

Programming Fundamentals

Lecture 07 – Arrays, Matrices, Animations and World Representations

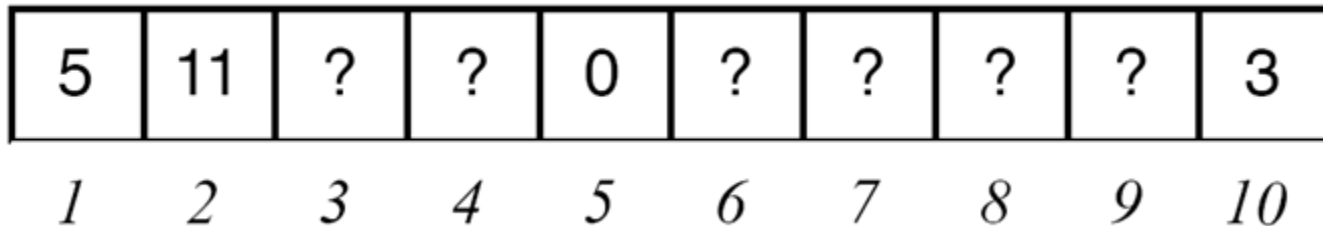
Edirlei Soares de Lima

<edirlei.lima@universidadeeuropeia.pt>



Arrays

- Arrays are sequences of items (like variables) of the same type.
 - Each item is identified by an index (integer).



- With arrays we can store in memory sequences of values (numbers, text, imagens, etc.), which are all associated with a single variable (the array).

Lua Tables

- In Lua, all data structures, including arrays, are implemented using tables.
- Tables are the main (in fact, the only) data structuring mechanism in Lua (a very powerful one).
- A Lua table is defined by a set of **pairs key-data**, where the data is referenced by the key.
 - The key (index) can be of any type of data (except nil).

Arrays in Lua

- In Lua, are implemented through tables indexed by **integer numbers**.
- Different from many other programming languages, in Lua **we don't need to define the maximum size** of an array.
- **Creating a new array:**

```
my_array = {}
```

Arrays in Lua

- **Initializing some positions of the array:**

```
my_array[1] = 5  
my_array[2] = 11  
my_array[5] = 0  
my_array[10] = 3
```

5	11	?	?	0	?	?	?	?	3
<i>1</i>	<i>2</i>	<i>3</i>	<i>4</i>	<i>5</i>	<i>6</i>	<i>7</i>	<i>8</i>	<i>9</i>	<i>10</i>

Arrays in Lua

- We can access the **values stored in the array** using their **index**.

```
myarray = { }
```

1 2 3 4 5

5	?	?	8	1
---	---	---	---	---

```
myarray[1] = 5
```

```
myarray[4] = 8
```

```
myarray[5] = 1
```

Arrays in Lua

- **Declaring and initializing an array:**

```
a = {}      -- new array

for i=1, 1000, 1 do
    a[i] = 0
end
```

- Indirectly we are defining that the maximum size of the array is 1000.
- We can also declare and initialize an array using a single statement:

```
squares = {1, 4, 9, 16, 25, 36, 49, 64}
```

Example 1: Displaying the Values of an Array

```
myarray = {2.3, 5.4, 1.0, 7.6, 8.8, 3.9}

for x = 1, 6, 1 do
    io.write(myarray[x], "\n")
end
```

- Inverse order?

```
myarray = {2.3, 5.4, 1.0, 7.6, 8.8, 3.9}

for x = 6, 1, -1 do
    io.write(myarray[x], "\n")
end
```


Example 2: Sum of the Values of an Array

```
myarray = {2.3, 5.4, 1.0, 7.6, 8.8, 3.9}

total = 0

for x = 1, 6, 1 do
    total = total + myarray[x]
end

io.write(total)
```

Exemplo 3: Finding the Larger Value

```
myarray = {2.3, 5.4, 1.0, 7.6, 8.8, 3.9}

largervalue = myarray[1]

for x = 2, 6, 1 do
    if myarray[x] > largervalue then
        largervalue = myarray[x]
    end
end

io.write(largervalue)
```

Back to the “Hello World”

- In the last implementation of the “Hello World”, we implemented the movement of 20 “Hello World’s”.
 - **Problem:** the 20 “Hello World’s” moved all together



- How can we make the “Hello World’s” to move independently of each other?

```
local px    -- position of the text in the x axis

function love.load()
    love.graphics.setColor(0, 0, 0)
    love.graphics.setBackgroundColor(1, 1, 1)
    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
```

```
local array_px = {}  -- array of positions in the x axis
```

```
function love.load()
```

```
    love.graphics.setColor(0, 0, 0)
```

```
    love.graphics.setBackgroundColor(1, 1, 1)
```

```
    for y = 1, 20, 1 do
```

```
        array_px[y] = love.math.random(0, 800)
```

```
    end
```

```
end
```

```
function love.update(dt)
```

```
    for y = 1, 20, 1 do
```

```
        array_px[y] = array_px[y] + (100 * dt)
```

```
        if array_px[y] > love.graphics.getWidth() then
```

```
            array_px[y] = 0
```

```
        end
```

```
    end
```

```
end
```

```
function love.draw()
```

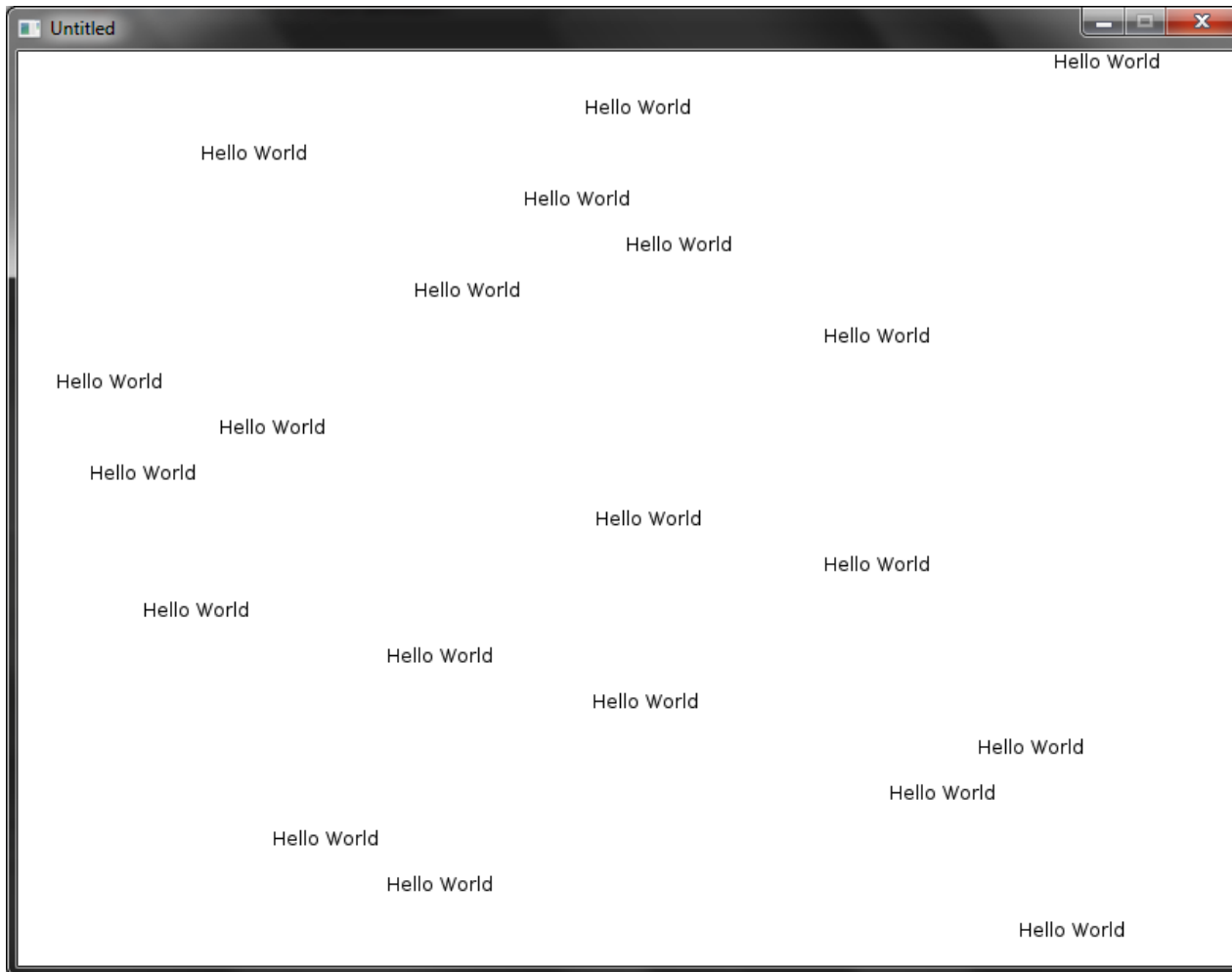
```
    for y = 1, 20, 1 do
```

```
        love.graphics.print("Hello World", array_px[y], y * 30)
```

```
    end
```

```
end
```

Back to the “Hello World”



Generating Random Numbers

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

Example 1:

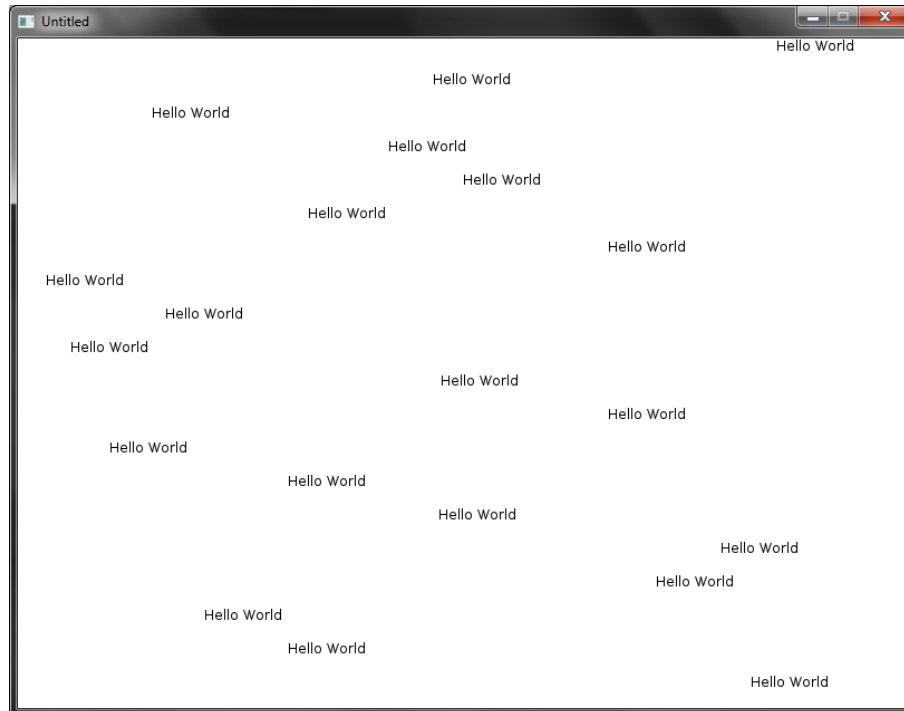
```
x = love.math.random(1, 100) -- generates a number between 1  
                             and 100
```

Exemplo 2:

```
vet = {} -- fill an array with 100 random numbers  
for i=1, 100, 1 do  
    vet[i] = love.math.random(1, 100)  
end
```

Exercise 1

- 1) Modify the code of the “Hello World” in order to have each “Hello World” moving with a different speed (randomly chosen).



Arrays and Animations

- Animations are created by **sequences of images**.

– **Example:**



- We can store animations as arrays of images!

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

```
local hero_walk = {}  -- array of images
local hero_anim_frame = 1
local hero_pos_x = 100
local hero_pos_y = 225

function love.load()
  for x = 1, 4, 1 do  -- load the animation frames
    hero_walk[x] = love.graphics.newImage("Hero_Walk_0" .. x .. ".png")
  end
end

function love.update(dt)
  if love.keyboard.isDown("right") then

    hero_pos_x = hero_pos_x + (100 * dt)  -- moves the character
    hero_anim_frame = hero_anim_frame + 1  -- increases the anim. index
    if hero_anim_frame > 4 then  -- animation loop
      hero_anim_frame = 1
    end
  end
end

function love.draw()  -- draw the character using the animation index
  love.graphics.draw(hero_walk[hero_anim_frame], hero_pos_x, hero_pos_y)
end
```

Example of Animation



- **Problem:** we didn't control the speed of the animation!
 - The faster the computer, the faster the animation will be played.

```
local hero_walk = {}
local hero_anim_frame = 1
local hero_pos_x = 100
local hero_pos_y = 225
local hero_anim_time = 0  -- variable to control the animation time

function love.load()
  for x = 1, 4, 1 do
    hero_walk[x] = love.graphics.newImage("Hero_Walk_0" .. x .. ".png")
  end
end

function love.update(dt)
  if love.keyboard.isDown("right") then
    hero_pos_x = hero_pos_x + (100 * dt)
    hero_anim_time = hero_anim_time + dt  -- increases the time with dt
    if hero_anim_time > 0.1 then  -- when time gets to 0.1
      hero_anim_frame = hero_anim_frame + 1  -- go to the next frame
      if hero_anim_frame > 4 then
        hero_anim_frame = 1
      end
      hero_anim_time = 0  -- reset the time counter
    end
  end
end

function love.draw()
  love.graphics.draw(hero_walk[hero_anim_frame], hero_pos_x, hero_pos_y)
end
```

Tables in Lua

- Lua tables allow us to associate names to its elements.
- With this feature, we can create structures to store related data:

```
player1 = {  
    name      = "John",  
    image     = love.graphics.newImage("John.png"),  
    score     = 1000,  
    lives     = 3,  
    power     = 50,  
    px        = 300,  
    py        = 300  
}
```

Tables in Lua

- We can access the elements of a table by its name:

```
io.write(player1.score)
```

```
•
```

```
•
```

```
love.graphics.draw(player1.image, player1.px,  
                    player1.py)
```

```
•
```

```
•
```

Exercise 2

- 2) Modify the code of the animation example in order to organize and store the variables that are related with the character in a table:

```
local hero_walk = {}  
local hero_anim_frame = 1  
local hero_pos_x = 100  
local hero_pos_y = 225  
local hero_anim_time = 0
```

- Also change the code in order to use the new table. At the end, everything must be working the same way as before the introduction of the table.

Exercise 3

- 3) Continue the implementation of the last exercise by adding the animations and movement of the character to all other directions. Use the following images:



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

Table Functions

- Size of an array:

```
arr = {1, 2, 1, 6}  
size = table.getn(arr)
```

- Add elements:

```
arr = {1, 2, 1, 6}  
table.insert(arr, 8)  
table.insert(arr, 1, 10)
```

- Remove elements:

```
arr = {1, 2, 1, 6}  
table.remove(arr, 4)  
table.remove(arr, 1)
```

Example 1: Collision Detection and Arrays

```
require "vector2"
```

world.lua

```
function CreateObject(x, y, w, h)
```

```
    return {position = vector2.new(x, y), size = vector2.new(w, h)}  
end
```

```
function DrawWorld(world)
```

```
    for i = 1, table.getn(world), 1 do  
        love.graphics.rectangle("fill", world[i].position.x,  
                                world[i].position.y, world[i].size.x, world[i].size.y)  
    end  
end
```

```
function GetBoxCollisionDirection(x1,y1,w1,h1,x2,y2,w2,h2)
```

```
    local xdist = math.abs((x1 + (w1 / 2)) - (x2 + (w2 / 2)))  
    local ydist = math.abs((y1 + (h1 / 2)) - (y2 + (h2 / 2)))
```

collision.lua

```
    local combinedwidth = (w1 / 2) + (w2 / 2)  
    local combinedheight = (h1 / 2) + (h2 / 2)
```

```
    ...
```

Example 1: Collision Detection and Arrays

collision.lua

```
if(xdist > combinedwidth) then
    return vector2.new(0, 0)
end
if(ydist > combinedheight) then
    return vector2.new(0, 0)
end
local overlapx = math.abs(xdist - combinedwidth)
local overlapx = math.abs(ydist - combinedheight)
local playerdir = vector2.normalize(vector2.sub(vector2.new(x1, y1),
                                                vector2.new(x2, y2)))

local collisiondir
if overlapx > overlapx then
    collisiondir = vector2.normalize(vector2.new(0, playerdir.y *
                                                overlapx))
elseif overlapx < overlapx then
    collisiondir = vector2.normalize(vector2.new(playerdir.x *
                                                overlapx, 0))
else
    collisiondir = vector2.normalize(vector2.new(playerdir.x *
                                                overlapx, playerdir.y * overlapx))
end
return collisiondir
end
```


Example 1: Collision Detection and Arrays

player.lua

```
local friction = vector2.mult(player.velocity, -1)
friction = vector2.normalize(friction)
friction = vector2.mult(friction, player.frictioncoefficient)
acceleration = vector2.applyForce(friction, player.mass,
                                   acceleration)

local movedirection = vector2.new(0, -1)
if love.keyboard.isDown("right") then
    local move = vector2.new(500, 0)
    acceleration = vector2.applyForce(move, player.mass, acceleration)
    movedirection.x = 1
end
if love.keyboard.isDown("left") then
    local move = vector2.new(-500, 0)
    acceleration = vector2.applyForce(move, player.mass, acceleration)
    movedirection.x = -1
end
if love.keyboard.isDown("up") and (player.onGround) then
    local jump = vector2.new(0, -50000)
    acceleration = vector2.applyForce(jump, player.mass, acceleration)
    movedirection.y = 1
    player.onGround = false
end
```

Example 1: Collision Detection and Arrays

player.lua

```
local futurevelocity = vector2.add(player.velocity,  
                                   vector2.mult(acceleration, dt))  
futurevelocity = vector2.limit(futurevelocity, player.maxspeed)  
local futureposition = vector2.add(player.position,  
                                   vector2.mult(futurevelocity, dt))  
  
acceleration = CheckCollision(world, futureposition,  
                              movedirection, acceleration)  
  
player.velocity = vector2.add(player.velocity,  
                               vector2.mult(acceleration, dt))  
player.velocity = vector2.limit(player.velocity, player.maxspeed)  
player.position = vector2.add(player.position,  
                              vector2.mult(player.velocity, dt))  
  
end  
  
function DrawPlayer()  
    love.graphics.setColor(0.2, 0.2, 0.9)  
    love.graphics.rectangle("fill", player.position.x,  
                           player.position.y, player.size.x, player.size.y)  
  
end
```

Example 1: Collision Detection and Arrays

```
function CheckCollision(world, futureposition, movedirection,
                        acceleration)
    for i = 1, table.getn(world), 1 do
        local collisiondir = GetBoxCollisionDirection(futureposition.x,
            futureposition.y, player.size.x, player.size.y,
            world[i].position.x, world[i].position.y,
            world[i].size.x, world[i].size.y)
        if not (collisiondir.x == 0 and collisiondir.y == 0) then
            if collisiondir.y == movedirection.y then --down collision
                player.velocity.y = 0
                acceleration.y = 0
                player.onGround = true
            elseif collisiondir.y == 1 then --up collision
                player.velocity.y = 0
                acceleration.y = 0
            elseif movedirection.x ~= collisiondir.x then --side collision
                player.velocity.x = 0
                acceleration.x = 0
            end
        end
    end
end
return acceleration
end
```

player.lua

Example 1: Collision Detection and Arrays

```
require "vector2"  
require "world"  
require "player"  
  
local world = {}  
  
function love.load()  
    world[1] = CreateObject(50, 550, 700, 50)  
    world[2] = CreateObject(500, 450, 50, 50)  
    world[3] = CreateObject(200, 500, 100, 50)  
end  
  
function love.update(dt)  
    UpdatePlayer(dt, world)  
end  
  
function love.draw()  
    love.graphics.setColor(0.2, 0.2, 0.9)  
    DrawPlayer()  
    love.graphics.setColor(0.2, 0.8, 0.2)  
    DrawWorld(world)  
end
```

main.lua

Matrices

- Matrices are two-dimensional arrays.
- A matrix stores data in an organized form with rows and columns.

3	1	8	6	1
7	2	5	4	9
1	9	3	1	2
5	8	6	7	3
6	4	9	2	1

Matrices in Lua

- **Declaring and initializing a matrix:**

```
my_matrix = {}          -- new matrix

for i=1, 10, 1 do
  my_matrix[i] = {}    -- new row
  for j=1, 10, 1 do
    my_matrix[i][j] = 0
  end
end
end
```

- We are defining and initializing a matrix with 10 columns e 10 rows.

Matrices in Lua

- We can access the values **stored in the matrix** using their two-dimensional **indexes**.

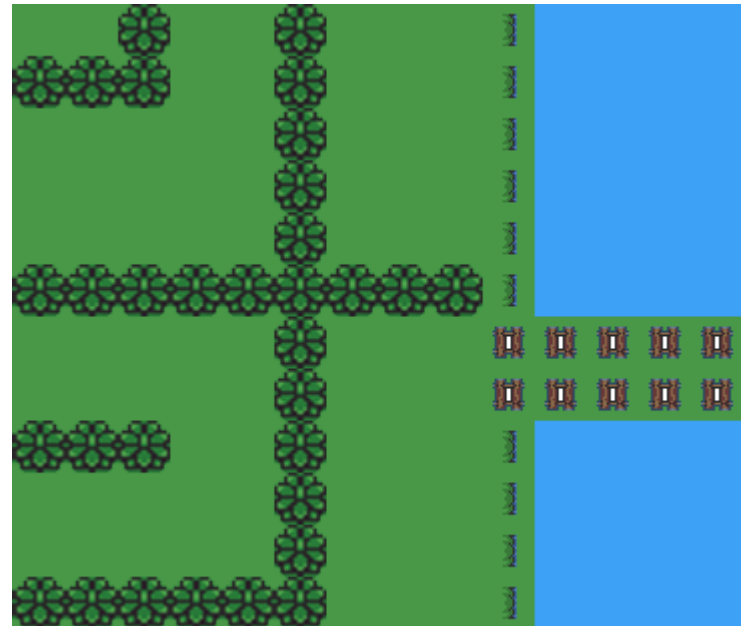
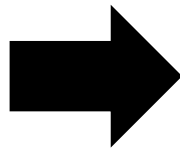
	0	1	2
0	5	?	1
1	?	?	?
2	?	8	?

```
my_matrix[0][0] = 5;  
my_matrix[1][2] = 8;  
my_matrix[2][0] = 1;
```

Matrices and Game Worlds

- We can use matrices to represent 2D game worlds.
 - **Example:**

```
XXGXXGXXXEAAAA  
GGGXXGXXXEAAAA  
XXXXXGXXXEAAAA  
XXXXXGXXXEAAAA  
XXXXXGXXXEAAAA  
GGGGGGGGGEAAAA  
XXXXXGXXXPPPPP  
XXXXXGXXXPPPPP  
GGGXXGXXXEAAAA  
XXXXXGXXXEAAAA  
XXXXXGXXXEAAAA  
GGGGGGGXXXEAAAA
```



Matrices and Game Worlds – Example 1

- Generating a random matrix and drawing its elements on screen using colors.

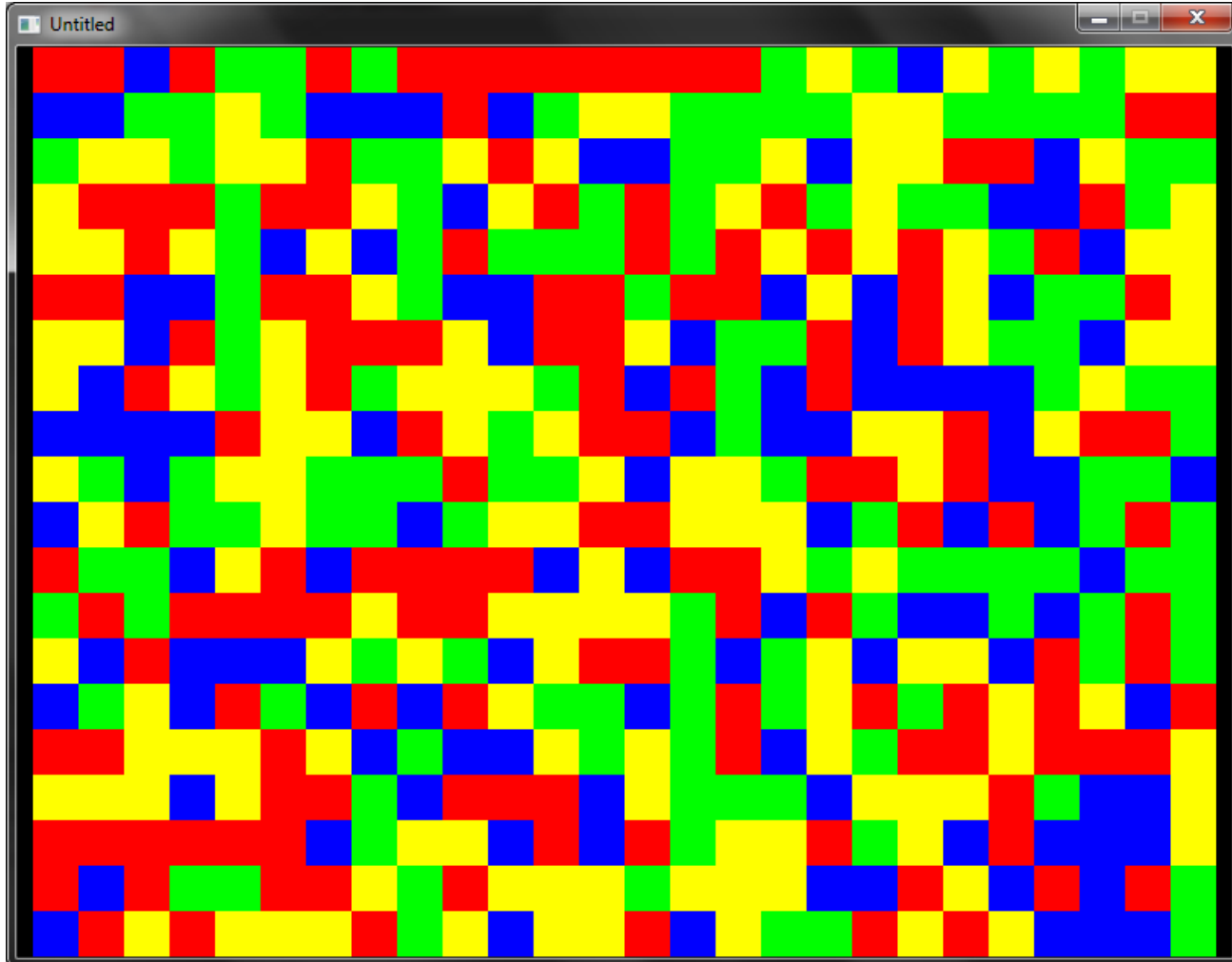
```
1312103010112210
2113021120130132
1203201321012021
2113013023120231
3120312031031201
2010311103013130
0310312130310321
1010101203102031
3210321013210130
3203132031201110
```

```
local world = {}

function love.load()
  for i=1, 26, 1 do    -- initializes a random matrix (26 x 20)
    world[i] = {}
    for j=1, 20, 1 do
      world[i][j] = love.math.random(0, 3)
    end
  end
end

function love.draw()
  for i=1, 26, 1 do -- iterates through the matrix and draw the elements
    for j=1, 20, 1 do
      if (world[i][j] == 0) then
        love.graphics.setColor(1, 0, 0)
      elseif (world[i][j] == 1) then
        love.graphics.setColor(0, 1, 0)
      elseif (world[i][j] == 2) then
        love.graphics.setColor(0, 0, 1)
      elseif (world[i][j] == 3) then
        love.graphics.setColor(1, 1, 0)
      end
      love.graphics.rectangle("fill", (i * 30)-20, (j * 30)-30, 30, 30)
    end
  end
end
```

Matrices and Game Worlds – Example 1



Matrices and Game Worlds – Example 2

- Reading a matrix from a file and drawing its elements on screen using colors.

World1.txt

```
GGGGGGAGGGGGGGG
GGGGGGAGGGGGGGG
GGGGGGAGGGGGGGG
GGGGGGAGGGGGGGG
GGGGGGAAAGGGGGG
GGGGGGGGAGGGGGG
GGGGGGGGAGGGGGG
PPPPPPPPAPPPPP
AAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAA
```


Matrices and Game Worlds – Example 2

```
local world = {}

function LoadMap(filename)      -- reads the content of the file
    local file = io.open(filename)
    local i = 1
    for line in file:lines() do
        world[i] = {}
        for j=1, #line, 1 do
            world[i][j] = line:sub(j,j)
        end
        i = i + 1
    end
    file:close()
end

function love.load()
    LoadMap("World1.txt")
end

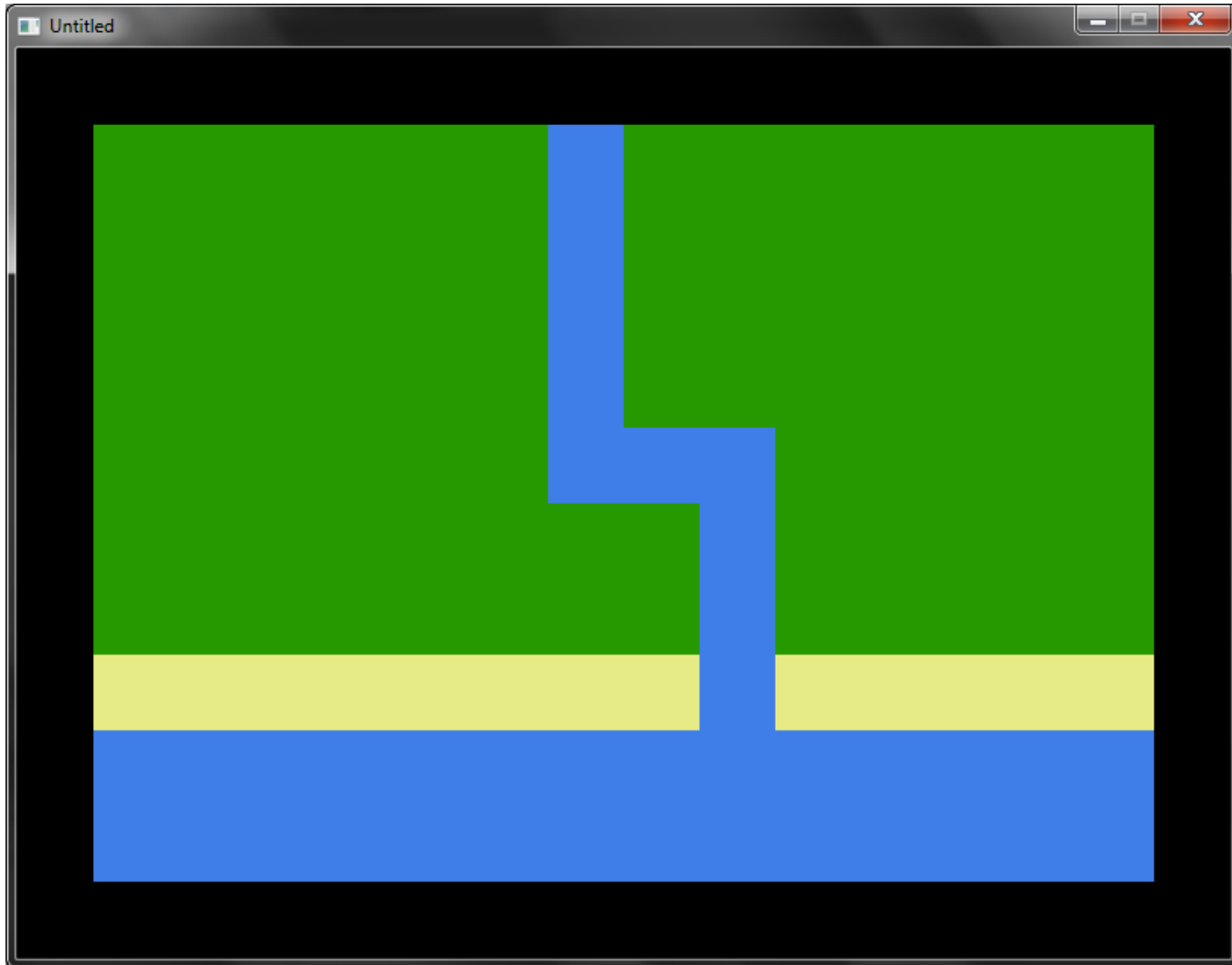
.
.
.
```

Matrices and Game Worlds – Example 2

-
-
-

```
function love.draw()  
  for i=1, 10, 1 do -- iterates through the matrix and draw the elements  
    for j=1, 14, 1 do  
      if (world[i][j] == "P") then  
        love.graphics.setColor(0.901, 0.921, 0.525)  
      elseif (world[i][j] == "G") then  
        love.graphics.setColor(0.149, 0.6, 0)  
      elseif (world[i][j] == "A") then  
        love.graphics.setColor(0.250, 0.490, 0.909)  
      end  
      love.graphics.rectangle("fill", (j * 50), (i * 50), 50, 50)  
    end  
  end  
end
```

Matrices and Game Worlds – Example 2



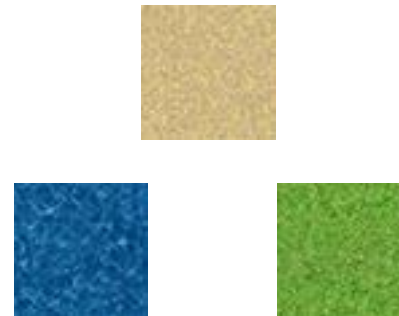
Matrices and Game Worlds – Example 3

- Reading a matrix from a file and drawing its elements on screen using images.

World1.txt

```
GGGGGGGAGGGGGGGG
GGGGGGGAGGGGGGGG
GGGGGGGAGGGGGGGG
GGGGGGGAGGGGGGGG
GGGGGGGAAAGGGGGG
GGGGGGGGGAGGGGGG
GGGGGGGGGAGGGGGG
PPPPPPPPAPPPPP
AAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAA
```

Images:



Matrices and Game Worlds – Example 3

```
local world = {}
local tile_grass
local tile_water
local tile_sand

function LoadMap(filename) -- reads the content of the file
    local file = io.open(filename)
    local i = 1
    for line in file:lines() do
        world[i] = {}
        for j=1, #line, 1 do
            world[i][j] = line:sub(j,j)
        end
        i = i + 1
    end
    file:close()
end

end

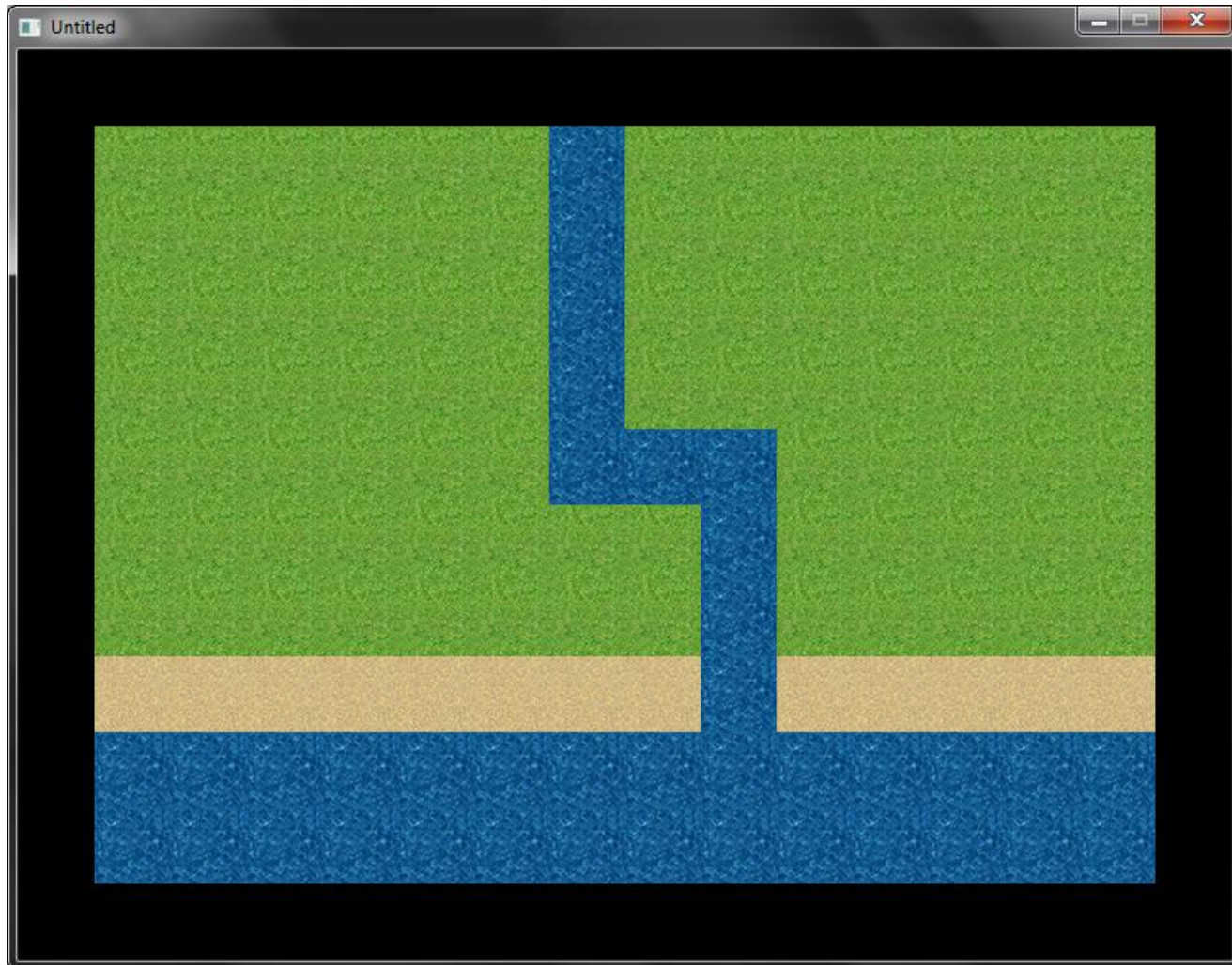
.
.
.
```

-
-
-

```
function love.load()  
  LoadMap("World1.txt")  
  tile_grass = love.graphics.newImage("grass.png")  
  tile_water = love.graphics.newImage("water.png")  
  tile_sand = love.graphics.newImage("sand.png")  
end
```

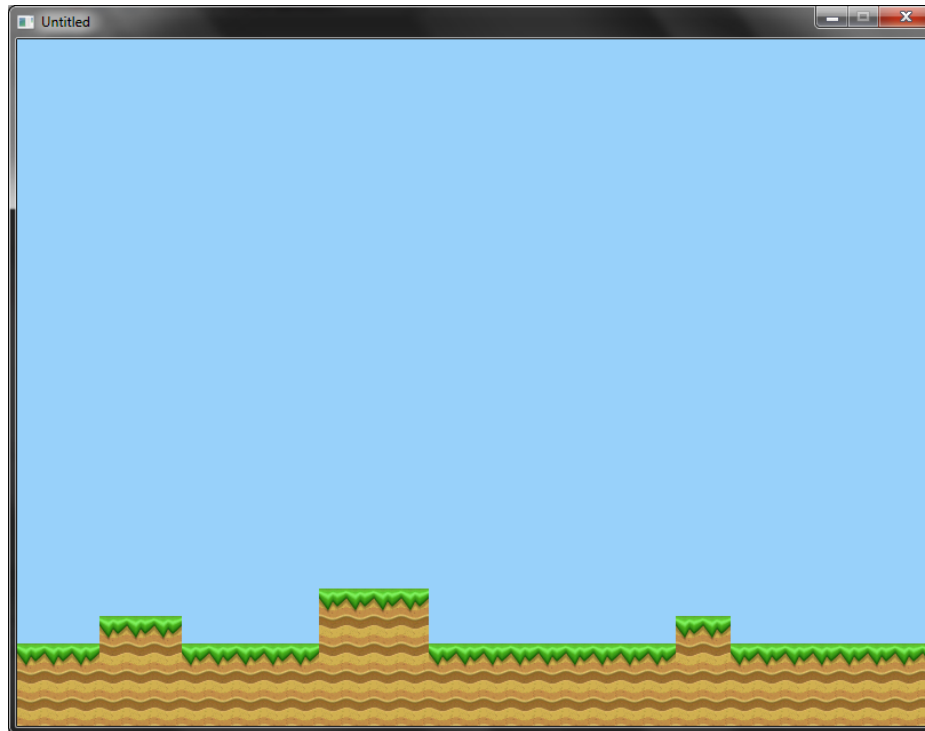
```
function love.draw()  
  for i=1, 10, 1 do -- iterates through the matrix and draw the elements  
    for j=1, 14, 1 do  
      if (world[i][j] == "P") then  
        love.graphics.draw(tile_sand, (j * 50), (i * 50))  
      elseif (world[i][j] == "G") then  
        love.graphics.draw(tile_grass, (j * 50), (i * 50))  
      elseif (world[i][j] == "A") then  
        love.graphics.draw(tile_water, (j * 50), (i * 50))  
      end  
    end  
  end  
end  
end
```

Matrices and Game Worlds – Example 3



Exercise 4

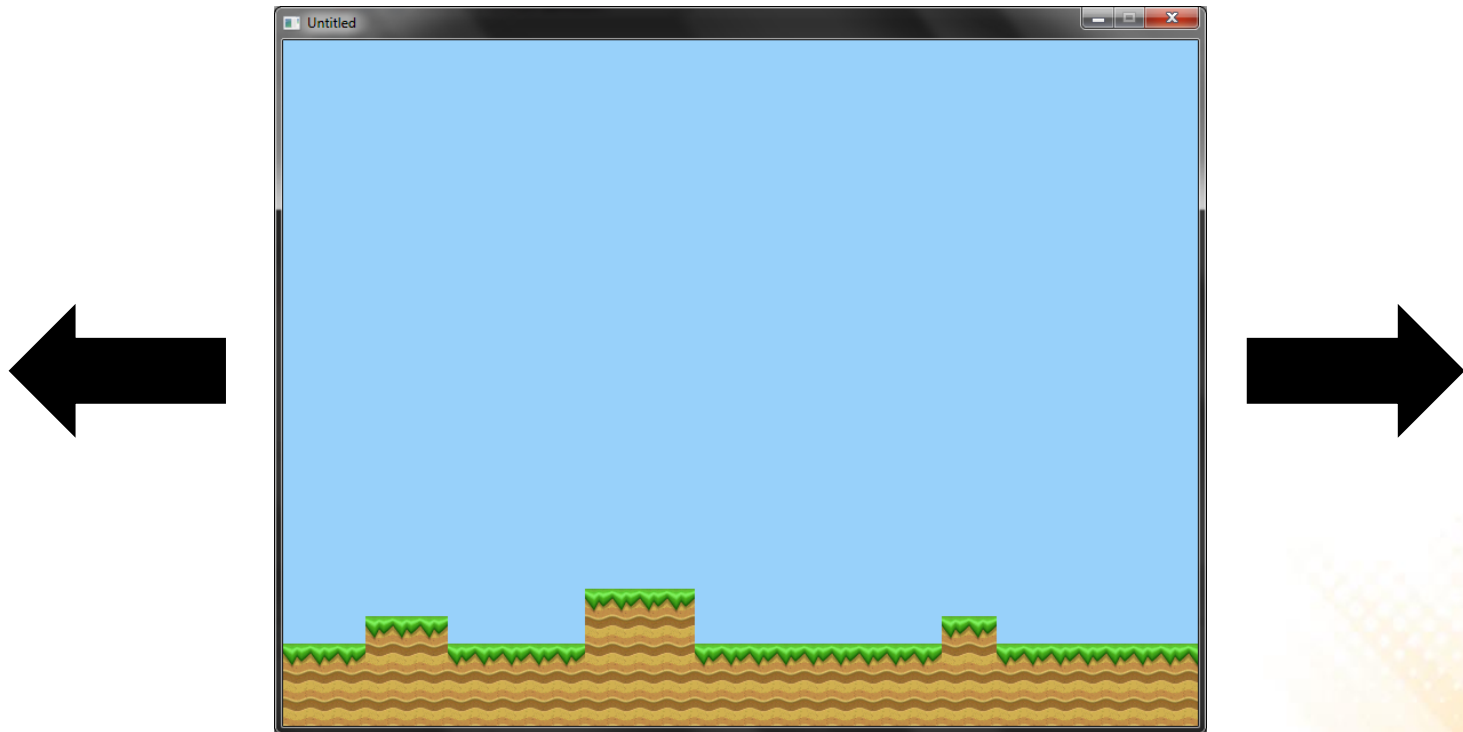
- 4) Implement a program to display the world of a platform game, which is stored in a text file.



Images: http://www.inf.puc-rio.br/~elima/intro-eng/mapa_exercicio3.zip

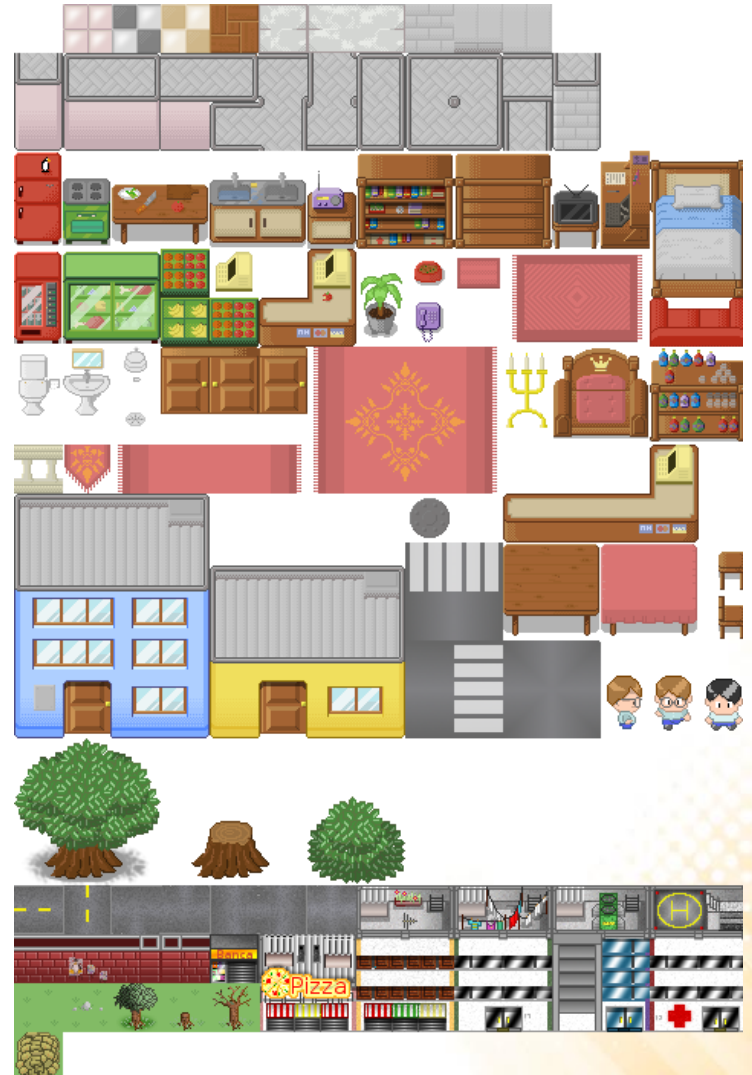
Exercise 5

- 5) Continue the implementation of the last exercise and code a virtual camera to allow the player to move and see the entire environment.
- **Important:** don't draw parts of world that are not visible in the screen.



Texture Atlas

- A texture atlas (also called a sprite sheet or an image sprite) is an image containing a collection of smaller images, usually packed together to reduce the atlas size.
- It is often more efficient to store the textures in a texture atlas which is treated as a single unit by the graphics hardware.



Texture Atlas – Example

World

```
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 – Example

```
local world = {}
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

.
.
.
```

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

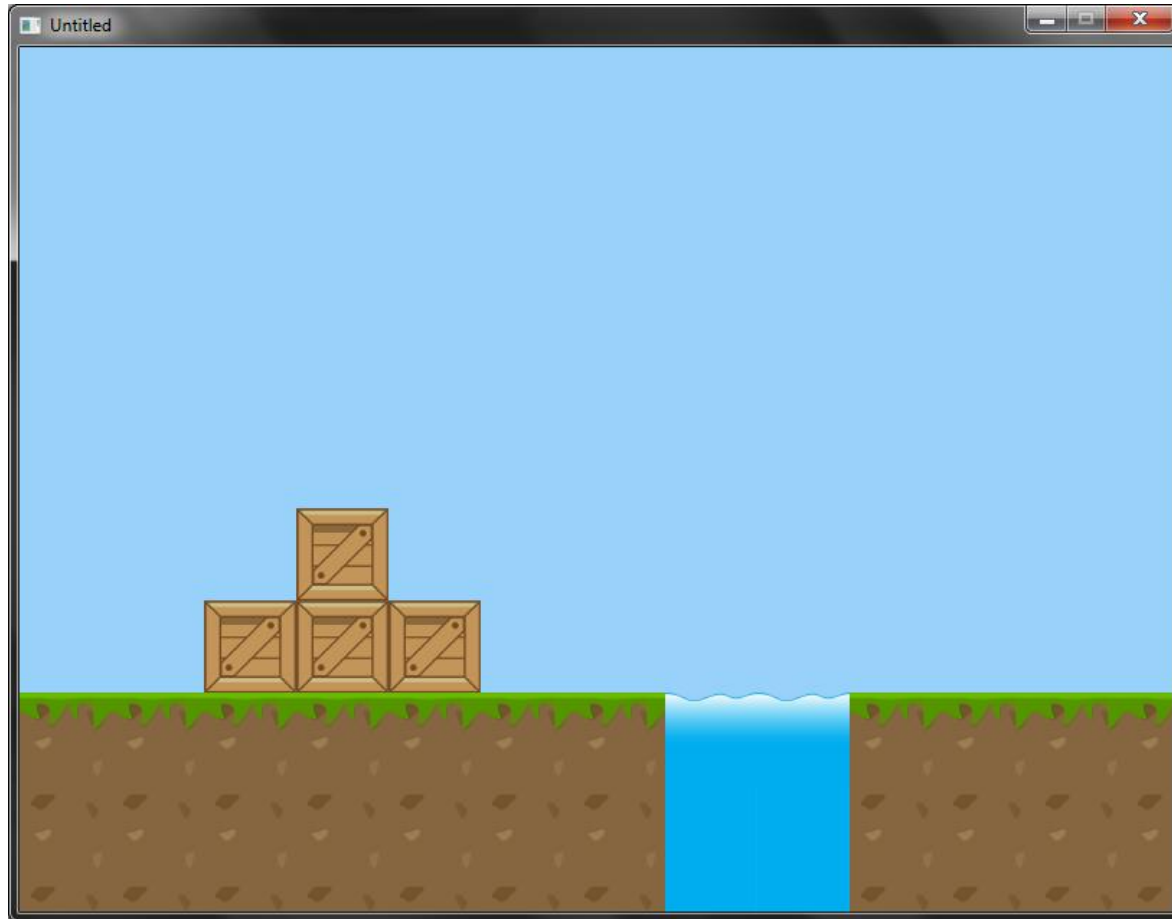
```
function love.load()  
  LoadMap("plataform_map.txt")  
  LoadTiles("plataform_tileset.png", 2, 2)  
  love.graphics.setBackgroundColor(0.6, 0.819, 0.980)  
end
```

```
.  
.br/>.
```

.
.br/>.

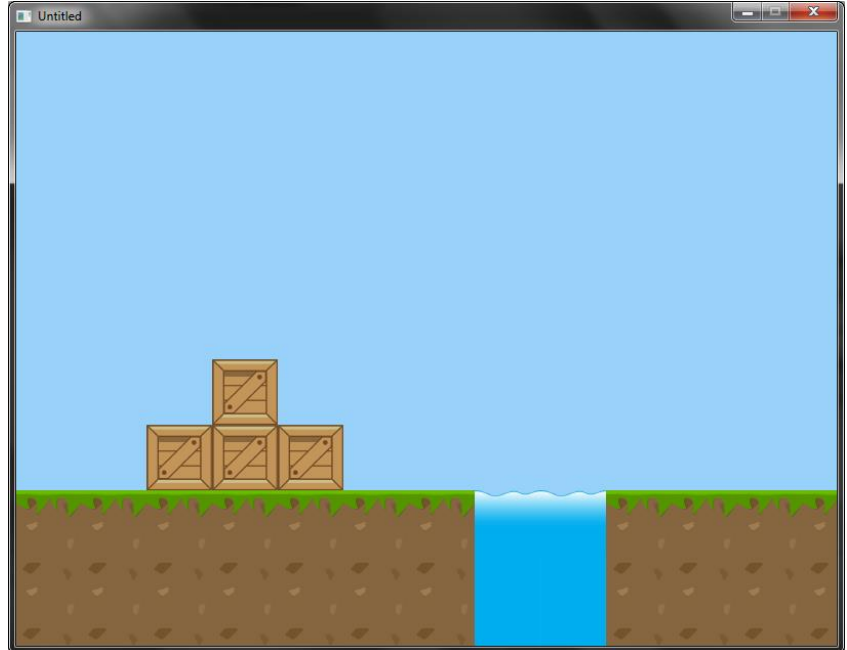
```
function love.draw()  
  for i=1, 10, 1 do  
    for j=1, 14, 1 do  
      if (world[i][j] == "G") then  
        love.graphics.draw(tilesetImage, tileQuads[1],  
          (j * tileSize) - tileSize, (i * tileSize) - tileSize)  
      elseif (world[i][j] == "T") then  
        love.graphics.draw(tilesetImage,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

Texture Atlas – Example



Tile-Based Scrolling – Example

```
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX  
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX  
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX  
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX  
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX  
XXXBXXXXXXXXXXXXXXXXXXBXXXXXXXXXXXXXXXXXX  
XXBBBXXXXXXXXXXXXXXXXXXBBBXXXXXXXXXXXXXXXX  
GGGGGGGAAGGGGGGGGGGGGGGAAGGGGGG  
TTTTTTTTPPTTTTTTTTTTTTTTTTTTTTTTTTTTTTT  
TTTTTTTTPPTTTTTTTTTTTTTTTTTTTTTTTTTTTTT
```



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

Tile-Based Scrolling – Example

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

local world_config = {
    worldSize_x = 28,
    worldSize_y = 10,
    worldDisplay_x = 14,
    worldDisplay_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
        world[i] = {}
        for j=1, #line, 1 do
            world[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(0.6, 0.819, 0.980)
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 > world_config.worldSize_x * tileSize -
        world_config.worldDisplay_x * tileSize - 1 then
        camera.pos_x = world_config.worldSize_x * tileSize -
            world_config.worldDisplay_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, world_config.worldDisplay_y, 1 do
    for x=1, world_config.worldDisplay_x, 1 do
      if (world[y][first_tile_x + x] == "G") then
        love.graphics.draw(tilesetImage, tileQuads[1],
          ((x - 1)*tileSize) - offset_x , ((y-1)*tileSize))
      elseif (world[y][first_tile_x + x] == "T") then
        love.graphics.draw(tilesetImage, tileQuads[4],
          ((x-1)*tileSize) - offset_x , ((y-1)*tileSize))
      elseif (world[y][first_tile_x + x] == "A") then
        love.graphics.draw(tilesetImage, tileQuads[7],
          ((x-1)*tileSize) - offset_x , ((y-1)*tileSize))
      elseif (world[y][first_tile_x + x] == "P") then
        love.graphics.draw(tilesetImage, tileQuads[8],
          ((x-1)*tileSize) - offset_x , ((y-1)*tileSize))
      elseif (world[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
```

Tile-Based Scrolling – Example

