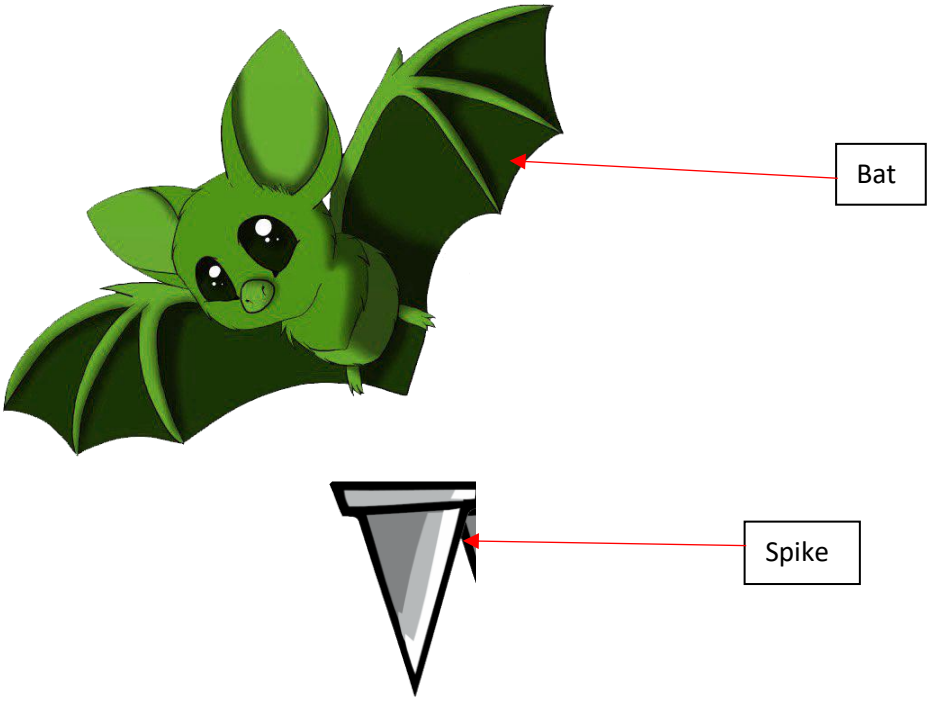
GAME DEVELOPMENT JOURNAL

Group 3

19270151, 19240058, 19258879, 18196497, 18267823, 15172856

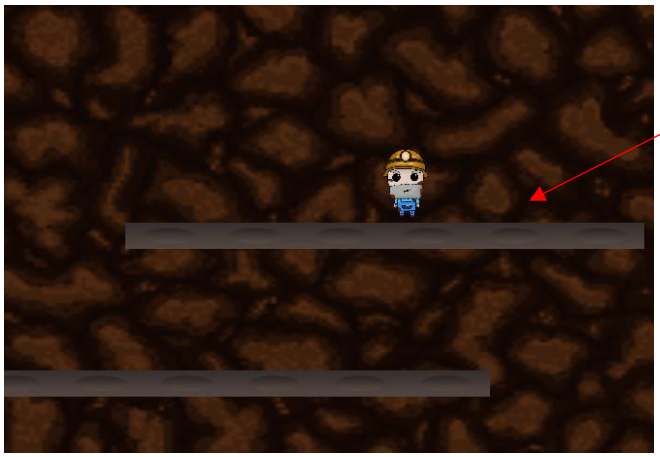
When we started trying to make our idea for the game into reality, we were unsure to how to begin. We used the tutorials on the corona site and also the API. However, there was no good direct source of information. The tutorials were extremely lack lustre and the API gave only short brief descriptions of the tools but gave us no tangible information on how to implement it into our specific code.

Some ideas we had for the game never made it in, because of how tedious and frustrating corona is to use as well as certain things taking longer than expected. Such things as powerups and the dig ability didn't make the cut. Neither did health but that was more of a design choice. Also, the spikes were replaced with bats that fly across the screen. We also removed our idea of having the character jump. This was a design choice included in the final game draft but due to close deadline of the game, we saved time by not implementing jump.



We also had issues with playing a guessing game with the coordinate system. There was no way of directly seeing distances within the game. This was extremely annoying, so we just had to enter in different values until it was placed where we wanted it.

The platforms and figuring out the physics of how to make objects interact the way we want for our game took longer than expected. We had an issue with the platforms as we wanted them to push the character upwards as they rose. However, we initially used the coding from the simple asteroid game that we did in one of the labs to send the platforms up with a set velocity. The problem with this was that the platforms would not move if we set them to be static objects and if we set them as dynamic, they would have a collision with our character and start spinning off in another direction. Also, when we implemented gravity, it caused the platforms to slow down as they approached the top of the screen. To overcome this problem, we found the transition to feature which allowed us to have more control of the platforms speed and also stopped them being treated as a projectile. However, this didn't fix how our character interacted with the platforms, so we eventually found the kinematic feature which perfectly fit what we wanted the platforms to be.



Platform Code

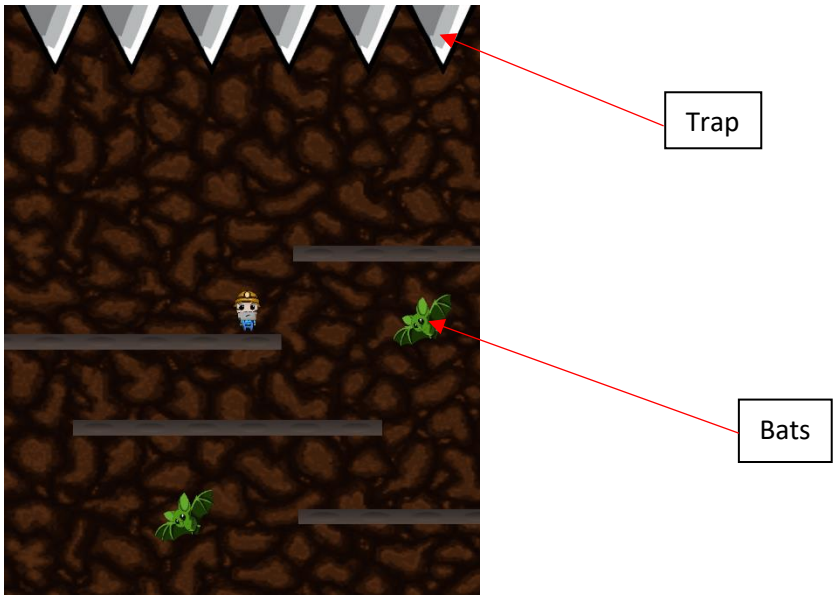
```lua
local platformsTable = {}
-- Holds the last used randomNum for the platform position
-- Kept outside of createPlatform() so it can save between func. calls.
local last_platform_number = nil

 local function create(mainGroup)
  local newPlatform = display.newImageRect(mainGroup,"scene/game/map/platform.png", 4000,200)
  table.insert(platformsTable, newPlatform)
  physics.addBody(newPlatform, "kinematic", {
    isSensor = false,
    bounce = 0,
    outline = "scene/game/map/platform.png"
  })
  newPlatform.myName = "platform"

  local threeSect = math.random(1700) -825 -- 1
  local oneSect = math.random(900) + 1550 -- 2
  local fourSect = math.random(1500) + 3200  -- 3

  -- The maximum positions that platforms can spawn at.
  local max_sections = 3
  local randomNum = math.random(max_sections)

  -- See if we rolled the same position as the last platform.
  if randomNum == last_platform_number then
    -- If the last position was the maximum positions, we loop back to 1.
    -- Otherwise, it just increments to avoid having the same position as the last platform.
    randomNum = (last_platform_number == max_sections) and 1 or (last_platform_number + 1)
  end

  -- Save the last used position's random number for later checking
  last_platform_number = randomNum

  if (randomNum == 1) then

    newPlatform.x = threeSect
    newPlatform.y = 16000
    transition.to(newPlatform, {x = threeSect, y = -1000, time = 7680})

  elseif (randomNum == 2) then

    newPlatform.x = oneSect
    newPlatform.y = 16000
    transition.to(newPlatform, {x = oneSect, y = -1000, time = 7680})
```

We coded the platforms to have 4 different spawn locations (later cut down to 3). These were chosen at random. However, we encountered that the same spawn could happen multiple times in a row, leading to the player escaping the dungeon. To solve this, we implemented a counter to record the pervious platform spawn and guaranteed that it won't spawn twice in a row. The platforms were looking better but they were coming up in exact straight paths and all the platforms were the same length. To fix this, we set a up some random variables which would offset the spawn locations every time but not by too much to have gaps on the edges that wouldn't allow the player to drop down.

```
1  local platformsTable = {}
2  -- Holds the last used randomNum for the platform position
3  -- Kept outside of createPlatform() so it can save between func. calls.
4  local last_platform_number = nil
5
```

We also implemented some safety features to prevent the player from getting below the screen, to do this we set the spawn locations far down below the screen, so that even if the player gets below the screen, he will still be pushed up. However, the issue with doing this is that it caused the player to fall off the screen as the platforms took time to rise. This looked messy and to fix this, we thought about suspending the character in the air for a few seconds before the game began. Unfortunately, we didn't know how to do this and instead implemented a transition fading into the game to disguise this issue.

We decided to make the background move at the same time as the platforms to create the illusion of going downwards. We found a tutorial to do this. However, it showed us how to do it the wrong way around, so we had to reengineer the code to put it the right way.



Challenges encountered:

Delegating tasks each week among members was an ever on going challenge we faced. Our team had six members. Because of the large number, we had to schedule regular meetings to monitor our progress throughout the project's progression. Workload had to be evenly distributed among all team members.

Another challenge we dealt with was how we approached issues in our code. Members would be assigned to working on sections of code. It was difficult at the start to be able to communicate effectively on what the issue involved and what process we should take to solve it. We found that using an application called discord to hold meetings to communicate with all members verbally and visually about a problem was the very effective in dealing with an issue.

One of the first challenges we encountered was being tasked to develop a game in a game engine and programming language that no member of the group had any previous experience in. When we started trying to make our idea for the game into reality, we were unsure on how to begin. We used the tutorials on the corona site and the API. However, there was no good direct source of information. The tutorials on corona labs were extremely lack lustre and the API gave only short brief descriptions of the tools but gave us no tangible information on how to implement it into our specific code. To overcome this challenge, we looked at code for other previously made games that were developed using the corona game engine for inspiration. We also watched YouTube tutorials that were based on creating 2D games in the corona game engine.

A challenge we faced as a team was implementing one of the core mechanics of the game where the game could be endlessly be played. The level would not be predetermined/defined. The level would be randomly generated so every time the player gets a unique experience. We wanted platforms to constantly spawn at the bottom of the screen and rise upwards to the top of the screen, we wanted this to be endlessly looped while the game is being played.

The first early on idea took inspiration from a game presented as a tutorial on corona labs. In the game asteroids would spawn off the sides of the screen and float into view on the screen continuously and the asteroids would be removed from the game if they reached on certain boundary line set. This allowed memory to be freed and to allow a set number of asteroids to be on the screen at once.

Misconceptions encountered:

We were unable to find a fix for this. In addition, one of the biggest misconceptions encountered was that we underestimated the importance of a “game. Lua” module which has caused us a setback of 2-5 days in the project of just trying to make all the modules connect to each other. Not to mention, the struggle to understand how corona and Lua works in general which has caused us to misunderstand different functions and codes.

Some misconceptions we had about making the game was the time frame that we had to complete the game in. The reason for that being we were completely inexperienced in coding using corona. Even more, we were inexperienced in working together on a large-scale gaming project using corona. We had to learn how to properly organise ourselves. Organising this team meant being able to efficiently split the tasks evenly amongst each other to complete our game Dungeon Downfall. All this took time and cooperation from each individual member of this team.

One of the many misconception's we had encountered along the way was the thought that everyone on the team was on the same page. After every meeting we always ended up fine tuning our code because it usually turned out that the game specification was not properly understood.

```
local composer = require ("composer")
local physics = require ("physics")

local backGroup = display.newGroup()
local mainGroup = display.newGroup()
local uiGroup = display.newGroup()
local scene = composer.newScene()

local background = require ("scene.game.library.Background")
local platforms = require ("scene.game.library.Platforms")
local trap = require ("scene.game.library.Trap")
local design = require ("scene.game.library.Designs")
local walls = require ("scene.game.library.Walls")
local bat = require ("scene.game.library.bat")
local hero = require ("scene.game.library.hero")
local score = require ("scene.game.library.score")
local healthBar = require ("scene.game.library.HealthBar")
```

Assets used:

We were initially going to implement our own background, spikes/bats and designs. We already had an idea on how to do it using Photoshop, but we didn’t manage to do them due to the deadline. But we managed to make our own Hero Sprite and our own music. The music used in the game was made by one of our online friends. They gave us full consent to use their music.