Mario Project Review

By Stephen Jackson (j018873k)

Table of Contents

[Introduction 3](#_Toc70089341)

[Gameplay Overview 3](#_Toc70089342)

[Menu 3](#_Toc70089343)

[Level 1 Screen 3](#_Toc70089344)

[Mario/Luigi 3](#_Toc70089345)

[Koopa 4](#_Toc70089346)

[Coin 4](#_Toc70089347)

[POW Block 4](#_Toc70089348)

[Audio 4](#_Toc70089349)

[Images of Game 5](#_Toc70089350)

[5](#_Toc70089351)

[6](#_Toc70089352)

# Introduction

This report will contain a summary of the work done on the recent GEC assignment. The game created was a Mario style game whereby the player(s) can control either Mario or Luigi. The objective of the game is to collect the coins whilst avoiding or killing the enemies (Koopas) who would move left/right across the screen on different platforms.

# Gameplay Overview

The aim of this game is to collect coins as either Mario, Luigi or both whilst jumping on or over the Koopas who will kill you if you touch them from anywhere except the top.

# Menu

Once loaded, the Game Screen Manager class loads the Menu screen by default. This Menu screen is used as an indicator of how to play the game by showing the player the character controls along with what they should avoid (Koopas) and what they should collect (Coins). Hitting ‘RETURN’ on the keyboard begins the game as advised and cues the Screen Manager to switch to Level 1.

The ‘Menu’ screen, was edited together using assets provided by Staffordshire University for use on this project but compiled together by the author using Photoshop.

# Level 1 Screen

After the ‘RETURN’ key has been pressed on the menu screen, the music will start and the game will begin. Level 1 features obstacles in the way of ledges. These ledges have collision via a tile system created in the Level Map class. If a tile is set to ‘0’ then there exists no collision however, if a tile is set to ‘1’ then neither Mario/Luigi can pass through these tiles. As well as the ledges, there are 2 enemies which the player needs to avoid/kill and 6 coins which the player can collect. This level also features four pipes (one in each corner) that the player/enemy can traverse through. If a player enters a pipe, they will be transported to the pipe diagonally opposite.

The ‘Level 1’ screen asset was provided by Staffordshire University for use on this project.

# Mario/Luigi

The game featured a control scheme whereby the Mario player could move Left and Right along the X-Axis by pressing Left Arrow or Right Arrow respectively. The player could also move Up along the Y-Axis by jumping by pressing the Up Arrow. Luigi has the exact same functionality as Mario bar the controls, whereby Left Arrow, Right Arrow and Up Arrow are replaced by ‘A’, ‘D’ and ‘W’.

Animations were added to both characters to ensure that they are always facing the direction in which they are moving. Alongside this, there is a 2-frame animation whilst moving left and right and a 1-frame animation for jumping.

Base sprites for Mario/Luigi were provided by Staffordshire University for use on this project. Additional sprites were edited by the author.

# Koopa

The Koopa is programmed to move left and right along the X-Axis. If a Koopa hits a wall, they will swap direction and begin moving in that direction. Koopas will kill Mario/Luigi upon colliding with them except if the value of the Y-Axis of either Mario/Luigi is higher than the Koopas, in which case, they will become injured and flip upside down for a set amount of time. If Mario/Luigi collide with the Koopa again whilst in this state, the Koopa will die.

There is a walking animation for the Koopa when uninjured and the switch to the Koopa’s shell when injured.

Base sprites used for Koopa was provided by Staffordshire University for use on this project. An additional sprite was edited by the author.

# Coin

On the Level 1 screen exist 6 coins which either Mario/Luigi can collect. No functionality has been added for these however, the author would have liked to add a scoring system whereby collecting a coin would add to the total score value.

Animations for the coins were implemented so that they rotate on the spot.

The assets used for the Coin was provided by Staffordshire University for use on this project.

# POW Block

There is one POW Block located in Level 1 of the Mario game which is set to ‘2’ on the LevelMap so as to avoid collision not working if set to ‘1’. The POW Block has three charges and if either Mario/Luigi jump into the POW Block from below, the POW Block will lose a charge, the size of the POW Block will shrink by a third, the screen will shake, any uninjured Koopa will be injured and any injured Koopa will have their injured time reset. Once the POW Block has been hit three times, it will disappear and the level map is set to replace the values to ‘0’ to ensure no further collision can be had on these tiles.

The assets for the POW Block were provided by Staffordshire University for use on this project.

# Audio

There is no audio on the Menu screen of the game. It is only when the player hits ‘RETURN’ on the keyboard where the music is initialised. This music loops throughout the level, only stopping once the game has been exited.

The assets for audio were provided by Staffordshire University for use on this project.

# Test Plan

Here is a list of the bugs I encountered whilst creating the game detailed with the actions I took in order to resolve them:

**Solution (GameScreenLevel1.cpp file):**

**Bug:**

|  |  |  |
| --- | --- | --- |
| **Bug** | **Solution** | **Relevant** **File** |
| Characters could walk off the edge of the screen | Added collision code so that Character stop when hitting sides of the screen. | Character.cpp |
| Koopas would stop progressing when they reached the edge of the screen | Added a function in Character.cpp called ‘AddWindowCollision()’ which both Mario and Luigi could call in their update functions so as to ensure that Koopas would be unaffected by this. Added code within the Koopa file to make them turn around when they collide with a wall to ensure that they follow a continuous path. | Character.cpp, CharacterKoopa.cpp, CharacterMario.cpp, CharacterLuigi.cpp |
| Koopas would just walk back and forth at the bottom of the screen | Added code to warp any character that touches a pipe to ensure that this does not happen. | Character.cpp |
| Characters could jump through tiles marked as 1 on the Level Map. | Created a new int named ‘head\_position’ to calculate the top of a character’s sprite sheet and used the existing code I had to add gravity when a tile was ‘0’ on the level map to ensure that the ‘CancelJump()’ function is called when the head matches the ‘1’ tiles. | Character.cpp |
| Adding the previous collision fix stopped characters from interacting with the POW block as their jump would be cancelled before the POW block could be hit. | Edited the Level Map for Level 1 and changed the value of the tiles from ‘1’ to ‘2’. As these weren’t 0, ‘AddGravity()’ function wouldn’t be called when on these tiles to they worked as blocks when stood on top of them and players were able to collide without issue. | GameScreenLevel1.cpp |
| Mario and Luigi animations would not play fluidly. | I placed the ‘WalkAnimation()’ function in the key presses for each character instead of simply seeing if the character was moving left/right and if so, to call the ‘WalkAnimation()’ function. | CharacterMario.cpp, CharacterLuigi.cpp |
| Mario and Luigi animation would be stuck on the Jump frame when returning to the ground after a jump | As the sprite sheet jumped to the ‘Jump’ sprite when IsJumping() equalled true, there was no reason for the ‘WalkAnimation()’ function to update the sprite once landing so the character’s remained in the ‘Jump’ sprite. To resolve this, I added an else statement to say if the character isn’t jumping and the character isn’t moving, then the frame should = 1 (idle sprite on sheet). | CharacterMario.cpp, CharacterLuigi.cpp |
| When implementing sprites for Mario, Luigi and the Koopas, they stopped colliding with the world correctly and would fall through tiles. | The Character header file was where the collisions were based thus it made sense that the issue would be found in this area. Upon thinking about what had changed before the bug showed itself, it became clear that the Character class was still looking at ‘m\_texture’ as a whole however, sub-classes such as ‘CharacterMario’ and ‘CharacterLuigi’ were cutting their ‘m\_texture’ width by 3 so the answer to this was to take the ‘m\_texture’ in the Character class and also cut this by 3. | Character.cpp |
| Cutting ‘m\_texture’ in Character class by 3 resulted in most collision being fixed however, there were still certain tiles that characters would fall through. | Uncertainas toexactly why this did not resolve the issue as setting ‘m\_texture’ to a static 32 pixels to represent a single sprite seemed to resolve the issue. This fix is not great as it relies on any character inheriting from this class to also be 32 pixels in width which may not be realistic for other projects. | Character.cpp |
| When attempting to load a .wav file into ‘g\_high’, whole program would crash citing that ‘g\_high’ was set to NULL. | Was not able to resolve. Code is in place to ‘cout’ the error message however, as the whole program crashes, I am not able to read the console in time to find out what the error was. Googling a general error has provided search results of people experiencing similar issues but has not provided a suitable answer. |  |

# 

# Images of Game

Start Screen:

# 

During Gameplay:

# 