ax, ay, ar, bx, by, br)
    local dx = bx - ax
    local dy = by - ay
    local dist = math.sqrt(dx * dx + dy * dy)
    return dist < ar + br
end

function love.update(dt)
    if love.keyboard.isDown("left") then
        player1.x = player1.x - (120 * dt)
    end
    if love.keyboard.isDown("right") then
        player1.x = player1.x + (120 * dt)
    end
    if love.keyboard.isDown("up") then
        player1.y = player1.y - (120 * dt)
    end
    if love.keyboard.isDown("down") then
        player1.y = player1.y + (120 * dt)
    end
    if CheckCircularCollision(player1.x, player1.y, player1.raio,
                             circle1.x, circle1.y, circle1.raio) or
        CheckCircularCollision(player1.x, player1.y, player1.raio,
                             circle2.x, circle2.y, circle2.raio) then
        player1.collided = true
    else
        player1.collided = false
    end
end
end

```



```
function love.draw()  
  love.graphics.setColor(255,255,255)  
  love.graphics.circle("fill", circle1.x, circle1.y, circle1.raio, 100)  
  love.graphics.circle("fill", circle2.x, circle2.y, circle2.raio, 100)  
  
  if player1.collided == true then  
    love.graphics.setColor(255,0,0)  
  end  
  
  love.graphics.circle("fill", player1.x, player1.y, player1.raio, 100)  
end
```



Geração de Números Aleatórios

```
number = love.math.random(min, max)
```

Exemplo 1:

```
x = love.math.random(1, 100) -- Gera um número entre 1 e 100
```

Exemplo 2:

```
vet = {} -- Gera um vetor com 100 números aleatórios  
for i=1, 100, 1 do  
    vet[i] = love.math.random(1, 100)  
end
```

Executando o Programa em Tela Cheia

```
success = love.window.setFullscreen(fullscreen, fstype)
```

Exemplo 1:

```
-- coloca o programa em tela cheia na resolução atual do windows  
love.window.setFullscreen(true, "desktop")
```

Exemplo 2:

```
-- coloca o programa em tela cheia mudando a resolução do monitor  
love.window.setFullscreen(true, "normal")
```

Modificando o Tamanho da Janela

```
success = love.window.setMode(width, height, flags)
```

Exemplo 1:

```
-- define o tamanho da janela em 1024 x 768 resizable  
love.window.setMode(1024, 768, {resizable=true})
```

Exemplo 2:

```
-- define o tamanho da janela em 1024 x 768 e em tela cheia  
love.window.setMode(1024, 768, {fullscreen=true})
```

Modificando o Titulo da Janela

```
love.window.setTitle(title)
```

Exemplo:

```
love.window.setTitle("Meu Jogo!")
```

Modificando a Fonte

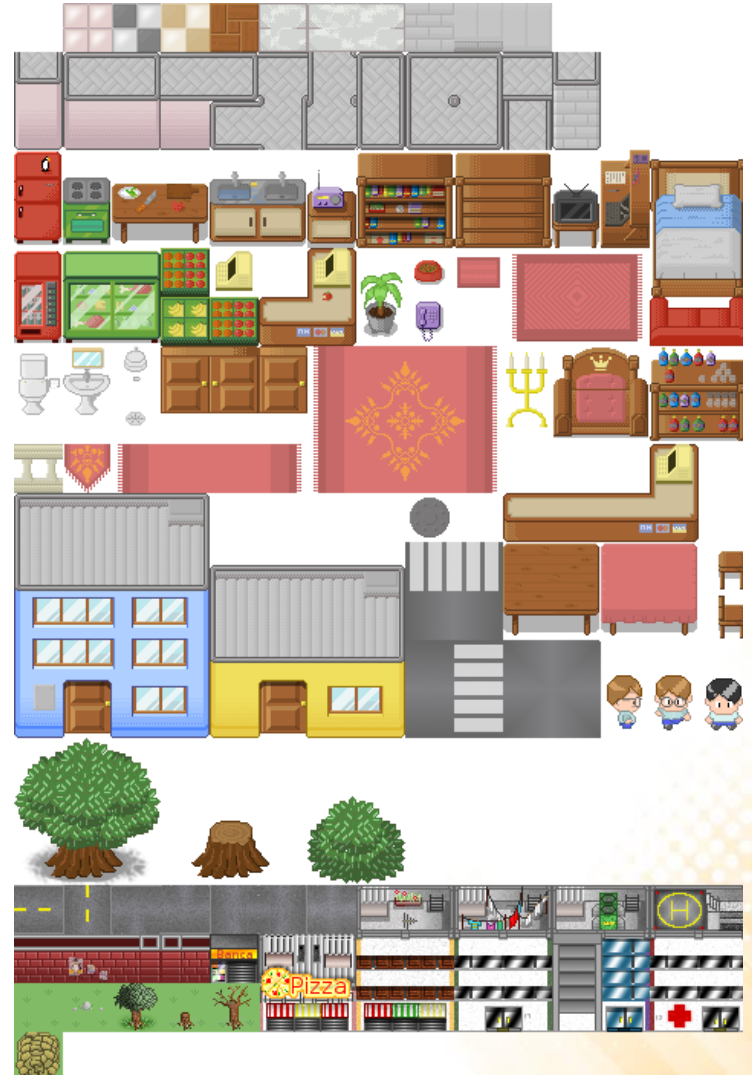
```
font = love.graphics.newFont(filename, size)
```

Exemplo:

```
function love.load()  
    myfont = love.graphics.newFont("anyfont.ttf", 20)  
    love.graphics.setFont(myfont)  
end  
  
function love.draw()  
    love.graphics.print("Hello world!", 100, 200)  
end
```

Texture Atlas

- Forma mais eficiente de armazenar sprites na memória;
 - Somente uma textura é armazenada
- Simplifica a importação de sprites;



Texture Atlas

Mapa

```
XXXXXXXXXXXXXXXXXX  
XXXXXXXXXXXXXXXXXX  
XXXXXXXXXXXXXXXXXX  
XXXXXXXXXXXXXXXXXX  
XXXXXXXXXXXXXXXXXX  
XXXBXXXXXXXXXXXXX  
XXBBBXXXXXXXXXXXX  
GGGGGGGAAGGGGG  
TTTTTTTTTPTTTTT  
TTTTTTTTTPTTTTT
```

Tileset



http://www.inf.puc-rio.br/~elima/intro-eng/plataform_map1.zip

Texture Atlas

```
local mapa = {}
local tilesetImage
local tileQuads = {}
local tileSize = 64

function LoadTiles(filename, nx, ny)
    tilesetImage = love.graphics.newImage(filename)
    local count = 1
    for i = 0, nx, 1 do
        for j = 0, ny, 1 do
            tileQuads[count] = love.graphics.newQuad(i * tileSize ,
                                                       j * tileSize, tileSize, tileSize,
                                                       tilesetImage:getWidth(),
                                                       tilesetImage:getHeight())

            count = count + 1
        end
    end
end

end
.
.
.
```

```
.  
.   
.   
function LoadMap(filename)  
    local file = io.open(filename)  
    local i = 1  
    for line in file:lines() do  
        mapa[i] = {}  
        for j=1, #line, 1 do  
            mapa[i][j] = line:sub(j,j)  
        end  
        i = i + 1  
    end  
    file:close()  
end
```

```
function love.load()  
    LoadMap("plataform_map.txt")  
    LoadTiles("plataform_tileset.png", 2, 2)  
    love.graphics.setBackgroundColor(152,209,250)  
end
```

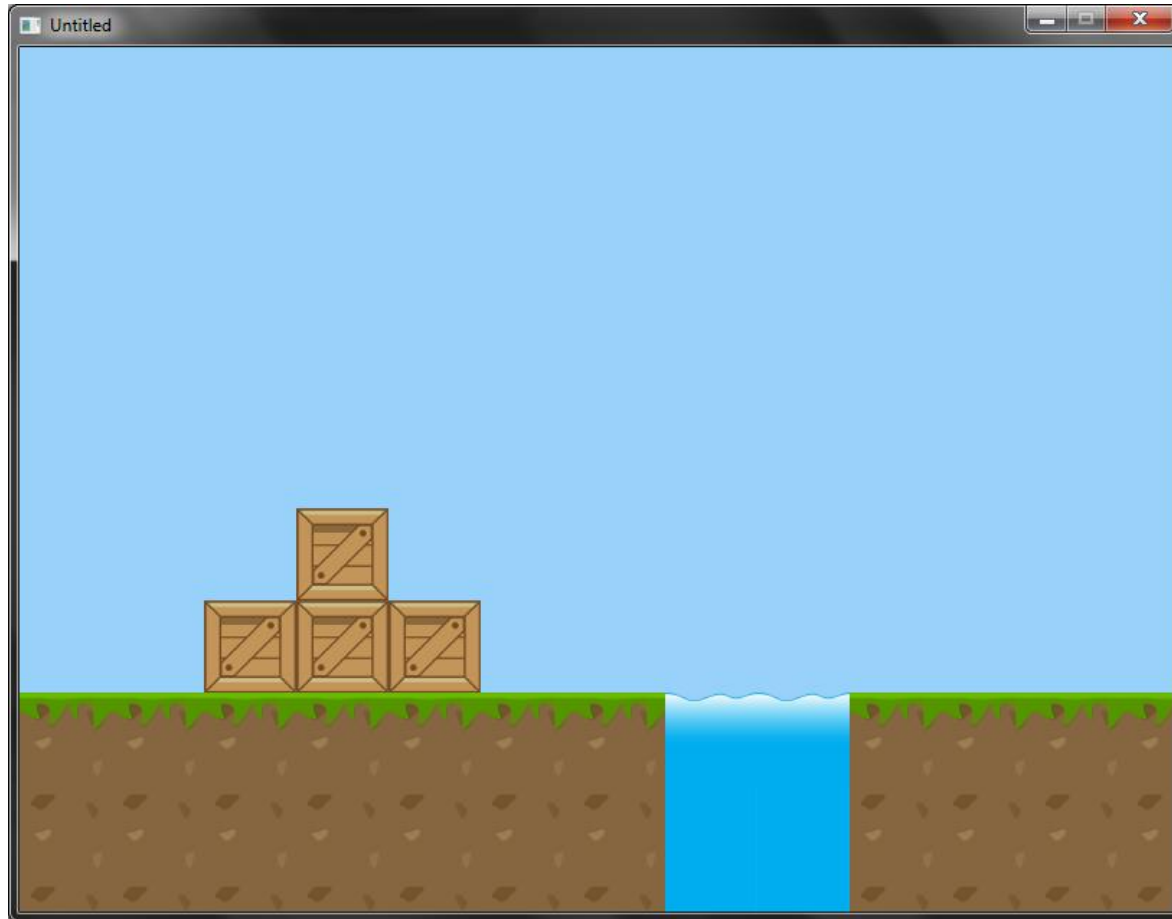
```
.  
.   
.   

```

.
.br/>.

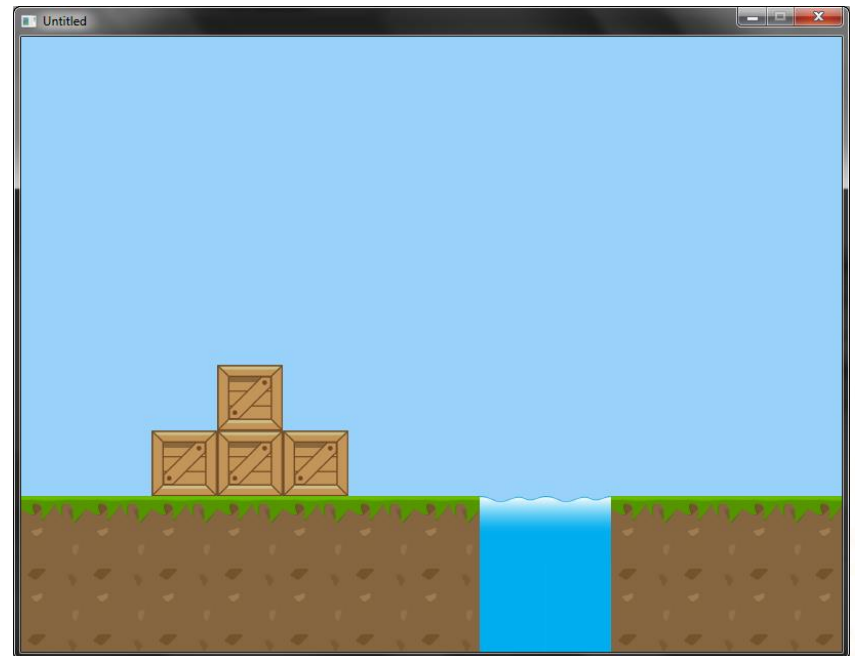
```
function love.draw()  
  for i=1, 10, 1 do      --Percorre a matriz e desenha quadrados imagens  
    for j=1, 14, 1 do  
      if (mapa[i][j] == "G") then  
        love.graphics.draw(tilesetImage, tileQuads[1],  
          (j * tileSize) - tileSize, (i * tileSize) - tileSize)  
      elseif (mapa[i][j] == "T") then  
        love.graphics.draw(tilesetImage, tileQuads[4],  
          (j * tileSize) - tileSize, (i * tileSize) - tileSize)  
      elseif (mapa[i][j] == "A") then  
        love.graphics.draw(tilesetImage, tileQuads[7],  
          (j * tileSize) - tileSize, (i * tileSize) - tileSize)  
      elseif (mapa[i][j] == "P") then  
        love.graphics.draw(tilesetImage, tileQuads[8],  
          (j * tileSize) - tileSize, (i * tileSize) - tileSize)  
      elseif (mapa[i][j] == "B") then  
        love.graphics.draw(tilesetImage, tileQuads[6],  
          (j * tileSize) - tileSize, (i * tileSize) - tileSize)  
      end  
    end  
  end  
end  
end
```

Texture Atlas



Tile-Based Scrolling

```
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX  
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX  
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX  
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX  
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX  
XXXBXXXXXXXXXXXXXXXXXBXXXXXXXXXXXXX  
XXBBBXXXXXXXXXXXXXXXXBBBXXXXXXXXXXXXX  
GGGGGGGAAGGGGGGGGGGGGGGAAGGGGGG  
TTTTTTTTPTTTTTTTTTTTTTTTTTPTTTTTT  
TTTTTTTTPTTTTTTTTTTTTTTTTTPTTTTTT
```



http://www.inf.puc-rio.br/~elima/intro-eng/plataform_map2.zip

Tile-Based Scrolling

```
local mapa = {}
local tilesetImage
local tileQuads = {}
local tileSize = 64

local mapa_config = {
    mapaSize_x = 28,
    mapaSize_y = 10,
    mapaDisplay_x = 14,
    mapaDisplay_y = 10
}

local camera = {
    pos_x = 1,
    pos_y = 1,
    speed = 120
}
```

```
function LoadTiles(filename, nx, ny)
    tilesetImage = love.graphics.newImage(filename)
    local count = 1
    for i = 0, nx, 1 do
        for j = 0, ny, 1 do
            tileQuads[count] = love.graphics.newQuad(i * tileSize, j *
                tileSize, tileSize, tileSize,
                tilesetImage:getWidth(),
                tilesetImage:getHeight())

            count = count + 1
        end
    end
end
```

```
function LoadMap(filename)
    local file = io.open(filename)
    local i = 1
    for line in file:lines() do
        mapa[i] = {}
        for j=1, #line, 1 do
            mapa[i][j] = line:sub(j,j)
        end
        i = i + 1
    end
    file:close()
end
```

```
function love.load()
    LoadMap("plataform_map.txt")
    LoadTiles("plataform_tileset.png", 2, 2)
    love.graphics.setBackgroundColor(152,209,250)
end

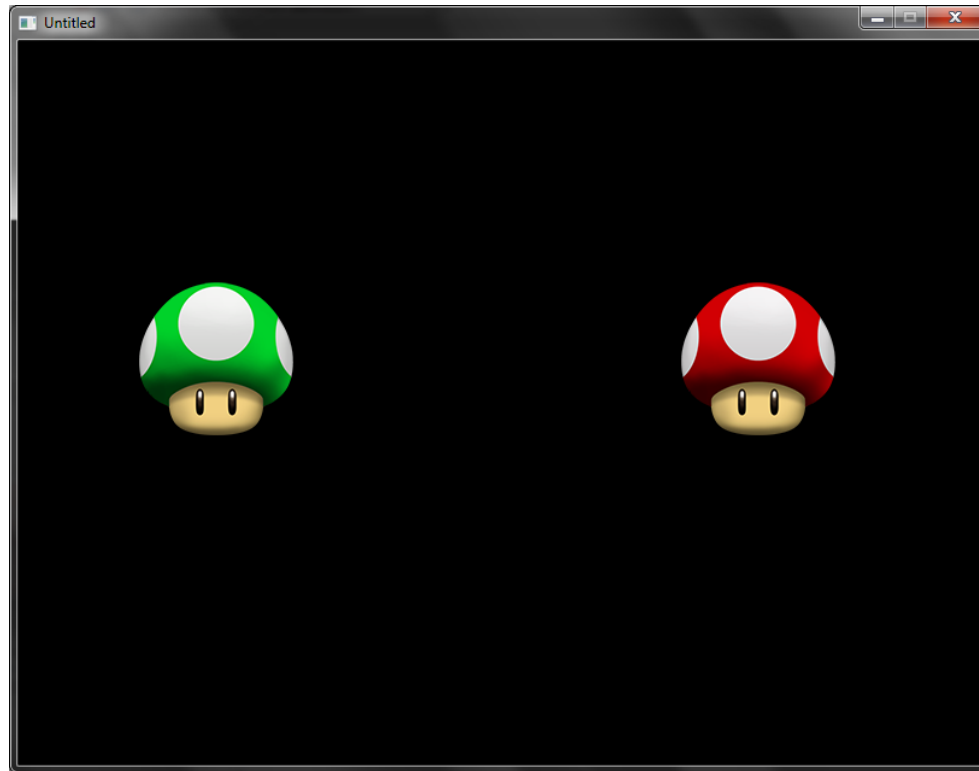
function love.update(dt)
    if love.keyboard.isDown("right") then
        camera.pos_x = camera.pos_x + (camera.speed * dt)
    elseif love.keyboard.isDown("left") then
        camera.pos_x = camera.pos_x - (camera.speed * dt)
    end

    if camera.pos_x < 0 then
        camera.pos_x = 0
    elseif camera.pos_x > mapa_config.mapaSize_x * tileSize -
        mapa_config.mapaDisplay_x * tileSize - 1 then
        camera.pos_x = mapa_config.mapaSize_x * tileSize -
            mapa_config.mapaDisplay_x * tileSize - 1
    end
end
```



```
function love.draw()
    offset_x = math.floor(camera.pos_x % tileSize)
    first_tile_x = math.floor(camera.pos_x / tileSize)
    for y=1, mapa_config.mapaDisplay_y, 1 do
        for x=1, mapa_config.mapaDisplay_x, 1 do
            if (mapa[y][first_tile_x + x] == "G") then
                love.graphics.draw(tilesetImage, tileQuads[1],
                    ((x - 1)*tileSize) - offset_x , ((y-1)*tileSize))
            elseif (mapa[y][first_tile_x + x] == "T") then
                love.graphics.draw(tilesetImage, tileQuads[4],
                    ((x-1)*tileSize) - offset_x , ((y-1)*tileSize))
            elseif (mapa[y][first_tile_x + x] == "A") then
                love.graphics.draw(tilesetImage, tileQuads[7],
                    ((x-1)*tileSize) - offset_x , ((y-1)*tileSize))
            elseif (mapa[y][first_tile_x + x] == "P") then
                love.graphics.draw(tilesetImage, tileQuads[8],
                    ((x-1)*tileSize) - offset_x , ((y-1)*tileSize))
            elseif (mapa[y][first_tile_x + x] == "B") then
                love.graphics.draw(tilesetImage, tileQuads[6],
                    ((x-1)*tileSize) - offset_x , ((y-1)*tileSize))
            end
        end
    end
end
end
end
```

Áudio



www.inf.puc-rio.br/~elima/intro-eng/exemplo_audio.zip

Áudio

```
local icon_1up
local icon_super
local audio_1up
local audio_super

function love.load()
    icon_1up = love.graphics.newImage("1up.png")
    icon_super = love.graphics.newImage("super.png")
    audio_1up = love.audio.newSource("1up.mp3", "static")
    audio_super = love.audio.newSource("super.mp3")
end

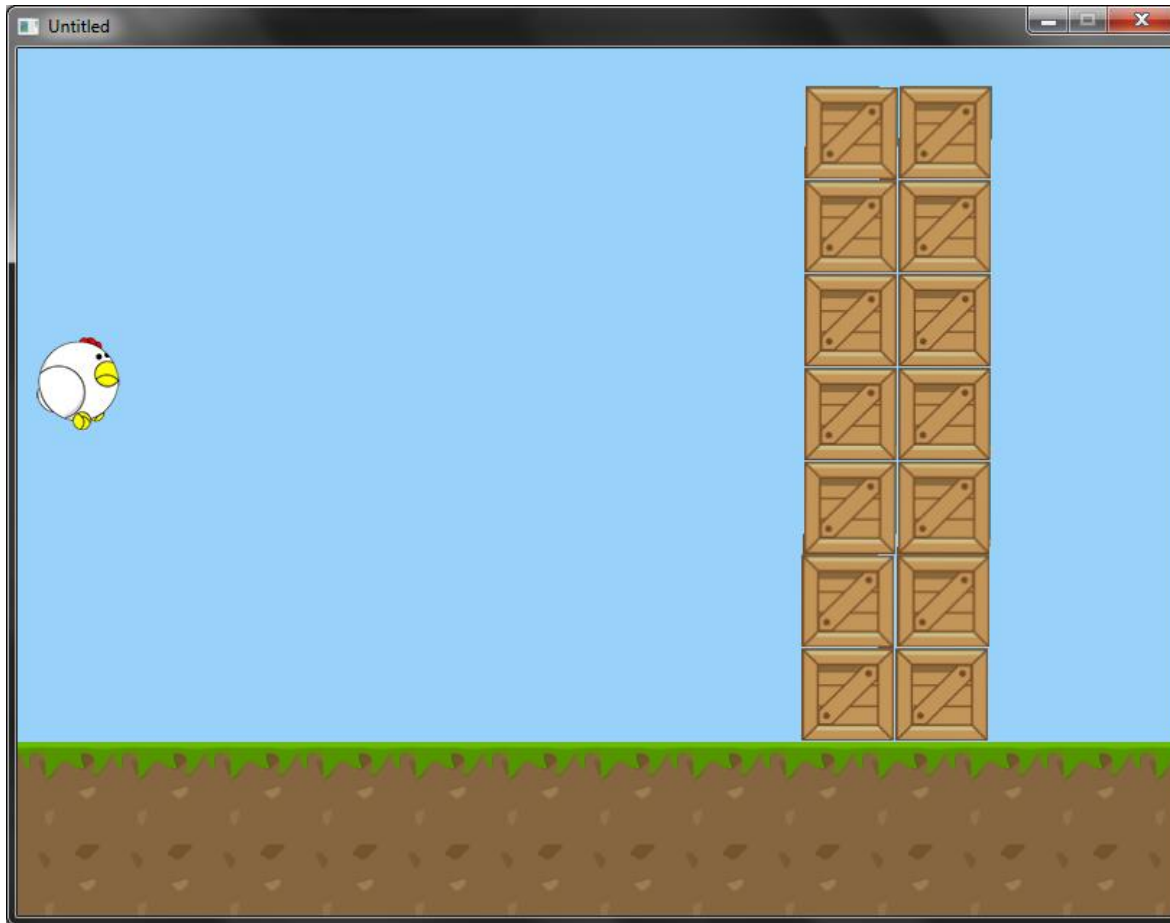
function CheckClick(x1,y1,w1,h1,x2,y2)
    return x1 < x2+1 and x2 < x1+w1 and y1 < y2+1 and y2 < y1+h1
end

function love.draw()
    love.graphics.draw(icon_1up, 100, 200)
    love.graphics.draw(icon_super, 550, 200)
end
```

Áudio

```
function love.mousepressed(x, y, button)
  if button == "l" then
    if CheckClick(550,200,128,128,x,y) then
      love.audio.play(audio_super)
    end
    if CheckClick(100,200,128,128,x,y) then
      love.audio.play(audio_lup)
    end
  elseif button == "r" then
    if CheckClick(550,200,128,128,x,y) then
      love.audio.stop(audio_super)
    end
    if CheckClick(100,200,128,128,x,y) then
      love.audio.stop(audio_lup)
    end
  end
end
end
```

Física Simples



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