

Programming Fundamentals

Lecture 03 – Introduction to Löve 2D

Edirlei Soares de Lima

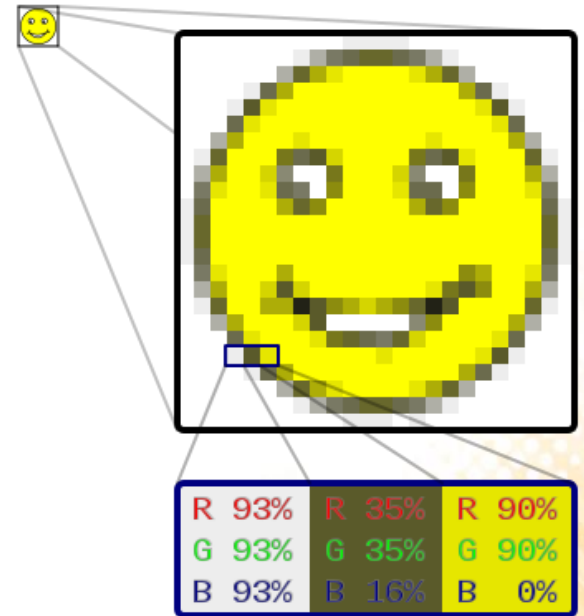
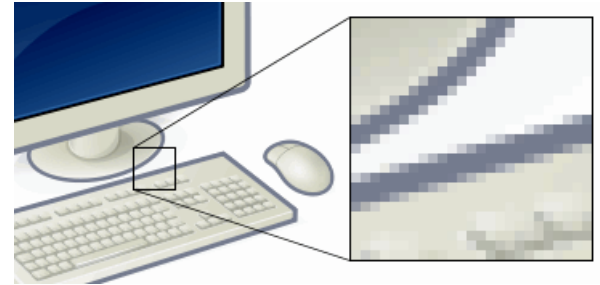
<edirlei.lima@universidadeeuropeia.pt>



Computer Graphics Concepts

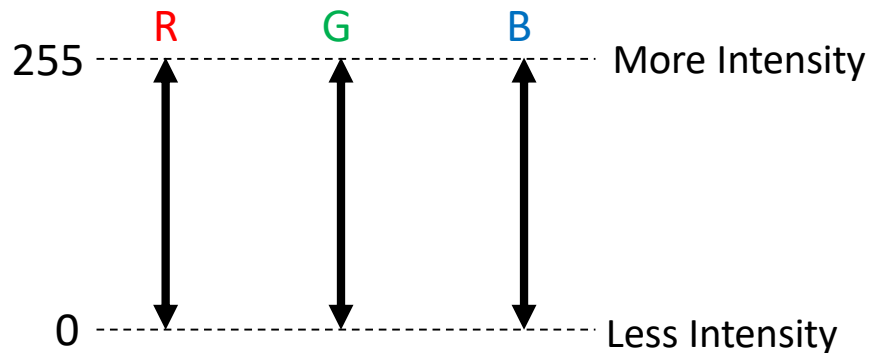
- **What is a pixel?**

- In digital imaging, a pixel is a single square or rectangle point in a raster image (or the smallest addressable element in a display device).
- Pixels are placed in a grid-like fashion and together they draw images on screen.
- The location of a pixel is usually referred by its position x on the horizontal axis and y on the vertical axis of the grid (pixel coordinates).

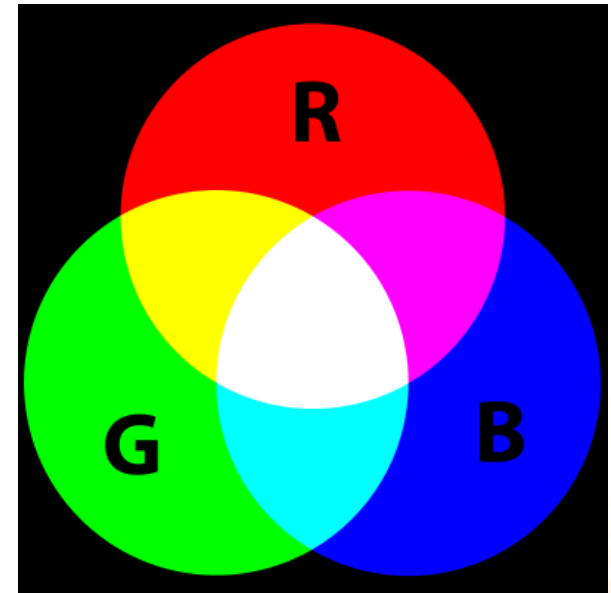


Computer Graphics Concepts

- In computer graphics, **colors** are generally defined by the intensity (chromaticity) of three additive primaries color (or channels): red, green, and blue.
- **RGB Scale:**



Important: the RGB scale used by Löve is between 0 and 1.

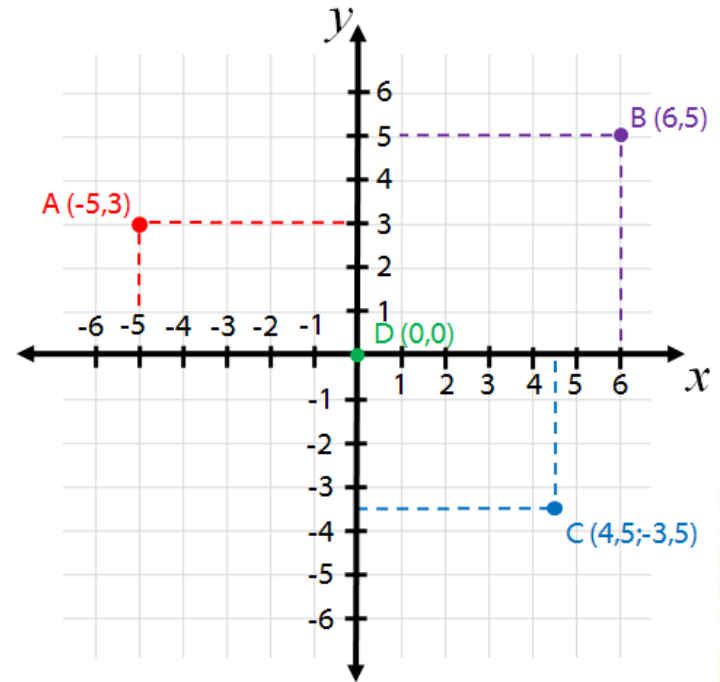


Don't know the RGB value of the color that you want?

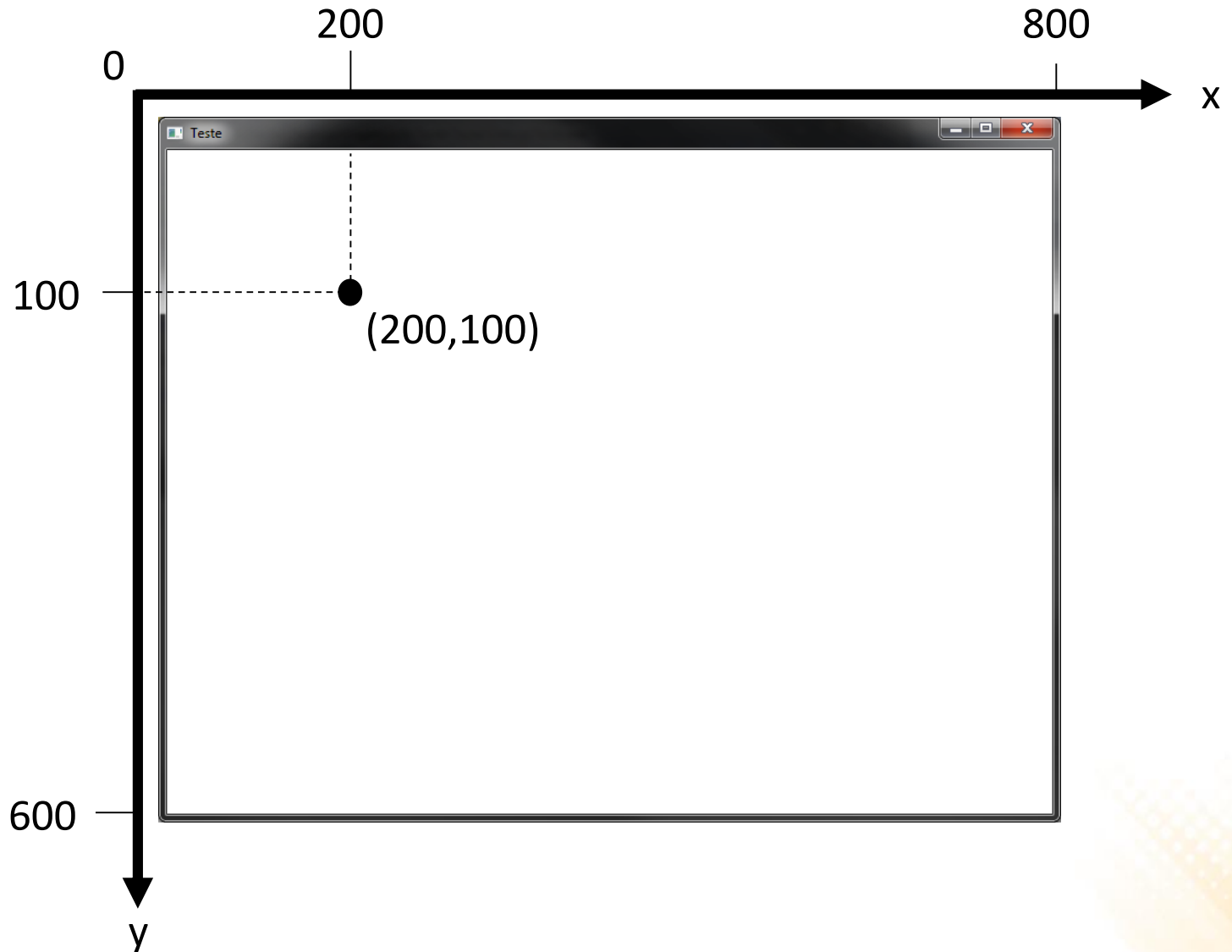
http://doc.instantreality.org/tools/color_calculator/

Computer Graphics Concepts

- Computer graphics uses **coordinate systems** to represent positions on a virtual scene.
- There are usually:
 - 2 axis to define a 2D space (x and y);
 - 3 axis to define a 3D space (x, y, and z);
- Warning: different tools/frameworks use different coordinate systems
 - 1 unit = 1 pixel / arbitrary scene units;
 - Origin at the top left / origin at the center / origin at the bottom left;
 - y goes up / y goes down;

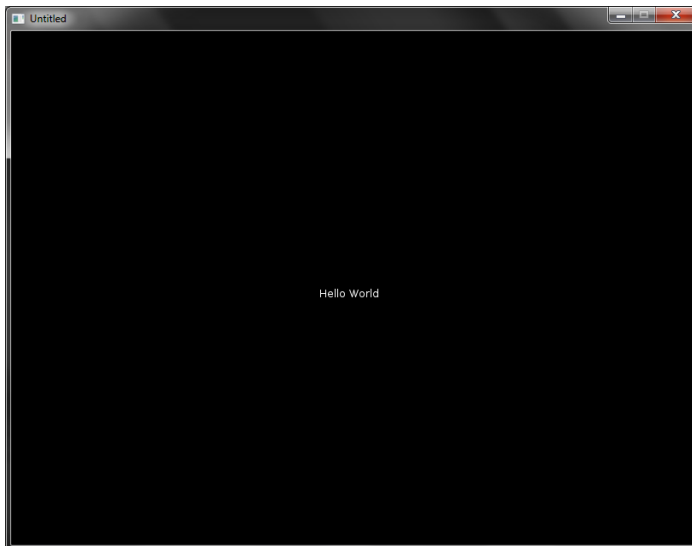


Löve 2D Coordinate System



“Hello World” in Löve

```
function love.draw()  
  love.graphics.print("Hello World", 360, 300)  
end
```



The function `love.graphics.print` is used to draw a text on screen. The last two parameters represent the position (x and y) where the text will be drawn.

Programming in Löve

- Programming in Löve involves the implementation of **callback** functions. A callback is a function that you code and Löve automatically calls at certain times.
- **Example:**

```
love.draw()
```

- The callback `love.draw` is called continuously to draw all the graphical elements (images, geometric shapes, text, etc.) on the screen every frame.

Löve Callbacks

- Löve has several callbacks to perform various tasks (all of them are optional):
 - Initialization, rendering, update, user input keyboard/mouse/joystick, ...
- A fully-featured game experience would probably utilize nearly all of Löve callbacks, so it's wise to know what they are.
 - List of Löve callbacks: <https://www.love2d.org/wiki/Category:Callbacks>
- More common callbacks:
 - `love.load()`
 - `love.draw()`
 - `love.update(dt)`

Callback `love.load()`

- The callback `love.load()` is called exactly once at the beginning of the game.
- Is usually used to:
 - Load resources (images, audio, etc.)
 - Initialize variables
 - Set specific settings

```
function love.load()  
    image = love.graphics.newImage("cake.jpg")  
    love.graphics.setColor(0, 0, 0)  
    love.graphics.setNewFont(12)  
    love.graphics.setBackgroundColor(255, 255, 255)  
end
```

Back to the “Hello World”

```
function love.load()  
    love.graphics.setColor(0, 0, 0)  
    love.graphics.setBackgroundColor(1, 1, 1)  
end  
  
function love.draw()  
    love.graphics.print("Hello World", 360, 300)  
end
```



The function `love.graphics.setColor` defines the color used to draw things on screen (RGB model)

The function `love.graphics.setBackgroundColor` defines the background color (RGB model)

Callback `love.update(dt)`

- The callback `love.update(dt)` is called continuously while the game is running (every frame). The parameter 'dt' stands for "delta time" and it represents amount of seconds since the last time this function was called (usually a small value like 0.02571).
- Is usually used to:
 - Implementation of the game logic
 - Physics simulations
 - Artificial intelligence computations

Calculates the value of `px` at a constant rate (independently of the speed of the computer)

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

Back to the “Hello World”

```
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)
end

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

Löve Modules

- Löve comprises several **modules**:
 - Every module has a set of functions and data types that can be used for game programming.
 - All modules are contained in a global module called `love`.
- **Example of module:** `love.graphics`
 - In the previous examples we used some functions from the `love.graphics` module.
 - The function `love.graphics.print` is part of the `love.graphics` module.
- List of Löve modules: <https://love2d.org/wiki/love>

Module `love.graphics`

- The `love.graphics` module contains functions dedicated for graphical operations:
 - Draw lines, geometric shapes, text, images, etc.
 - Load external files (images, fonts, etc.) into memory.
 - Create special objects (particle system, canvas, etc.)
 - Manipulate the screen
- A complete list of functions of the `love.graphics` module is available at: <https://love2d.org/wiki/love.graphics>

Module `love.graphics`

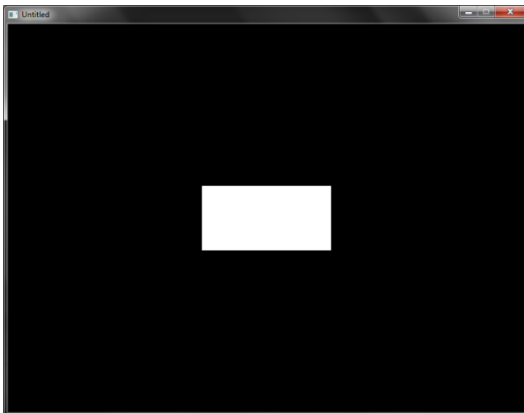
- Drawing basic **geometric shapes**:

- **Rectangle**:

```
love.graphics.rectangle(mode, x, y, width, height)
```

Example:

```
love.graphics.rectangle("fill", 300, 250, 200, 100)
```



mode: "fill" to draw the shape filled
or "line" to draw just an outline.

Module `love.graphics`

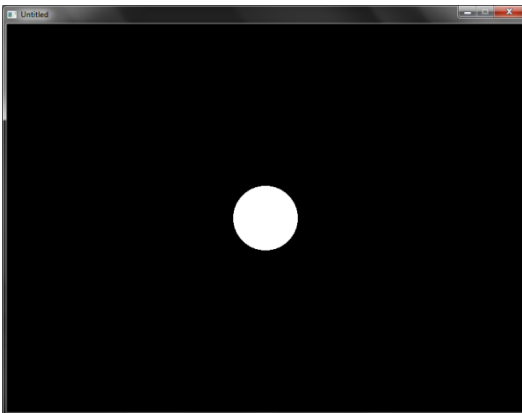
- Drawing basic **geometric shapes**:

- **Circle:**

```
love.graphics.circle(mode, x, y, radius, segments)
```

Example:

```
love.graphics.circle("fill", 400, 300, 50, 100)
```



Number of segments used for drawing the circle

Module `love.graphics`

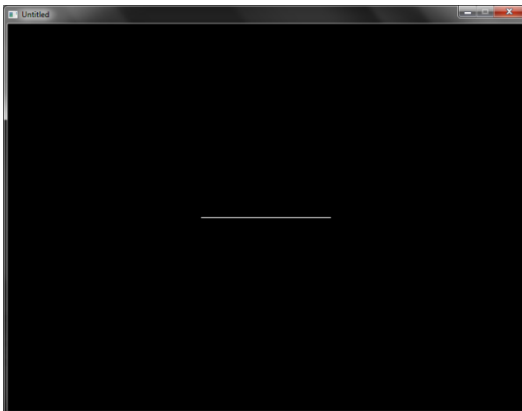
- Drawing basic **geometric shapes**:

- **Line:**

```
love.graphics.line(x1, y1, x2, y2, ...)
```

Example:

```
love.graphics.line(300, 300, 500, 300)
```



More points are accepted as parameters.

Module `love.graphics`

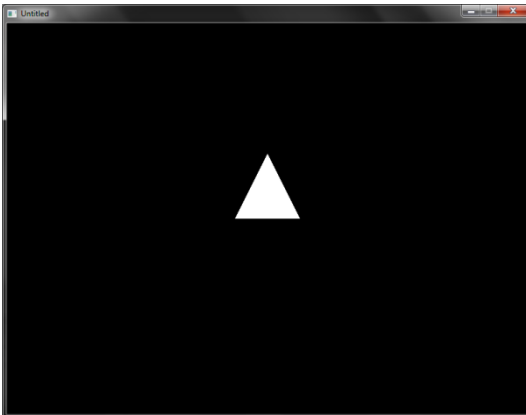
- Drawing basic **geometric shapes**:

- **Polygon**:

```
love.graphics.polygon(mode, ...)
```

Example:

```
love.graphics.polygon("fill", 350, 300, 450, 300, 400, 200)
```



More points are accepted as parameters.

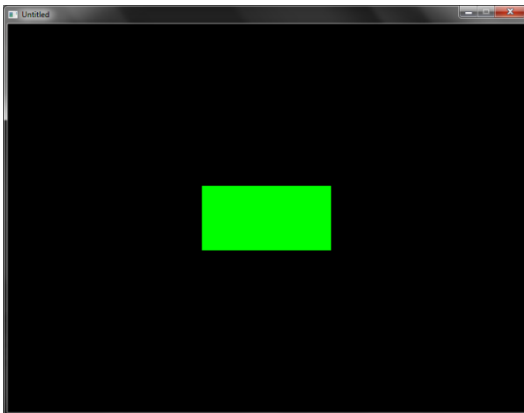
Module `love.graphics`

- Drawing basic **geometric shapes**:
 - **Changing the color of the geometric shapes:**

```
love.graphics.setColor(red, green, blue, alpha)
```

Example:

```
love.graphics.setColor(0, 1, 0)  
love.graphics.rectangle("fill", 300, 250, 200, 100)
```



The alpha is optional and can be used to define colors with transparency.

Geometric Shapes - Example

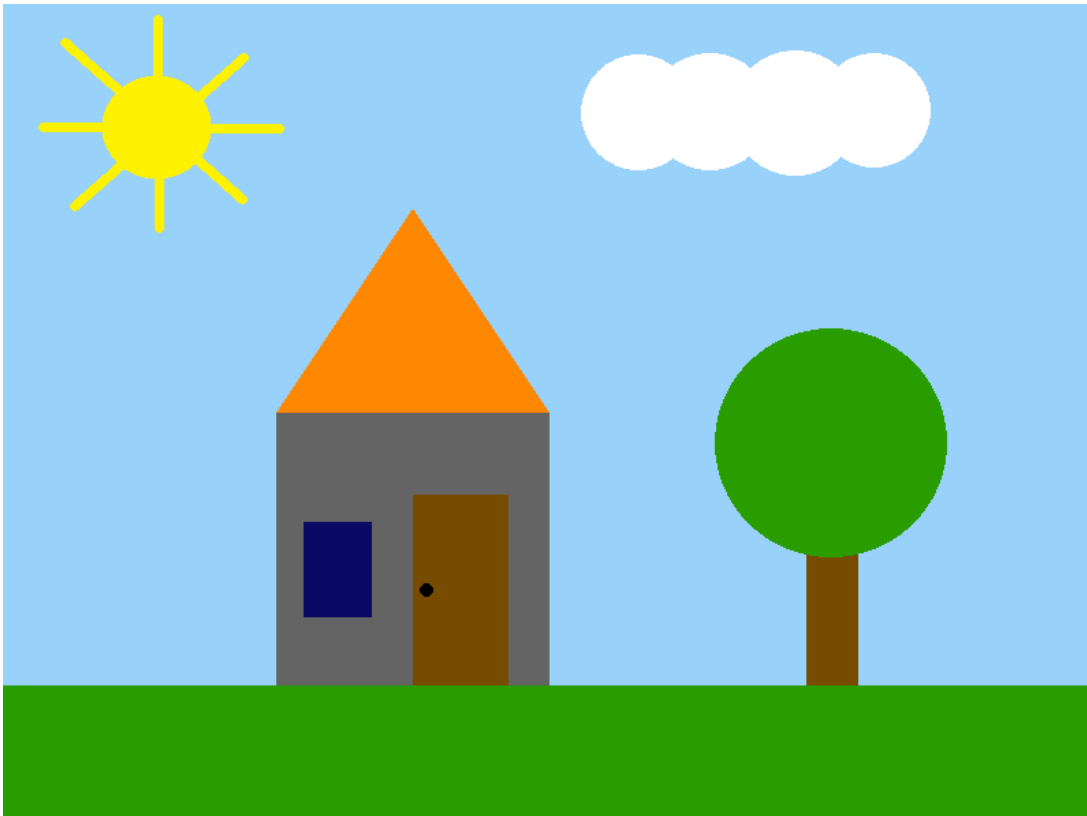
```
function love.draw()  
  
  -- draw a rectangle  
  love.graphics.setColor(0, 0.521, 0)  
  love.graphics.rectangle("fill", 100, 100, 600, 400)  
  
  -- draw a polygon  
  love.graphics.setColor(0.988, 0.988, 0)  
  love.graphics.polygon("fill", 120, 300, 400, 120,  
                        680, 300, 400, 480)  
  
  -- draw a circle  
  love.graphics.setColor(0, 0, 0.552)  
  love.graphics.circle("fill", 400, 300, 120, 100)  
  
end
```

Geometric Shapes - Example



Exercise 1

- 1) Using basic geometric shapes (lines, rectangles, circles, and polygons), implement a program to draw a scene similar to the one illustrated below:



Hints:

- Start simple and add one element at a time.
- Test after adding each element.
- Try to create a function for each element.

Extra challenge:

- Draw a character in the scene using basic shapes:

