MINI PROJECT
(2020-21)

BIRDY

MID-TERM REPORT

Institute of Engineering & Technology

Submitted by
Abhishek Singh
(181500028)

Supervised By: -

Mr. Piyush Vashisth
Technical Trainer
Department of Computer Engineering & Applications

Contents

Abstract

In this project I am going to build a 2d animated game called birdy which is a type of flappy bird game developed by Dong Ngyuyen which is a background scroll based game.

Introduction

1.1 History

The history of game making begins with the development of the first video games, although which video game is the first depends on the definition of *video game*. The first games created had little entertainment value, and their development focus was separate from user experience in fact, these games required mainframe computers to play them. *OXO*, written by Alexander S. Douglas in 1952, was the first computer game to use a digital display. In 1958, a game called *Tennis for Two*, which displayed its output on an oscilloscope, was made by Willy Higinbotham, a physicist working at the Brookhaven National Laboratory. In 1961, a mainframe computer game called *Spacewar!*was developed by a group of Massachusetts Institute of Technology students led by Steve Russell.

True commercial design and development of games began in the 1970s, when arcade video games and first-generation consoles were marketed. In 1971, *Computer Space* was the first commercially sold, coin-operated video game. It used a black-and-white television for its display, and the computer system was made of 74 seriesTTLchips. In 1972, the first home console system was released called Magnavox Odyssey, developed by Ralph H. Bity. The commercial success of *Pong* led other companies to develop *Pong* clones, spawning the video game industry.

Programmers workedaer. That same year, Atari released *Pong*, an arcade game that increased video game popular within the big companies to produce games for these devices. The industry did not see huge innovation in game design and a large number of consoles had very similar games. Many of these early games were often *Pong* clones. Some games were different, however, such as *Gun Fight*, which was significant for several reasons: an early 1975 on-foot, multi-directional shooter, which depicted game characters, game violence, and human-to-human combat.Tomohiro Nishikado's original version was based on discrete logic, which Dave Nutting adapted using the Intel 8080, making it the first video game to use a microprocessor. Console manufacturers soon started to produce consoles that were able to play independently developed games,and ran on microprocessors, marking the beginning of second-generation consoles, beginning with the release of the Fairchild Channel F in 1976.

The flood of *Pong* clones led to the video game crash of 1977, which eventually came to an end with the mainstream success of Taito's 1978 arcade shooter game*Space Invaders*, marking the beginning of the golden age of arcade video games and inspiring dozens of manufacturers to enter the market. Its creator Nishikado not only designed and programmed the game, but also did the artwork, engineered the arcade hardware, and put together a microcomputer from scratch. It was soon ported to the Atari 2600, becoming the first "killer app" and quadrupling the console's sales. At the same time, home computers appeared on the market, allowing individual programmers and hobbyists to develop games. This allowed hardware manufacturer and software manufacturers to act separately. A very large amount of games could be produced by single individuals, as games were easy to make because graphical and memory limitation did not allow for much content. Larger companies developed, who focused selected teams to work on a title. The developers of many early home video games, such as *Zork*, *Baseball*, *Air Warrior*, and *Adventure*, later transitioned their work as products of the early video game industry.

> “I wouldn't recommend [designing computer games] for someone with a weak heart or a large appetite”

—Jon Freeman, 1984

The industry expanded significantly at the time, with the arcade video game sector alone (representing the largest share of the gaming industry) generating higher revenues than both pop music and Hollywood films combined. The home video game industry, however, suffered major losses following the North American video game crash of 1983. In 1984 Jon Freeman warned in *Computer Gaming World*:

Q: Are computer games the way to fame and fortune?

A: No. Not unless your idea of fame is having your name recognized by one or two astute individuals at Origins ... I've been making a living (after a fashion) designing games for most of the last six years. I wouldn't recommend it for someone with a weak heart or a large appetite, though.

Chris Crawford and Don Daglow in 1987 similarly advised prospective designers to write games as a hobby first, and to not quit their existing jobs early. The home video game industry was revitalized soon after by the widespread success of the Nintendo Entertainment System.

By 1987 a video game required 12 months to develop and another six to plan marketing. Projects remained usually solo efforts, with single developers delivering finished games to their publishers. With the ever-increasing processing and graphical capabilities of arcade, console and computer products, along with an increase in user expectations, game design moved beyond the scope of a single developer to produce a marketable game in a reasonable time. This sparked the beginning of team-based development.In broad terms, during the 1980s,

pre-production involved sketches and test routines of the only developer. In the 1990s, pre-production consisted mostly of game art previews. In the early 2000s, pre-production usually produced a playable demo.

1.2 General Introduction to the topic

Flappy Bird is a mobile game developed by Vietnamese video game artist and programmer Dong Nguyen (Vietnamese: *Nguyễn Hà Đông*), under his game development company dotGears.[1] The game is a side-scroller where the player controls a bird, attempting to fly between columns of green pipes without hitting them. Nguyen created the game over the period of several days, using a bird protagonist that he had designed for a cancelled game in 2012.

The game was released in May 2013 but received a sudden rise in popularity in early 2014. *Flappy Bird* received poor reviews from some critics, who criticized its high level of difficulty, alleged plagiarism in graphics and game mechanics, while other reviewers found it addictive. At the end of January 2014, it was the most downloaded free game in the App Store for iOS. During this period, its developer said that *Flappy Bird* was earning $50,000 a day from in-app advertisements as well as sales.

Flappy Bird was removed from both the App Store and Google Play by its creator on February 10, 2014. He claims that he felt guilt over what he considered to be its addictive nature and overuse. The game's popularity and sudden removal caused phones with it pre-installed to be put up for sale for high prices over the Internet.[2][3][4] Games similar to *Flappy Bird* became popular on the iTunes App Store in the wake of its removal, and both Apple and Google have removed games from their app stores for being too similar to the original.

In August 2014, a revised version of *Flappy Bird*, called *Flappy Birds Family*, was released exclusively for the Amazon Fire TV. Bay Tek Games also released a licensed coin-operated *Flappy Bird* arcade game.[5]

1.3 Gameplay

Faby after passing the first pair of pipes

Flappy Bird is an arcade-style game in which the player controls the bird Faby, which moves persistently to the right. The player is tasked with navigating Faby through pairs of pipes that have equally sized gaps placed at random heights. Faby automatically descends and only ascends when the player taps the touchscreen. Each successful pass through a pair of pipes awards the player one point. Colliding with a pipe or the ground ends the gameplay. During the game over screen, the player is awarded a bronze medal if they reached ten or more points, a silver medal from twenty points, a gold medal from thirty points, and a platinum medal from forty points

1.4 Hardware Requirements

- Memory [4GB RAM (or higher)]
- Intel core i3 64-bit Processor (or higher)

1.4 Software requirements

- Love 2D
- Sublime text3(or any text editor that support language LUA)

Objective

The purpose of the project is to design and implement a 2-dimensional game written in LUA using gaming library of love 2d. The level will include everything that should be available in an arcade adventure game like the popular Nintendo classic Super Mario game. The game will be a single-player adventure game. The goals of this project is to create an easy to use, pick up and play game that could be played by all ages as long as they have a desktop computer or a laptop pc. The reason was as stated above that they are more gamers playing video games every day meaning a larger potential market.

.

Implementation Details

Part1: first is to download LOVE 2D application on which our game will be test after implementing the game of code in Sublime text 3 which is essentially most important part.
Part 2: we have to create two file called main and push which will be the most important part of the game.

Part 3: now add background,pipe,bird images in the game and implement the coding of bird(by which the player will move the bird).

Part 4: we create pipepair file so that we can see the pipe moving in pair from which our bird pass to advance.

Part 5: anti gravity update in which we see our bird jump when we press space key

Part 6: collision update in which game will over when bird will strike the ground or pipe.

Part 7: Pause and resume update.

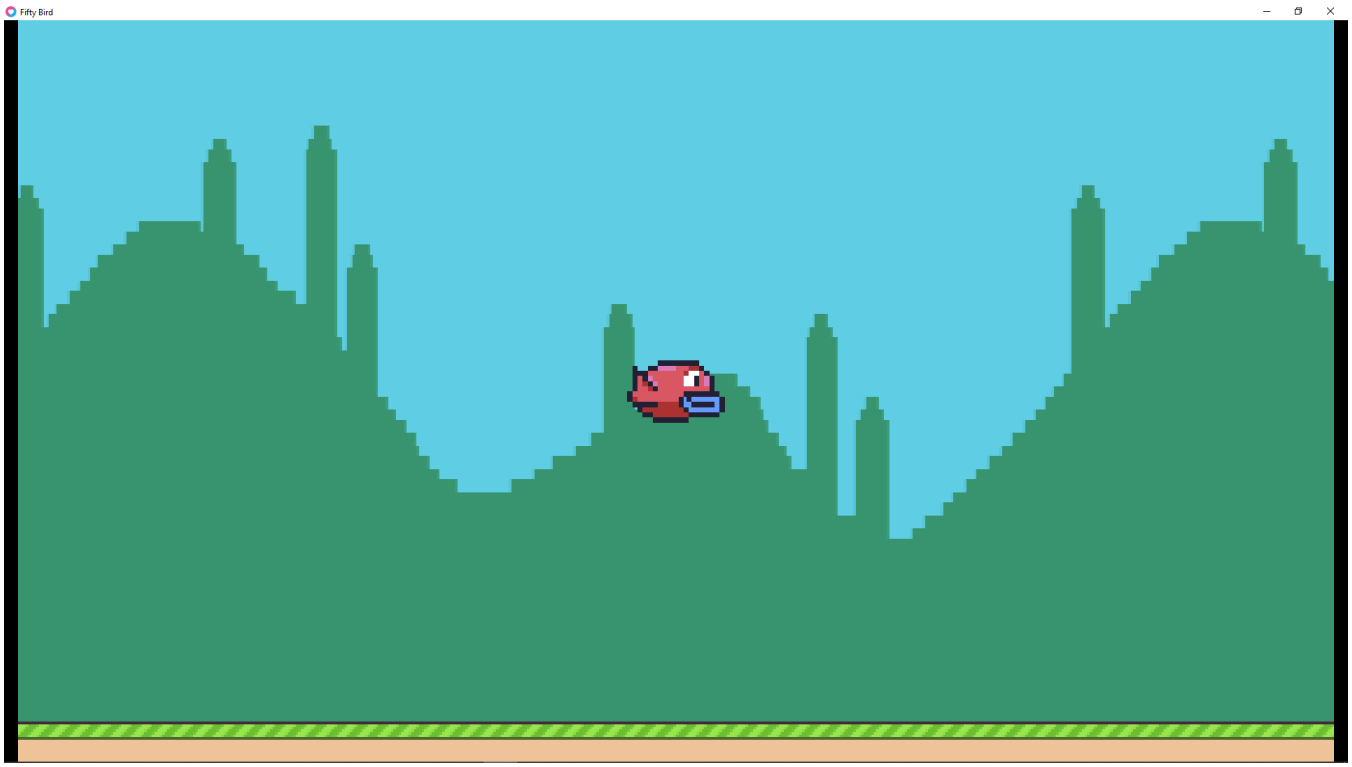
Part 8: Audio update.

Progress

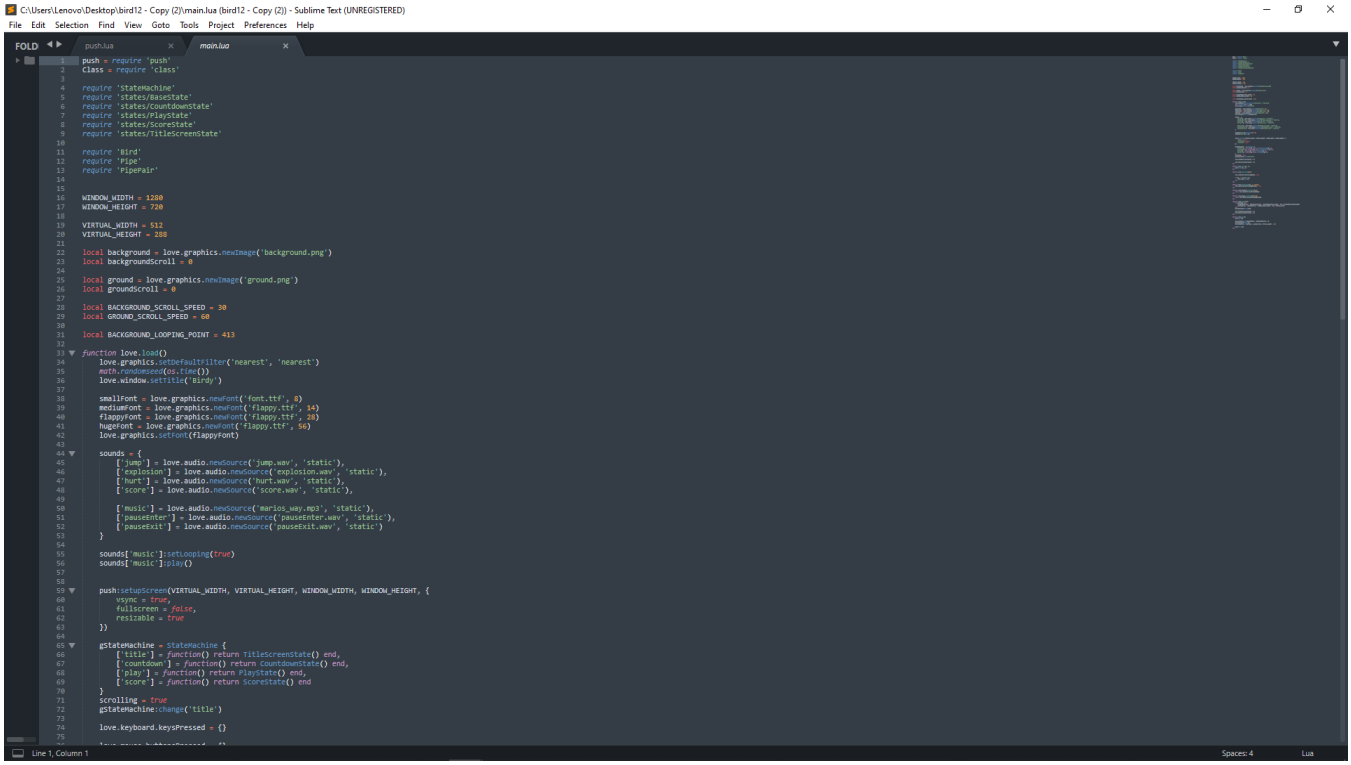
1.) Part 1 is completed
- Love 2d installed
- Sublime text3 installed

2.) Part 2 is completed
- Create a new project
- Create file called main and push

3.) Part 3 is completed
- Add background image(spites)
- Add Pipe image(spites)
- Add bird Image(spites)
- Implement code for bird to move



SCREENSHOTS



C:\Users\Lenovo\Desktop\bird12 - Copy (2)\main.lua (bird12 - Copy (2)) - Sublime Text (UNREGISTERED)

File Edit Selection Find View Goto Tools Project Preferences Help

FOLD push.lua main.lua

```lua
    sounds['music']:setLooping(true)
    sounds['music']:play()


    push:setupScreen(VIRTUAL_WIDTH, VIRTUAL_HEIGHT, WINDOW_WIDTH, WINDOW_HEIGHT, {
        vsync = true,
        fullscreen = false,
        resizable = true
    })

    gStateMachine = StateMachine {
        ['title'] = function() return TitleScreenState() end,
        ['countdown'] = function() return CountdownState() end,
        ['play'] = function() return PlayState() end,
        ['score'] = function() return ScoreState() end
    }
    scrolling = true
    gStateMachine:change('title')

    love.keyboard.keysPressed = {}

    love.mouse.buttonsPressed = {}
end

function love.resize(w, h)
    push:resize(w, h)
end

function love.keypressed(key)

    love.keyboard.keysPressed[key] = true

    if key == 'escape' then
        love.event.quit()
    end
end

function love.mousepressed(x, y, button)
    love.mouse.buttonsPressed[button] = true
end

function love.keyboard.wasPressed(key)
    return love.keyboard.keysPressed[key]
end

function love.mouse.wasPressed(button)
    return love.mouse.buttonsPressed[button]
end

function love.update(dt)
    if scrolling then
        backgroundScroll = (backgroundScroll + BACKGROUND_SCROLL_SPEED * dt) % BACKGROUND_LOOPING_POINT
        groundScroll = (groundScroll + GROUND_SCROLL_SPEED * dt) % VIRTUAL_WIDTH
    end
    gStateMachine:update(dt)

    love.keyboard.keysPressed = {}
    love.mouse.buttonsPressed = {}
end

function love.draw()
    push:start()

    love.graphics.draw(background, -backgroundScroll, 0)
    gStateMachine:render()
    love.graphics.draw(ground, -groundScroll, VIRTUAL_HEIGHT - 16)

    push:finish()
end
```

Line 1, Column 1 Spaces: 4 Lua

C:\Users\Lenovo\Desktop\bird12 - Copy (2)\push.lua (bird12 - Copy (2)) - Sublime Text (UNREGISTERED)

File Edit Selection Find View Goto Tools Project Preferences Help

FOLD push.lua main.lua

```lua
local push = {

  defaults = {
    fullscreen = false,
    resizable = false,
    pixelperfect = false,
    highdpi = true,
    canvas = true
  }

}
setmetatable(push, push)


function push:applySettings(settings)
  for k, v in pairs(settings) do
    self["_" .. k] = v
  end
end

function push:resetSettings() return self:applySettings(self.defaults) end

function push:setupScreen(WWIDTH, WHEIGHT, RWIDTH, RHEIGHT, settings)

  settings = settings or {}

  self._WWIDTH, self._WHEIGHT = WWIDTH, WHEIGHT
  self._RWIDTH, self._RHEIGHT = RWIDTH, RHEIGHT

  self:applySettings(self.defaults)
  self:applySettings(settings)

  love.window.setMode( self._RWIDTH, self._RHEIGHT, {
    fullscreen = self._fullscreen,
    resizable = self._resizable,
    highdpi = self._highdpi
  } )

  self:initValues()

  if self._canvas then
    self:setupCanvas({ "default" })
  end

  self._borderColor = {0, 0, 0}

  self._drawFunctions = {
    ["start"] = self.start,
    ["end"] = self.finish
  }

  return self
end

function push:setupCanvas(canvases)
  table.insert(canvases, { name = "_render" })

  self._canvas = true
  self.canvases = {}

  for i = 1, #canvases do
    self.canvases[i] = {
      name = canvases[i].name,
      shader = canvases[i].shader,
      canvas = love.graphics.newCanvas(self._WWIDTH, self._WHEIGHT)
    }
  end

  return self
end

function push:setCanvas(name)
  if not self._canvas then return true end
  return love.graphics.setCanvas( self:getCanvasTable(name).canvas )
end
```

Line 82, Column 4 Spaces: 2 Lua

C:\Users\Lenovo\Desktop\bird12 - Copy (2)\push.lua (bird12 - Copy (2)) - Sublime Text (UNREGISTERED)

File Edit Selection Find View Goto Tools Project Preferences Help

FOLD push.lua main.lua

```lua
    self.canvases[i] = {
      name = canvases[i].name,
      shader = canvases[i].shader,
      canvas = love.graphics.newCanvas(self._WWIDTH, self._WHEIGHT)
    }
  end

  return self
end

function push:setCanvas(name)
  if not self._canvas then return true end
  return love.graphics.setCanvas( self:getCanvasTable(name).canvas )
end
function push:getCanvasTable(name)
  for i = 1, #self.canvases do
    if self.canvases[i].name == name then
      return self.canvases[i]
    end
  end
end
function push:setShader(name, shader)
  if not shader then
    self:getCanvasTable("_render").shader = name
  else
    self:getCanvasTable(name).shader = shader
  end
end

function push:initValues()
  self._PSCALE = self._highdpi and love.window.getDPIScale() or 1

  self._SCALE = {
    x = self._RWIDTH/self._WWIDTH * self._PSCALE,
    y = self._RHEIGHT/self._WHEIGHT * self._PSCALE
  }

  if self._stretched then
    self._OFFSET = {x = 0, y = 0}
  else
    local scale = math.min(self._SCALE.X, self._SCALE.y)
    if self._pixelperfect then scale = math.floor(scale) end

    self._OFFSET = {x = (self._SCALE.X - scale) * (self._WWIDTH/2), y = (self._SCALE.y - scale) * (self._WHEIGHT/2)}
    self._SCALE.x, self._SCALE.y = scale, scale --apply same scale to X and Y
  end

  self._GWIDTH = self._RWIDTH * self._PSCALE - self._OFFSET.X * 2
  self._GHEIGHT = self._RHEIGHT * self._PSCALE - self._OFFSET.y * 2
end

--[[ DEPRECATED ]]--
function push:apply(operation, shader)
  if operation == "start" then
    self:start()
  elseif operation == "finish" or operation == "end" then
    self:finish(shader)
  end
end

function push:start()
  if self._canvas then
    love.graphics.push()
    love.graphics.setCanvas(self.canvases[1].canvas)
  else
    love.graphics.translate(self._OFFSET.x, self._OFFSET.y)
    love.graphics.setScissor(self._OFFSET.x, self._OFFSET.y, self._WWIDTH*self._SCALE.x, self._WHEIGHT*self._SCALE.y)
    love.graphics.push()
    love.graphics.scale(self._SCALE.X, self._SCALE.y)
  end
end

function push:finish(shader)
  love.graphics.setBackgroundColor(unpack(self._borderColor))
  if self._canvas then
```

Line 82, Column 4 Spaces: 2 Lua

C:\Users\Lenovo\Desktop\bird12 - Copy (2)\push.lua (bird12 - Copy (2)) - Sublime Text (UNREGISTERED)

File Edit Selection Find View Goto Tools Project Preferences Help

push.lua | main.lua

```lua
    self._SCALE.x, self._SCALE.y = scale, scale --apply same scale to X and Y
  end

  self._GWIDTH = self._RWIDTH * self._PSCALE - self._OFFSET.x * 2
  self._GHEIGHT = self._RHEIGHT * self._PSCALE - self._OFFSET.y * 2
end

--[[ DEPRECATED ]]--
function push:apply(operation, shader)
  if operation == "start" then
    self:start()
  elseif operation == "finish" or operation == "end" then
    self:finish(shader)
  end
end

function push:start()
  if self._canvas then
    love.graphics.push()
    love.graphics.setCanvas(self.canvases[1].canvas)
  else
    love.graphics.translate(self._OFFSET.x, self._OFFSET.y)
    love.graphics.setScissor(self._OFFSET.x, self._OFFSET.y, self._WWIDTH*self._SCALE.x, self._WHEIGHT*self._SCALE.y)
    love.graphics.push()
    love.graphics.scale(self._SCALE.x, self._SCALE.y)
  end
end

function push:finish(shader)
  love.graphics.setBackgroundColor(unpack(self._borderColor))
  if self._canvas then
    local _render = self:getCanvasTable("_render")

    love.graphics.pop()

    love.graphics.setColor(255, 255, 255)

    love.graphics.setCanvas(_render.canvas)
    for i = 1, #self.canvases - 1 do --do not draw _render yet
      local _table = self.canvases[i]
      love.graphics.setShader(_table.shader)
      love.graphics.draw(_table.canvas)
    end
    love.graphics.setCanvas()

    love.graphics.translate(self._OFFSET.x, self._OFFSET.y)
    love.graphics.setShader(shader or self:getCanvasTable("_render").shader)
    love.graphics.draw(self:getCanvasTable("_render").canvas, 0, 0, 0, self._SCALE.x, self._SCALE.y)

    for i = 1, #self.canvases do
      love.graphics.setCanvas( self.canvases[i].canvas )
      love.graphics.clear()
    end

    love.graphics.setCanvas()
    love.graphics.setShader()
  else
    love.graphics.pop()
    love.graphics.setScissor()
  end
end

function push:setBorderColor(color, g, b)
  self._borderColor = g and {color, g, b} or color
end

function push:toGame(x, y)
  x, y = x - self._OFFSET.x, y - self._OFFSET.y
  local normalX, normalY = x / self._GWIDTH, y / self._GHEIGHT

  x = (x >= 0 and x <= self._WWIDTH * self._SCALE.x) and normalX * self._WWIDTH or nil
  y = (y >= 0 and y <= self._WHEIGHT * self._SCALE.y) and normalY * self._WHEIGHT or nil

  return x, y
end
```

Line 82, Column 4 Spaces: 2 Lua

C:\Users\Lenovo\Desktop\bird12 - Copy (2)\push.lua (bird12 - Copy (2)) - Sublime Text (UNREGISTERED)

File Edit Selection Find View Goto Tools Project Preferences Help

push.lua | main.lua

```lua

function push:setBorderColor(color, g, b)
  self._borderColor = g and {color, g, b} or color
end

function push:toGame(x, y)
  x, y = x - self._OFFSET.x, y - self._OFFSET.y
  local normalX, normalY = x / self._GWIDTH, y / self._GHEIGHT

  x = (x >= 0 and x <= self._WWIDTH * self._SCALE.x) and normalX * self._WWIDTH or nil
  y = (y >= 0 and y <= self._WHEIGHT * self._SCALE.y) and normalY * self._WHEIGHT or nil

  return x, y
end

--doesn't work - TODO
function push:toReal(x, y)
  return x+self._OFFSET.x, y+self._OFFSET.y
end

function push:switchFullscreen(winw, winh)
  self._fullscreen = not self._fullscreen
  local windowWidth, windowHeight = love.window.getDesktopDimensions()

  if self._fullscreen then --save windowed dimensions for later
    self._WINWIDTH, self._WINHEIGHT = self._RWIDTH, self._RHEIGHT
  elseif not self._WINWIDTH or not self._WINHEIGHT then
    self._WINWIDTH, self._WINHEIGHT = windowWidth * .5, windowHeight * .5
  end

  self._RWIDTH = self._fullscreen and windowWidth or winw or self._WINWIDTH
  self._RHEIGHT = self._fullscreen and windowHeight or winh or self._WINHEIGHT

  self:initValues()

  love.window.setFullscreen(self._fullscreen, "desktop")
  if not self._fullscreen and (winw or winh) then
    love.window.setMode(self._RWIDTH, self._RHEIGHT) --set window dimensions
  end
end

function push:resize(w, h)
  local pixelScale = love.window.getDPIScale()
  if self._highdpi then w, h = w / pixelScale, h / pixelScale end
  self._RWIDTH = w
  self._RHEIGHT = h
  self:initValues()
end

function push:getWidth() return self._WWIDTH end
function push:getHeight() return self._WHEIGHT end
function push:getDimensions() return self._WWIDTH, self._WHEIGHT end

return push

```

Line 82, Column 4 Spaces: 2 Lua

C:\Users\Lenovo\Desktop\bird12 - Copy (2)\class.lua (bird12 - Copy (2)) - Sublime Text (UNREGISTERED)

File Edit Selection Find View Goto Tools Project Preferences Help

push.lua | class.lua

```lua
local function include_helper(to, from, seen)
    if from == nil then
        return to
    elseif type(from) ~= 'table' then
        return from
    elseif seen[from] then
        return seen[from]
    end

    seen[from] = to
    for k,v in pairs(from) do
        k = include_helper({}, k, seen)
        if to[k] == nil then
            to[k] = include_helper({}, v, seen)
        end
    end
    return to
end

local function include(class, other)
    return include_helper(class, other, {})
end


local function clone(other)
    return setmetatable(include({}, other), getmetatable(other))
end

local function new(class)
    -- mixins
    class = class or {}
    local inc = class.__includes or {}
    if getmetatable(inc) then inc = {inc} end

    for _, other in ipairs(inc) do
        if type(other) == "string" then
            other = _G[other]
        end
        include(class, other)
    end


    class.__index = class
    class.init    = class.init    or class[1] or function() end
    class.include = class.include or include
    class.clone   = class.clone   or clone


    return setmetatable(class, {__call = function(c, ...)
        local o = setmetatable({}, c)
        o:init(...)
        return o
    end})
end

if class_commons ~= false and not common then
    common = {}
    function common.class(name, prototype, parent)
        return new{__includes = {prototype, parent}}
    end
    function common.instance(class, ...)
        return class(...)
    end
end

return setmetatable({new = new, include = include, clone = clone},
    {__call = function(_,...) return new(...) end})
```

Line 9, Column 1 | Tab Size: 4 | Lua

C:\Users\Lenovo\Desktop\bird12 - Copy (2)\Bird.lua (bird12 - Copy (2)) - Sublime Text (UNREGISTERED)

File Edit Selection Find View Goto Tools Project Preferences Help

push.lua | Bird.lua

```lua
Bird = Class{}

local GRAVITY = 25

function Bird:init()
    self.image = love.graphics.newImage('bird.png')
    self.x = VIRTUAL_WIDTH / 2 - 8
    self.y = VIRTUAL_HEIGHT / 2 - 8

    self.width = self.image:getWidth()
    self.height = self.image:getHeight()

    self.dy = 0
end

function Bird:collides(pipe)

    if (self.x + 2) + (self.width - 4) >= pipe.x and self.x + 2 <= pipe.x + PIPE_WIDTH then
        if (self.y + 2) + (self.height - 4) >= pipe.y and self.y + 2 <= pipe.y + PIPE_HEIGHT then
            return true
        end
    end

    return false
end

function Bird:update(dt)
    self.dy = self.dy + GRAVITY * dt

    if love.keyboard.wasPressed('space') or love.mouse.wasPressed(1) then
        self.dy = -5
        sounds['jump']:play()
    end

    self.y = self.y + self.dy
end

function Bird:render()
    love.graphics.draw(self.image, self.x, self.y)
end
```

Line 1, Column 1 | Spaces: 4 | Lua

References

https://www.lua.org/

https://www.tutorialspoint.com/lua/index.htm

https://love2d.org/

https://www.sublimetext.com/3

https://www.wikipedia.org/