**Computer Games Development CW208**

**GDD and Project Report**

**Year IV**

**Keith Wilson**

**C00205321**

**[03/05/20]**

**Faculty of Science, Department of C & N**

**Open-Book and Remote Assessment Cover Page**

Student Name: Keith Wilson

Student Number: C00205321

Lecturer Name: Philip Bourke, Lei Shi

Module: CW208

Stage/Year: 4

Date: 03/05/2020

**Declaration**

This examination/assessment will be submitted using Turnitin as the online submission tool. By submitting my examination/assessment to Turnitin, I am declaring that this examination/assessment is my own work. I understand that I may be required to orally defend any of my answers, to the lecturer, at a given time after the examination/assessment has been completed, as outlined in the student regulations.

**Contents**

[Project Abstract](#fugkcpa2tq3s) 4

[Project Introduction](#cd4sm7980j0z) 4

[Background](#t8aa4t43csct) 5

[Description](#yc6lgkh96wlm) 5

[Features](#cjselwvkpns3) 6

[Review and Conclusions](#e087y9je3e8f) 8

Page 3 of 8

**Project Abstract**

The Raspberry Pi is a low cost computer chip around the size of a credit card created by the Raspberry Pi Foundation. The purpose of the Raspberry Pi was to provide a cheap but effective mini computer so that people all over the world could learn programming.

This project dives into the game development process on the Raspberry Pi 3. The focus of this research is to develop a 2D platformer game using SFML that runs efficiently on the Raspberry Pi 3, detail the development process of creating a SFML game on the Raspberry Pi and provide the resources and steps for someone else to follow if they wish to also make a SFML game for the raspberry pi. My research has helped me to figure out how to build and install SFML onto the raspberry pi, how to compile an SFML project using cmake to build an executable. After achieving a working build on the raspberry pi I set to work developing the game.

The result of this project has been a 2D platformer built using SFML that compiles and runs on the raspberry pi exceptionally well. I have also provided the information that i have used to build and install SFML and activate the raspberry pi’s openGL ES drivers so that someone who wants to build a similar project to mine can get set up and onto programming very quickly. The cmake files I used can also be used as a guide/baseline on how to use cmake to compile that person’s project.

**Project Introduction**

Raspberry Pi’s are a remarkable piece of technology at an affordable price with a broad range of uses which range from something as simple as running a Plex server to something as complex as being paired with an Arduino to make robots and other types of machines. The reason I chose this project was because I have had an interest in raspberry pi’s for a while and wanted to explore the possibilities and potential of games development on the raspberry pi and document it so others could use my project as a viable starting point for getting set up and running very quickly with minimal issues and an example project to learn from.

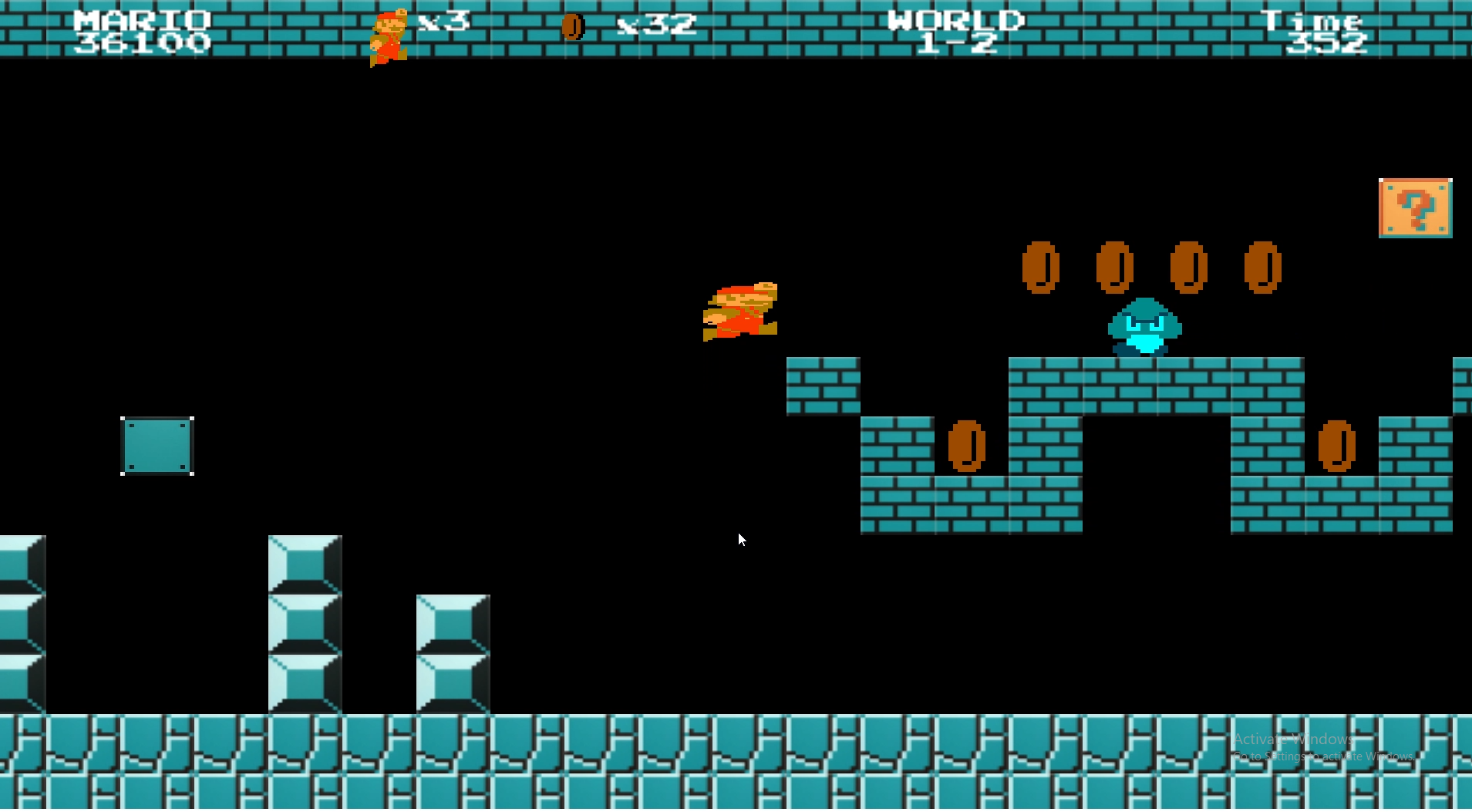
Page 4 of 8

**Background**

The focus of this project is to build a 2D platformer that runs efficiently on the raspberry pi. The 2D platformer I chose to clone was Super Mario bros. The reason I chose this game is because Mario is the usual go to example of a 2D platformer and also is a very nice theme to set for the game. Super Mario Bros also has reasonably big tile maps that make memory management end render culling necessary for good performance on the raspberry pi. If the levels are not unloaded after the end of each level they will not be removed from memory and will cause the pi to run out of memory very quickly. If the pi is rendering everything in the level regardless if it is on screen or not, large levels with a high amount of tiles overwhelm the GPU and will cause a massive reduction in framerate making the game unplayable.

**Description**

At the end of the project the platformer should closely resemble Mario and have a close feel to the original game. Gameplay should feel smooth and responsive and have no issues in regards to performance i.e. when the player jumps, he jumps immediately and to the expected height. When the player reaches the end of the level every element that was loaded in that level should be removed using dynamic memory deallocation and the new level should be then loaded. The project will also contain information on how to set up and install SFML and the cmake files used to compile the project.



Page 5 of 8

Technical Achievements:

* The game runs at a consistent framerate on the raspberry pi.
* I achieved proper memory management as a result of unloading the levels when they are completed thus releasing memory. This was very important as the Raspberry Pi 3 model A only has 512mb of RAM to work with so if there was a memory leak this memory would fill up very quickly.
* I optimised GPU and CPU processing by only rendering or updating collisions of game elements that are in view on the viewport. This helped a lot with bigger levels that originally caused the build to struggle.

Personal Achievements

* I have learned how to build projects for the raspberry pi, a platform I have never used before and had no previous experience with.
* I added a ReadMe to my project with instructions on how to compile and install SFML for the Raspberry Pi and how to enable the pi’s openGL ES drivers. This I hope combined with the cmake files available in the project will be of help to someone who is looking to build an SFML project for the pi and is looking for assistance for getting started.

**Features**

**Feature 1: Level Loading**

* Levels are loaded from text files at the start of the game.
* Levels are unloaded at the end of the level to release memory.

**Feature 2: Tile Collision**

* The player collides and interacts with the tile platforms in the level.
* Uses AABB collision: distance - (halfSizeA + halfSizeB) = Intersect.

**Feature 3: Player Movement and Jumping**

* The player can move left and right with the A and D key but not out off the level boundary.
* The player can jump a set height by pressing the spacebar. The player cannot jump again until they land on a platform.

Page 6 of 8

**Feature 4: Lives**

* If the player hits an enemy, runs out of time or falls out of bounds they die, the level restarts and they lose a life.
* If a player loses all of their lives they get sent to the game over screen and then back to the main menu where they can play again if they want to.

**Feature 5: Coins**

* The player can pick up coins freely in the world or by hitting the bottom of the yellow tiles with question marks on them.
* If a player collects 50 coins, they lose them and gain a life.
* A player earns a score of 200 everytime they pick up a coin

**Feature 6: Score**

* The player earns a score by killing enemies(100), collecting coins(200), completing the level(5000) and for the amount of time remaining(100 per second remaining).

**Feature 7: Time**

* The player starts with 400 seconds to complete the level. If the player runs out the player dies, the player loses a life and the level restarts.

**Feature 8: Enemies**

* Enemies move a set distance left and right.
* If the player jumps on top of them the enemy is killed.
* If the player hits the sides of the enemy the player dies and loses a life.

**Feature 9: Bonus Levels**

* At the end of each level there is a random bonus level.
* This gives the player a number of coins to help them earn a life if they’re low.

**Feature 10: Animation**

* The player, coins and enemies have animations.

Page 7 of 8

**Feature 11: Sound Effects**

* When the player performs a jump, dies, picks up a coin there is a sound effect.

**Project Review and Conclusions**

In conclusion, the game compiled and ran successfully on the raspberry pi and had no run time errors or memory leaks. The game ported from windows to the pi very smoothly and with no major problems. I achieved the aims I set for myself of successfully building an SFML project to the pi and running it efficiently and I also recorded in a ReadMe how to set up SFML for the raspberry pi and get it working so that someone else can easily follow the instructions and set it up for themselves. However as a result of settling on the mario theme late i left myself short on time to complete important features such as Bowser, more diverse enemies and more maps. I also did not have time to make a second pass through the code to clean up and simplify some inefficient code that I otherwise would have.

Page 8 of 8