MiniOmega
Farmgeon

By Barnabáš Bernášek C3c

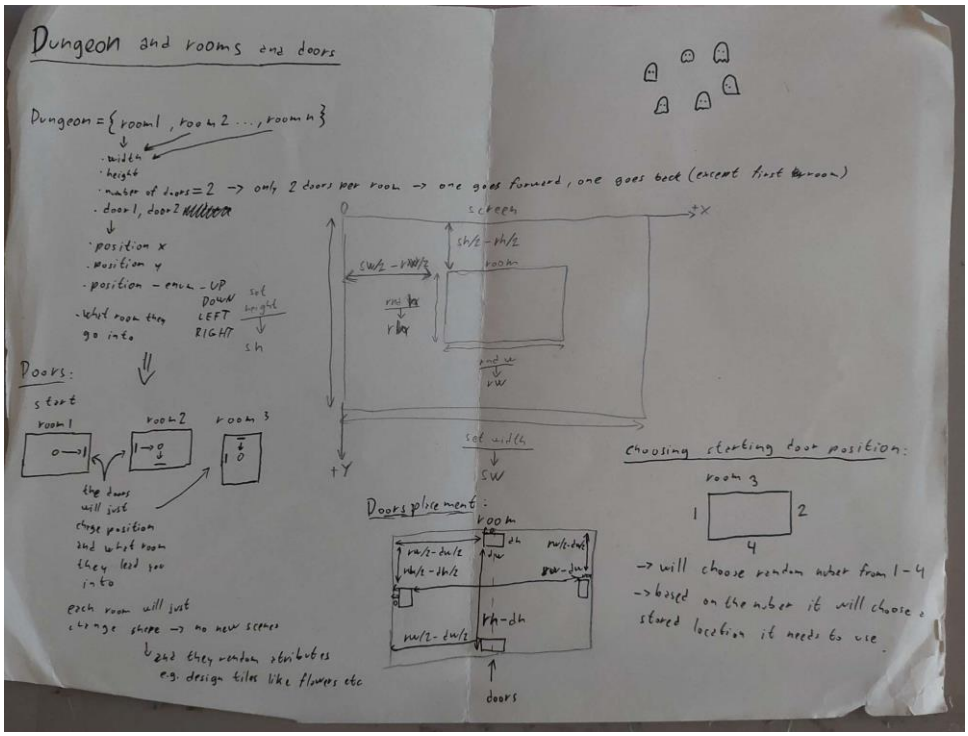
Ideas

First idea for this game came to me while playing certain farming game – Stardew Valley, I then decided to also make farming game, but I wanted to make it more original then just plain old boring farmin.

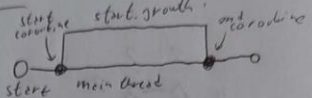
So I mixed two unusual game genres togather – already mentioned farming and rogue-like. Most of the times the main goal of farming games is to grow your own crops, on a field/farm. Rogue-like games are more action based, the basic principals of rogue-like game are some character growth – you go, you die, you gain experience/more money/skills, something that will boost you to another stage of the game, and randomly generated loot spawns/dungeons.

I mixed these two genres to create a MiniOmega Farmgeon – a game about farming in a dungeon, you as a player has to keep your field healthy and untouched so you can collect and sell your grown crops while waves of enemies rush at you with intention to hurt you and your precious crops.

Here is my brainstorming notes, how I made certain aspects of the game, and just my general notes:

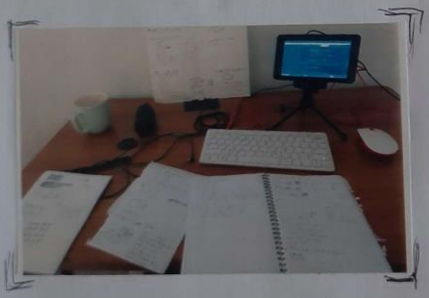


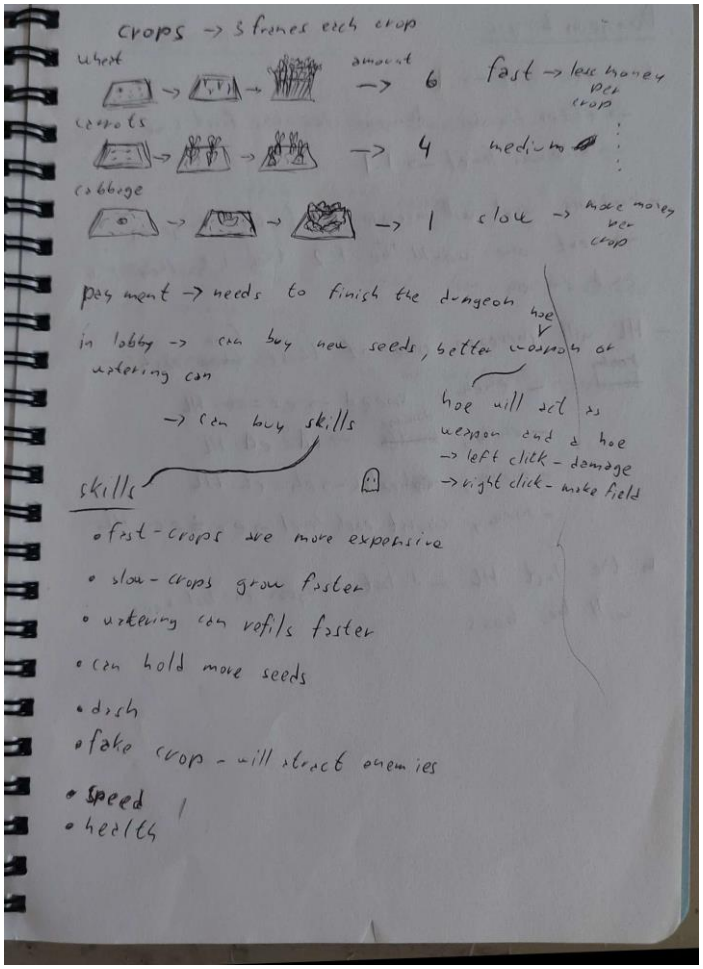
Coroutines in crops



→ plant seed
↓
start coroutine
↓after phase time
seed phase + 1 ← change texture of field
yield coroutine

→ the field is dry stop phase is 2
→ after watering the field
→ restart coroutine




crops -> 3 frames each crop
wheat
amount
-> 6
fast -> less money per crop
carrots
-> 4
medium
cabbage
-> 1
slow -> more money per crop
payment -> needs to finish the dungeon
in lobby -> can buy new seeds, better weapons or watering can
-> can buy skills
skills
hoe will set as weapon and a hoe
-> left click - damage
-> right click - make field
• fast - crops are more expensive
• slow - crops grow faster
• watering can refils faster
• can hold more seeds
• dash
• fake crop - will attract enemies
• speed
• health

Dungeon Logic

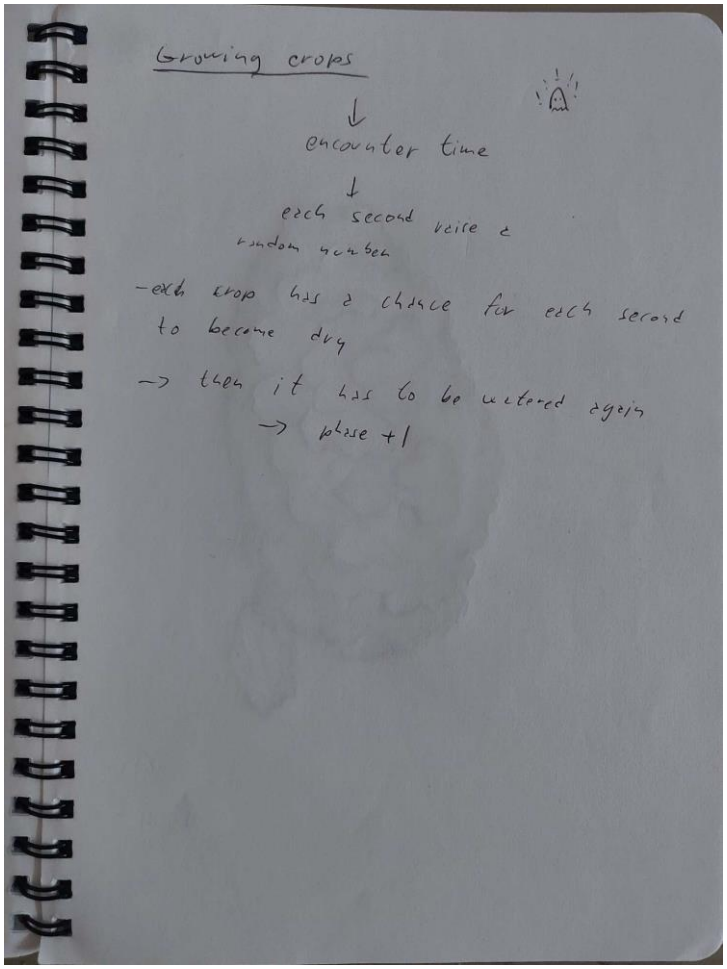
hardness level -> 1

-> after finishes dungeon for the first time

-> hardness level -> 1.1

- hardness level will increase by finished dungeons
  -> next one would be 1.2 then 1.3 then 1.4 and so on
- HL will increase one of these ~~three~~ four each ~~room level~~ :
  - enemy speed -> es = es · HL
  - enemy ~~health~~ damage -> ed = ed · HL
  - enemy health -> eh = eh · HL
  - enemy count each level -> ec ≐ ec · HL

in the last HL and last dungeon in last room will be boss



Game

The gam, as it is now, isn't fully finished. I have implemented some basic functions.

The game doesn't have end yet.

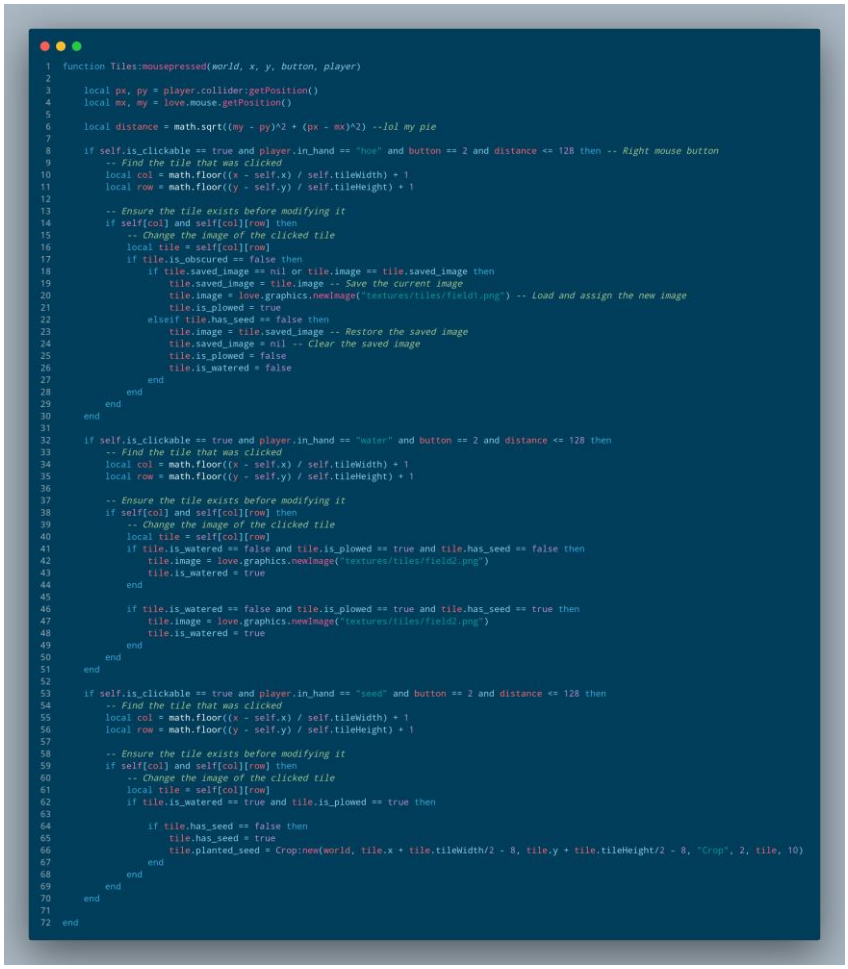
Code

As for the code and programming, I made Farmgeon using Lua programming language, main reason behind it is that Lua is very lightweight and so I could develop my game even on my Raspberry Pi 4B while being on practise in Brno. This decision was a double edged sword for me, since I did't know anything about Lua before starting this project. But I quickly addapted, mainly thanks to my knowladge of Python. Due to Lua being such light language, developing any project in playin Lua would be very hard, that's why I've decided to use LOVE2D library for Lua. LOVE is a 2D engine written in Lua, which allows developer using physics, audio/animation controll, and overall adds a basic tools for game development od 2D games in Lua.

Because later in my development I encountered a problem with collision detection, I've decided to use windfield wrapper for LOVE, windfield makes collision manipulation easier in LOVE.

With the help of LOVE engine and windfield wrapper I’ve started developing my game, but still my knowladge of the language and the engine was limited so I was using internet help extensively.

A few main functions:

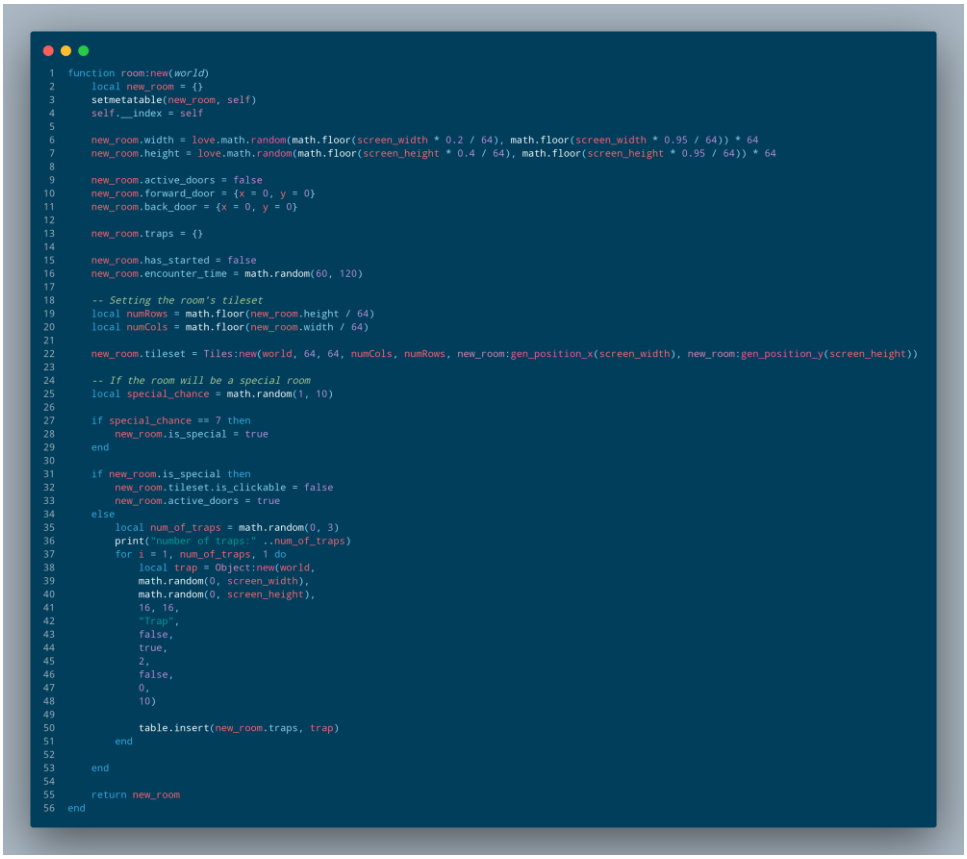


```
function Tiles:mousepressed(world, x, y, button, player)

    local px, py = player.collider:getPosition()
    local mx, my = love.mouse.getPosition()

    local distance = math.sqrt((my - py)^2 + (px - mx)^2) --lol my pie

    if self.is_clickable == true and player.in_hand == "hoe" and button == 2 and distance <= 128 then -- Right mouse button
        -- Find the tile that was clicked
        local col = math.floor((x - self.x) / self.tileWidth) + 1
        local row = math.floor((y - self.y) / self.tileHeight) + 1

        -- Ensure the tile exists before modifying it
        if self[col] and self[col][row] then
            -- Change the image of the clicked tile
            local tile = self[col][row]
            if tile.is_obscured == false then
                if tile.saved_image == nil or tile.image == tile.saved_image then
                    tile.saved_image = tile.image -- Save the current image
                    tile.image = love.graphics.newImage("textures/tiles/field1.png") -- Load and assign the new image
                    tile.is_plowed = true
                elseif tile.has_seed == false then
                    tile.image = tile.saved_image -- Restore the saved image
                    tile.saved_image = nil -- Clear the saved image
                    tile.is_plowed = false
                    tile.is_watered = false
                end
            end
        end
    end

    if self.is_clickable == true and player.in_hand == "water" and button == 2 and distance <= 128 then
        -- Find the tile that was clicked
        local col = math.floor((x - self.x) / self.tileWidth) + 1
        local row = math.floor((y - self.y) / self.tileHeight) + 1

        -- Ensure the tile exists before modifying it
        if self[col] and self[col][row] then
            -- Change the image of the clicked tile
            local tile = self[col][row]
            if tile.is_watered == false and tile.is_plowed == true and tile.has_seed == false then
                tile.image = love.graphics.newImage("textures/tiles/field2.png")
                tile.is_watered = true
            end

            if tile.is_watered == false and tile.is_plowed == true and tile.has_seed == true then
                tile.image = love.graphics.newImage("textures/tiles/field2.png")
                tile.is_watered = true
            end
        end
    end

    if self.is_clickable == true and player.in_hand == "seed" and button == 2 and distance <= 128 then
        -- Find the tile that was clicked
        local col = math.floor((x - self.x) / self.tileWidth) + 1
        local row = math.floor((y - self.y) / self.tileHeight) + 1

        -- Ensure the tile exists before modifying it
        if self[col] and self[col][row] then
            -- Change the image of the clicked tile
            local tile = self[col][row]
            if tile.is_watered == true and tile.is_plowed == true then

                if tile.has_seed == false then
                    tile.has_seed = true
                    tile.planted_seed = Crop:new(world, tile.x + tile.tileWidth/2 - 8, tile.y + tile.tileHeight/2 - 8, "Crop", 2, tile, 10)
                end
            end
        end
    end

end
```

This is a function Tile:mousepressed, it’s main purpose is detecting if mouse button has been pressed, and then do coresponding action on tiles. For example, if player is holding a hoe in hand, and the user presses the left mouse button, the player will attack, if the user presses right button, the player will make a field, where crops can grow.

```lua
function Dungeon:new (world)
    self.size = love.math.random(5, 15)
    for i = 1, self.size do
        local new_room = room:new(world)
        table.insert(self.rooms, new_room)
    end

    --adding doors to rooms cycle
    for i = 1, self.size, 1
    do
        local new_door_w = 100
        local new_door_h = 50

        if i == 1 then
            self.rooms[i].forward_door.x = sw/2
            self.rooms[i].forward_door.y = self.rooms[i]:gen_position_y(sh)
            self.rooms[i].back_door.x = 0
            self.rooms[i].back_door.y = 0
        end

        if i ~= 1 and i ~= self.size then
            self.rooms[i].forward_door.x = sw/2
            self.rooms[i].forward_door.y = self.rooms[i]:gen_position_y(sh)
            self.rooms[i].back_door.x = sw/2
            self.rooms[i].back_door.y = self.rooms[i]:gen_position_y(sh) + self.rooms[i].height
        end

        if i == self.size then
            self.rooms[i].forward_door.x = 0
            self.rooms[i].forward_door.y = 0
            self.rooms[i].back_door.x = sw/2
            self.rooms[i].back_door.y = self.rooms[i]:gen_position_y(sh) + self.rooms[i].height
        end

    end

    return self
end
```

Function Dungeon:new will create the whole dungeon, assigning its lenght and rooms.



```lua
function room:new(world)
    local new_room = {}
    setmetatable(new_room, self)
    self.__index = self

    new_room.width = love.math.random(math.floor(screen_width * 0.2 / 64), math.floor(screen_width * 0.95 / 64)) * 64
    new_room.height = love.math.random(math.floor(screen_height * 0.4 / 64), math.floor(screen_height * 0.95 / 64)) * 64

    new_room.active_doors = false
    new_room.forward_door = {x = 0, y = 0}
    new_room.back_door = {x = 0, y = 0}

    new_room.traps = {}

    new_room.has_started = false
    new_room.encounter_time = math.random(60, 120)

    -- Setting the room's tileset
    local numRows = math.floor(new_room.height / 64)
    local numCols = math.floor(new_room.width / 64)

    new_room.tileset = Tiles:new(world, 64, 64, numCols, numRows, new_room:gen_position_x(screen_width), new_room:gen_position_y(screen_height))

    -- If the room will be a special room
    local special_chance = math.random(1, 10)

    if special_chance == 7 then
        new_room.is_special = true
    end

    if new_room.is_special then
        new_room.tileset.is_clickable = false
        new_room.active_doors = true
    else
        local num_of_traps = math.random(0, 3)
        print("number of traps:" ..num_of_traps)
        for i = 1, num_of_traps, 1 do
            local trap = Object:new(world,
            math.random(0, screen_width),
            math.random(0, screen_height),
            16, 16,
            "trap",
            false,
            true,
            2,
            false,
            0,
            10)

            table.insert(new_room.traps, trap)
        end

    end

    return new_room
end
```

Function room:new will create a new room, this function is used in Dungeon:new function to create a new room, each room has random size, traps, doors and a chance to be a special room. In special room a chest with loot could be found.

Conclusion

I'm overall happy with my project, based on what time we had for the project. Of course my project would be looking differently if I had access to my desktop computer and could choose languge I'm already familliar with and I could also add more functions to the game if I had more time for the project.