Mobile Games Development: Development Blog

McGhee, Ruari
S1432402

13-11-20 – Game Ideation & Project Setup

Ideation

The plan for this project is to create a turn based strategy game similar to Worms Armageddon (https://store.steampowered.com/app/217200/Worms_Armageddon/) and Pocket Tanks (https://play.google.com/store/apps/details?id=com.blitwise.ptankshd&hl=en_GB&gl=US).

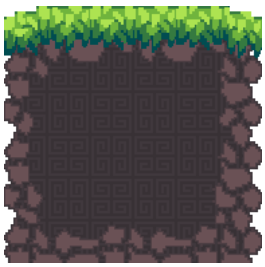
These types of games usually contain a wide variety of features, so to keep the project scope at a reasonable size the game Idea has been stripped down to a very simple form. The game idea is to have 2 players each taking turns to either jump to move round the map or shoot at the other player. Both of these mechanics will he handled through physics-based movement (Like how Worms handle projectiles, the movement will be affected by gravity, air-resistance, and wind-speed).

The overall aim of the game will be to reduce the opponent's health to 0, hitting the other player with projectiles is how the player manages this.

To give some more potential strategy to the game another main mechanic that the final game will contain is the ability for the players to "Destroy" the environment, unlike Pocket Tanks where this is very detailed (the environment reacts to destruction and applies physics to overhangs) the environment here will instead be handled in a more tile-based approach.

Project Setup

Also, some simple art assets have been made to give a rough idea of what the game's visual style could be. To keep things simple a low texture size, heavy pixel art style has been chosen.



Finally, the project's source control has been setup at https://github.com/Ruari026/MobileDev_Coursework

20-11-20 – Project Structure Development

Implementing an Entity Component pattern

To aid the development of the project an entity-component based pattern (like how unity uses) has been developed for the project.

The core of this pattern works on how every object in the game will be build up of components that add functionality to those objects. Using a pattern like this aids the previously mentioned code separation as each component will only affect a very specific aspect of an object's behaviour. Also, by keeping each component's behaviour constrained to a simple scope, this encourages those components to be designed so that they can work generically before adding specific details.

Also, to aid the development of multiple levels (the final game is expected to at least need a main menu, a main gameplay scene, and a game over menu) the concept of scenes has been added to the project. Simply, scenes just act as containers for a group of GameObjects and are responsible for ensuring that those objects update and render every game frame.

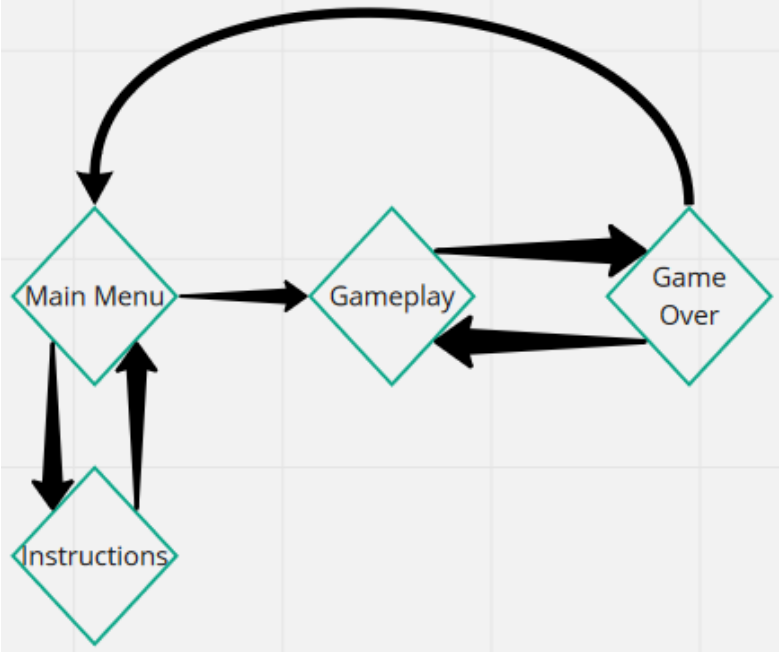


FIGURE 1 - GAME LEVEL FLOW DIAGRAM

These Scenes, GameObjects, and Components use JavaScript's implementation of classes as it allows for a simple way to handle the inheritance that these Classes would require (also JavaScript's implementation of classes also have a lot of "syntactical sugar" that make using them a lot friendlier to someone who has come from class-based object orientated languages)

27-11-20 – Physics Based Movement & Collisions

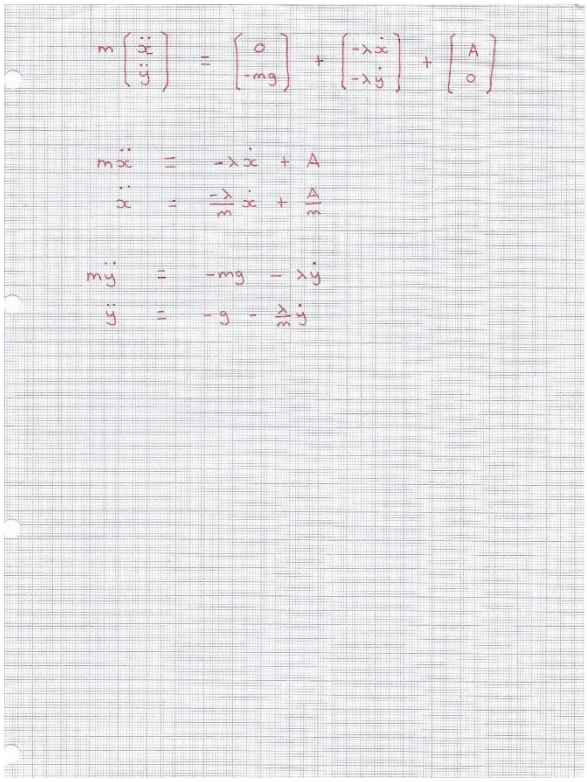
Physics Calculations

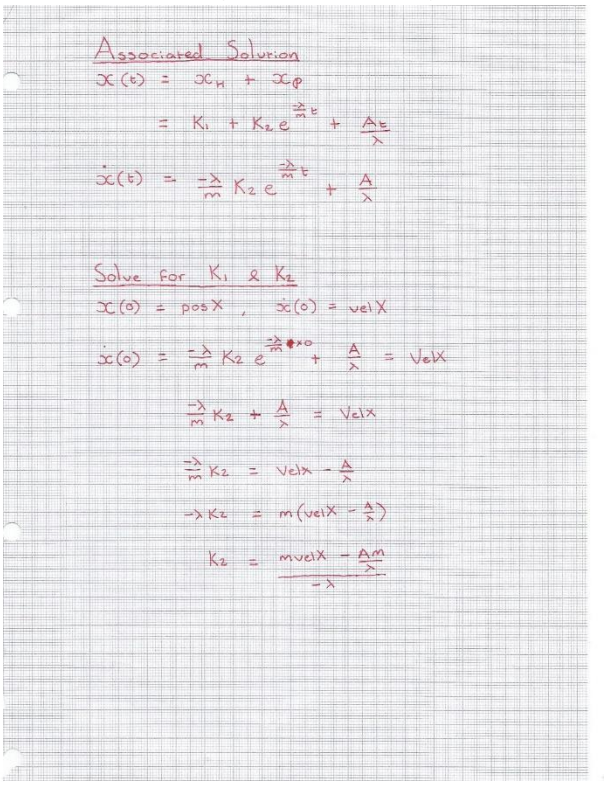
A large feature of the game is how the players are going to be able to jump around the environment, this movements is intended to create a simple simulation of real-world physics by having the GameObjects be affected by various forces.

On a base level the GameObjects will be affected by both Gravity (meaning the objects will always end up returning downwards to the ground which will be useful for preventing the players "Escaping" out of the top of the environment limits) and Air-Resistance which will help slow down all objects over the course of their "Flight Time"

Also, to add a bit of variability to the player's shots/ jumps the objects will also be affected by windspeed, this will only affect the objects on the X-Axis however the amount this affect the objects will change on a turn per turn basis meaning the players will have to adapt their shots according to the current wind speed (This also enables an interesting tactic where if the windspeed is high enough the players may fire into the wind and have the projectiles/their player characters curve back to the opposite direction)

Since the movement is being created from scratch, the equations of motion required for handling the speed and movement have had to be calculated from scratch too (this is because the project intends to have gravity, air-resistance, and wind speed affect the objects during their motion). This working has been taken from (and expanded on) from the Equations of Motion lessons from the 3D Mathematics for Simulation and Visualisation module from 2nd Year (M2I623008-18-A)





$$x(0) = K_1 + K_2 e^{\frac{-\lambda}{m} \times 0} + \frac{A \times 0}{\lambda} = \text{pos}X$$

$$= K_1 + K_2 = \text{pos}X$$

$$K_1 = \text{pos}X - K_2$$

$$= \text{pos}X - \frac{\text{mvel}X - \frac{Am}{\lambda}}{-\lambda}$$

Movement Equation in the X-Axis

$$x(t) = \left(\text{pos}X - \frac{\text{mvel}X - \frac{Am}{\lambda}}{-\lambda}\right) + \left(\frac{\text{mvel}X - \frac{Am}{\lambda}}{-\lambda}\right) e^{\frac{-\lambda}{m} t} + \left(\frac{At}{\lambda}\right)$$

$$\dot{x}(t) = \left(\frac{-\lambda}{m}\right) \times \left(\frac{\text{mvel}X - \frac{Am}{\lambda}}{-\lambda}\right) e^{\frac{-\lambda}{m} t} + \left(\frac{A}{\lambda}\right)$$

Y Direction Calculations

$$\ddot{y} + \frac{\lambda}{m}\dot{y} = -g \qquad \text{// Non-Homogenous Equation}$$

Homogenous Solution

$$\ddot{y} + \frac{\lambda}{m}\dot{y} = 0$$

$$b^2 + \frac{\lambda}{m} b = 0$$

$$b\left(b + \frac{\lambda}{m}\right) = 0$$

$$b_1 = 0 \qquad b_2 = \frac{-\lambda}{m}$$

$$y_H = K_3 e^{0t} + K_4 e^{\frac{-\lambda}{m} t}$$

$$= K_3 + K_4 e^{\frac{-\lambda}{m} t}$$

Particular Solution

$$y_p = \alpha t \qquad (1 \times 0) + \left(\frac{\lambda}{m} \times \alpha\right) + (0 \times \alpha t) = -g$$

$$\dot{y}_p = \alpha \qquad \frac{\lambda}{m}\alpha = -g$$

$$\ddot{y}_p = 0 \qquad \lambda\alpha = -gm$$

$$\alpha = \frac{-gm}{\lambda}$$

$$y_p = \frac{-gmt}{\lambda}$$

$$y(0) = K_3 + K_4 e^{\frac{-\lambda}{m} \times 0} - \frac{gm \times 0}{\lambda} = r$$

$$K_3 + K_4 = \text{pos}Y$$

$$K_3 + \left(\frac{\text{mvel}Y + \frac{gm^2}{\lambda}}{-\lambda}\right) = \text{pos}Y$$

$$K_3 = \text{pos}Y - \left(\frac{\text{mvel}Y + \frac{gm^2}{\lambda}}{-\lambda}\right)$$

Movement Equations in the Y-Axis

$$y(t) = \left(\text{pos}Y - \left(\frac{\text{mvel}Y + \frac{gm^2}{\lambda}}{-\lambda}\right)\right) + \left(\frac{\text{mvel}Y + \frac{gm^2}{\lambda}}{-\lambda}\right) e^{\frac{-\lambda}{m} t} - \left(\frac{gmt}{\lambda}\right)$$

$$\dot{y}(t) = \left(\frac{-\lambda}{m}\right) \times \left(\frac{\text{mvel}Y + \frac{gm^2}{\lambda}}{-\lambda}\right) e^{\frac{-\lambda}{m} t} - \left(\frac{gm}{\lambda}\right)$$

Associated Solution

$$y(t) = y_H + y_p$$

$$= K_3 + K_4 e^{\frac{-\lambda}{m} t} - \frac{gmt}{\lambda}$$

$$\dot{y}(t) = \frac{-\lambda}{m} K_4 e^{\frac{-\lambda}{m} t} - \frac{gm}{\lambda}$$

Solve for K_3 & K_4

$$y(0) = \text{pos}Y, \quad \dot{y}(0) = \text{vel}Y$$

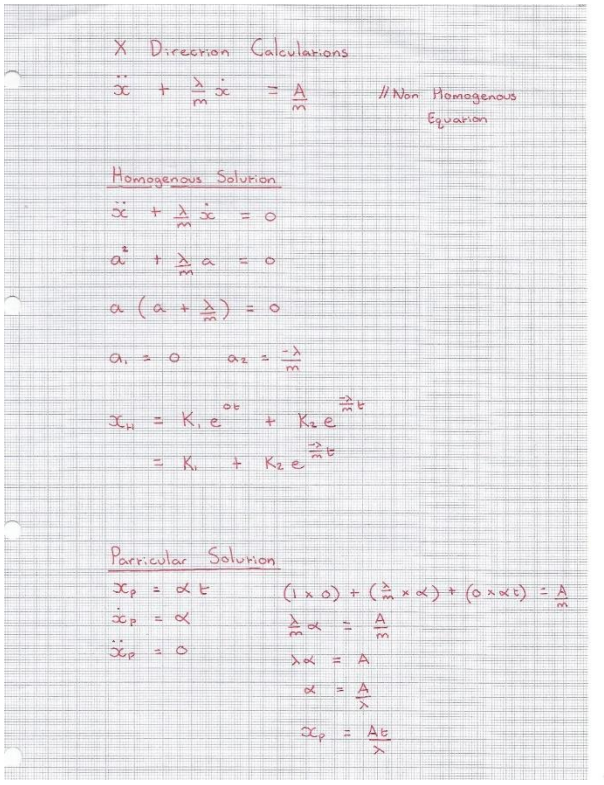
$$\dot{y}(0) = \frac{-\lambda}{m} K_4 e^{\frac{-\lambda}{m} \times 0} - \frac{gm}{\lambda} = \text{vel}Y$$

$$\frac{-\lambda}{m} K_4 - \frac{gm}{\lambda} = \text{vel}Y$$

$$\frac{-\lambda}{m} K_4 = \text{vel}Y + \frac{gm}{\lambda}$$

$$-\lambda K_4 = \text{mvel}Y + \frac{gm^2}{\lambda}$$

$$K_4 = \frac{\text{mvel}Y + \frac{gm^2}{\lambda}}{-\lambda}$$



Game Collisions

To keep things simple all collidable objects in the scene will have box colliders (they will not ever be rotated) and as such when checking collisions between object will use the standard Axis-Aligned bounding box checks for detecting overlaps between two objects.

4-12-20 – Projectiles & Destructible Environments

Projectiles

As mentioned before the game's major interactions comes from two actions main actions, shooting a projectile is the other action that a player can do during their turn to try and defeat the other player.

These projectiles reuse the physics handler component that the player characters use for the jumping mechanic. So that there is consistency between the game's major interactions. Due to reusing the physics handler this also allows the addition of a special event that occurs when the projectiles collide with anything else in the scene. The only exception to this is that projectiles cannot hit the player that spawned it.

Environment Creation

The environment that the player's will be moving around will be a simple block/tile-based approach. Specifically, it will be made up of many platforms/ walls which themselves will be made up of smaller 1x1 tiles. These tiles are the GameObjects that contain the box colliders that the players and projectiles can collide with.

To make level creation easier prefabs have been created for adding walls and platforms to the scene. The main restriction with using these prefabs is that the must be square/ rectangular (passing values through their constructor allows the variance in their size) although this limits the types of environments that can be made for the game, this has been done so that the prefabs can automatically change the sprites for each tile that it creates so that the whole wall/ platform has its textures repeat correctly.

FIGURE 2 - INDIVIDUAL TILEABLE SPRITES FOR ENVIRONMENT TILES

Environment Destruction

As mentioned before the environment is generated as a set of tiles that each have their own individual box colliders. Since the projectiles have a special event that occurs whenever they hit anything, when hitting an environment tile the projectile can make sure to not only destroy itself but also the tile that it hit (these objects wait to get removed from the scene till after the full update loop has finished)

Over the course of the game this means that the environment will be affected by the players actions. This enables the players to try out strategies like tunnelling away from danger and blasting through walls to get to the other player.

18-12-20 – Player 2 + Turn Based Management

Turn Based Controls

Since there are now two players in the scene more restrictions needed to be added to the game to detect when a player's turn had fully completed, for jumping this has been determined to be when the player has fully stopped moving and for the projectile is when either the projectile has hit something or has fallen too far downwards (the player missed the main environment and the projectile fell into the water)

Also work has been done on the main camera, specifically the camera can now follow other GameObjects in the scene. This was done because even though there is a visual change when the control of players switches over (one player's targeting reticule disappears whilst the others reappear), through testing it appeared that it was still confusing at times exactly which player had control (especially if the frogs were close together).

So now the main camera follows the position of the currently controllable player (the renderer's can correctly recalculate their new screen position since they already use the camera's position as their reference point). This also includes if that player is jumping and if the player fires a projectile then the camera will follow that projectile until it collides with something at which point the next player's turn will start and the camera will follow that player.

2nd Player Changes

To add more visual distinction between the two players a second palette swapped version of the frog sprite was added.

8-1-21 – Main Menu & Game Over Scenes + Game Juice

Additional Scenes

To flesh out the full game loop of the game, the final two scenes have been added to the game, the main menu also has been given a small sub menu that gives shows the players how the main game controls.

The game also now remembers which player wins the game, this has been done so that the Game over scene can correctly show the winning player whilst giving the players the options to immediately replay the game or go back to the main menu.

Game Juice

Finally, little bits of juice have been added to the game, this has primarily taken the form of sound effects being added to the various actions that the player can take (e.g., click sounds on the buttons to make them feel more responsive)

Also some more animation has been added to the main gameplay environment, a small wave sprite that tiles and moves across the bottom of the screen was added to flesh out the environment and show the players the bottom limits of the world.

14-1-21 – Final Presentation

The final presentation can be found at - https://youtu.be/cOSs0BsdCqY