**CSCU9N6 Game Report 2615649**

**Introduction**

The game begins with the player spawning through a portal into level 1. The player must navigate through the map, avoiding enemies to collect all the coins before they can proceed to the next level. The player has 3 lives, which are signified by the hearts in the top left corner and must collect 3 coins, shown in the right corner of the screen. If a player dies the option to reset the level is given. If the player makes it through all levels collecting all the coins, they win and can restart the game or quit.

**Prototype implementation**

**Sprite sheets and tile map**

The implementation process began by deciding on the theme to go with and finding sprite sheets and a tile pack to use. After a bit of searching I found some I wanted to use, I just had to create a few new tiles for water to land using an editing software.

**Player controls**

I implemented the player keyboard controls using the key listener to change the boolean state flags for each movement, extending off the skeleton code. These are checked in the playerInput() method using if statements to set the corresponding animation, velocity and sound(for jump). When the key is released, the boolean flag is set to false.

The mouse click teleportation was implemented if the user wants to cheat their way through the level. Using a mouse listener and mouse pressed method it checks the mouse position on the game window (with an offset added) and if the position is valid and empty space, the player can move there. This was mostly used to make testing easier and less time consuming. The controls are: Right/left arrow – move right/left, Up arrow – jump, Left click – teleport, P – Pause, R – Reset, H – Help, ESC – Quit.

**Tile maps and levels**

Initially, I created a basic tile map with the ground and a wall to allow me to test my tile map collision detection. After this was working, I created the tile maps for the 3 levels, switching after the player collected all coins and collides with the portal to the next level, the switchMap() method initializes the next level. Also, after I implemented the enemies, coins, health pack and collision detection, I layered a separate coin tile map and a health tile map in the top corners of the screen to track the number of lives left and the number of coins collected. These are implemented by tracking the coordinates in the tile maps and updating the tiles based on events in the game.

**Tile map collision**

The tile map collision works by getting the tile coordinates at several places on the bounding box of the sprite, checking the character of that tile and if it is a solid block (using the solid method which returns true if solid. E.g. air and water are not solid), then stop the sprite from entering the tile by setting the sprites coordinates to the edge of the tile. The player is also stopped from falling off the edge of the map using the same technique.

**Bounding box collision**

The bounding box collision method checks if 2 sprites collide if their bounding boxes overlap and if they do, return true else return false. The method is used for collectables and enemy collisions.

**Enemies and health**

I added enemy sprites using for loops to conveniently create, update, draw them etc. If the player collides with an enemy, the player loses a heart from the health tile map by changing the heart tile at the current coordinate to an empty tile and the shear draw transform is applied. If the player collects the health pack, a heart is put back using a similar method. If the player loses all lives they die and the game ends.

**Coins**

I implemented coins to each level as sprites. This was done by adding the coin sprites to an array list and adding the offsets, drawing, updating and checking collisions using for loops. When the player collides with a coin sprite, a coin collection tile map is updated in the top right corner by changing the faded coin tile to a gold coin tile.

**Sound**

I used sound for everything that wouldn’t be too annoying to the user by being excessive. I used sounds for spawning/portal noise, this sound demonstrates the sound filter I created which fades out to halfway and fades back in. When the player gets hurt, picks up a coin or health pack, dies or completes a level.

These are all .WAV files. I used a midi file for the background music. To ensure these sounds don’t repeat with every pass through the update loop I used counters or Booleans to see if the sound has already been played and reset this when the sound is finished.

**Game states / screens**

There are several different states that the game can be in. this is tracked using a boolean variable for each state. The states are: game in play, paused and on paused screen, on help screen, player dead and game won. This is a crucial part of the game as an error in the logic could cause problems like the player being able to move when dead or paused. Overlapping menus or even the player being locked from moving at all. Some states like paused, help, dead, !play and won will stop the game from updating and pause the game by returning from the update loop. Help and pause are toggled using key listeners changing the corresponding boolean variable.

**How the prototype could be extended**

* It could be extended to have more levels easily off the code that I’ve produced simply by having an initialize level method for each level to place the sprites. All if statements could include another else if to handle the extra level.
* The levels could be made longer by extending the tile map and drawing the background more times in the for loop. Same could be done for coins, enemies and clouds.
* Further controls like attacks and sprint could be added in extending the player input method

**Further functionality given extra time. I would like to:**

* Make the game more challenging.
* make more and larger levels with more variety between them like different sounds, backgrounds, tiles and enemies. Perhaps enemies that follow the player.
* have some sort of way for the player to attack and kill the enemies either by jumping on them of firing a projectile at them.
* Add a timer with a leaderboard that stores the fastest runs with the player incurring a time penalty for using the mouse to teleport.
* use the circular bounding box for collisions.
* find a better background music midi and turn on and off different layers depending on events in the game.
* Also, I would have liked to add a portal sound that gets louder as the player gets closer.

**Honest Evaluation**

**Where is was successful**

I think as a whole the game was a success. The parts I would say were the most successful:

* The use of layering the tile maps for health and coin displays. As the player loses a life, picks up a health pack or collects a coin the appropriate action is taken on the tile map, remove or add heart and add coin.
* The 3 layers of parallax scrolling. The background image came as one single image, so I used publisher to separate the image following the outline of the mountains and edited and split them into 2 different images to give more depth with layering of the images. The further back, the slower the images move. I used sprites and animations for these images to allow for more control.
* The use of sound was a nice touch and enhances the user experience and emersion. The creation of my own sound filter, that is played when spawning. It fades out until halfway and then fades back in.
* Some other small successes were the coin and health pack drawing and collection, making the player collect all coins before they can complete the level, the click teleportation only working when click on open space, managing the different game states and finally the appearance as I would say the menus, tile pack and sprites I chose look nice.

**Where needs improvement**

* There are a few areas where my game could be improved. The main being my tile map collision detection, although I would say that it is sufficient to be able to play and enjoy the game for the most part. For example, there are some parts where the player can get stuck in the side of tiles and defy gravity.
* I could have used transforms for the animations rather than having a sprite sheet for every movement. For example, rather than having idle left and idle right sprite sheets, I could have used the set scale method and use a negative x value to reverse the right idle image and draw transformed.
* I also think I could have handled the game state flags in a more efficient way perhaps using an enum, and the sound counters using a more efficient method perhaps.
* I would like to make the backing music replay once it finished and play in a loop, but this caused some issues with the thread when I attempted to do so.