

UNIVERSITI TUNKU ABDUL RAHMAN

LEE KONG CHIAN FACULTY OF ENGINEERING AND SCIENCE

UECS3173/3273

PROGRAMMING WITH GAME ENGINES

Software Engineering

Lecturer: Dr. Simon Lau Boung Yew

Team Name: Solo

Game: Jumpman

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Name** | **Student ID** | **Course** | **Practical Class** |
| 1 | Tan Ying Yao | 1703648 | SE | P1 |

**Brief Summary**

A 2D Action Platformer where the objective of the game is to progress through levels by defeating enemies, collecting items and solving puzzles without dying Power-up use in a two-dimensional gameplay is integral too. Jumpman hops on platforms and enemies while avoiding their attacks and moving to the right of the scrolling screen. The levels feature a single-exit objective which must be reached to win the level. There will be items which can be gained from item blocks through hitting them. The first level is designed to introduce core mechanics and the player moves from the left side of the screen to the right side in order to reach the flag pole at the end of each level.

The game world features coins scattered around for Jumpman to collect to serve as incentive for reward. Players start with a certain number of lives and may gain more by picking up coins. Jumpman loses a life if he takes damage 2 times in a row or falls in a bottomless pit. The game ends when the player runs out of lives, although a button input can be used on the game over screen to continue from the first level of the world in which the player died. Jumpman’s primary attack is by jumping on enemies and the enemies may try to circumvent this by moving around or avoiding them. The level design should be intuitive enough to serve as a tutorial for the player to learn easily. The opening section should be specifically designed in such a way that players would be forced to explore the mechanic of the game in order to be able to advance.

**Target Platform:**

**Generally content developed with Unity can run pretty much everywhere. How well it runs is dependent on the complexity of your project. More detailed requirements:**

* **Desktop:**
  + **OS: Windows 7 SP1+, macOS 10.12+, Ubuntu 16.04+**
  + **Graphics card with DX10 (shader model 4.0) capabilities.**
  + **CPU: SSE2 instruction set support.**
* **iOS player requires iOS 9.0 or higher.**
* **Android: OS 4.1 or later; ARMv7 CPU with NEON support or Atom CPU; OpenGL ES 2.0 or later.**
* **WebGL: Any recent desktop version of Firefox, Chrome, Edge or Safari.**
* **Universal Windows Platform: Windows 10 and a graphics card with DX10 (shader model 4.0) capabilities**
* **The target platform of this game is Windows**

**Technical Specifications:**

* **The program comprises of 4 level with a main menu and game-over scene.**
* **Character can move around in eight directions and runs and jumps when prompted**
* **The game contains a means to defeat a set number of targets and the level have platforms to jump onto. The targets have aggressive behavior and can hurt the player.**
* **The player character can be eliminated and respawned at a starting location/point.**
* **Sound and other assets are be taken from open-sources and SFXs are included.**

**Game Mechanics:**

As you progress through the game the screen will slowly advance to the right, therefore the player is prompted to move in this direction to proceed. At the end of the forth area the player will reach a large castle. Each area has a time limit and the player has to reach the end within the time limit or a life will be lost. Upon reaching the castle the next level is an end boss lava themed dungeon where the player will have to battle the evil Turtle King to save a Mushroom person. This then ends the level and a final score for the area is totaled, the player then advances to the next area. However getting to the flag pole at the end of an area is not a simple task, along the way players will encounter mountains, pits, sea, ugly monster and many puzzles. Jumpman can defeat enemies by either jumping on their head or throwing a fire ball at them if a fire flower power up has been collected. If Jumpman touches an enemy from any other direction than the head, falls down a pit, or touches lethal scenery (e.g. fire, spikes) a life will be lost and the player will start again from the beginning of that area. However if Jumpman has reached the middle of the area, if he dies he will start in the middle of the area so the player doesn’t have to start from the very beginning. If all lives are lost then the player is forced to restart from area one within that current world. Along his way through each area, Jumpman can head butt scenery some of which is destructible and may reveal secret items such as a Magic and 1-Up Mushrooms, Fire Flowers, Invincibility Stars and Coins. Collecting 100 coins will also give the player an extra life.

**Scoring:**

While the main aim is to complete Worlds and Areas, every enemy Jumpman kills, block smashed, coins/power ups collected, area completed give the player points. Points gained by killing enemies (all enemy descriptions and images can be found in the characters section) are as follows:

Collecting Power Ups 1000 pts   
Coins 200 pts   
Bashing a Block 50 pts   
Final Flag Pole 0-5000 pts

**Gameplay Elements**:

The game play is linear is that you have to move from left to right and only by reaching the end of an area can the player progress. Level design and enemy characters are in place to stop the player achieving this. Completing an Area takes timing and skill on the players’ part. Throughout an area any main structure within can be stood on, and in the case of bricks can be interacted with and destroyed by jumping into them. Climbing on these areas can lead to hidden areas, and breaking these blocks can lead to coins, Power-up mushrooms, fire flowers etc… Each area will also end with a ‘Finish Area’ that comprises of steps, a flag pole and the end castle. This bears no real importance apart from finishing the level and gaining extra points. Encountering enemies gives the player two options, avoiding them can be enough to complete an Area, however some may require to be killed to move past or killing them may provide a simpler route. Each enemy killed also offers points. It requires thought and skill to find out how this is done and how to go about either killing or avoiding them. When Jumpman hits an enemy from the side or above in the case of an enemy that can’t be jumped on, it will cause damage to the player. Jumpman can also ‘kick’ these shells at other enemies once the enemy has gone into hiding into it. Kicking them is carried out by simply running into them, and whatever is in the shells path once kicked will be killed. Any of the above collectible items can be used by Jumpman by running into them, each having their designated effect instantly. When in ‘Super’ or ‘Fiery’ Jumpman state, the player can get hit by an enemy twice before dying. However once Jumpman is small again it will take only one hit to kill him.

**Game Physics and Statistics:  
Movement**

Jumpman can move in eight directions, when stationary the direction pad allows Jumpman to either run to the left or right, and also duck when either ‘Super’ or ‘Fiery’ Jumpman. When left or right is held down along with the X(Customizable), Jumpman will pick up speed and begin to run in this direction. The Z Button is a major factor in Jumpman’s movement, as this button enables Jumpman to jump. This is an integral movement to the game play and can be used in conjunction with the direction pad allowing Jumpman to jump in a direction. The Z Button can also be used with the X Button and the direction pad to perform a super jump so that Jumpman can clear large pits and chasms. Enemies will move in a predictable pattern, with most just walking back and forth in a section of a level. Some enemies, especially those that fly may follow a set pattern that they will constantly repeat. No enemies have the same freedom of movement that Jumpman has. Damage and Recovery Running into enemies from the sides, jumping on spiky enemies, falling down pits, falling in lava all cause damage to Jumpman’s health. Jumpman will initially start the game ‘small’, which signifies one health point. If the player collects either a Magic Mushroom or Fire Flower, Jumpman will grow in height which signifies two health points. When Jumpman is hit by an enemy in his taller state, he will return to the ‘small’ Jumpman, in effect losing one health point. If hit by an enemy when ‘small’ Jumpman will lose a life. The player starts the game with three lives and once all lives are lost the game is over and the player is forced to start again.

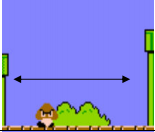
**General Physics Model:**

The game physics in Jumpman. are extremely simple. Jumpman’s movements are not based on reality and are all in proportion to the world around him. He moves faster than any enemies so can outrun them in any situation, and can also jump over all obstacles. Using the X Button Jumpman can speed up running faster and jump further, which is roughly double the speed he would normally run at. Most level design takes into consideration the movement of Jumpman so all pits are approximately the length Jumpman can jump, and all sections above Jumpman are at a height Jumpman can reach. Therefore this keeps the world in proportion.

**AI Features**:

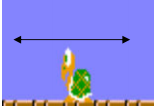
**Mushroom Goon**

The intelligence of the enemies can be categorised by how the enemies behave. This behaviour for most is extremely simple and can be sorted as follows:



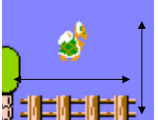
- These enemies behave very simply by moving along the floor in Jumpman’s direction. Once past Jumpman they will either leave the screen or if an obstacle is present they will change direction and move back the opposite way.

**Turtle**



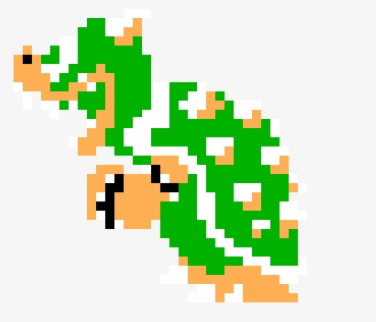
Turtles behave in a similar way to Goons as the only move along the floor and in the direction of Jumpman. However unlike Goons, Turtles have 2 health points so Jumpman has to jump on them twice.

**Flying Turtle**

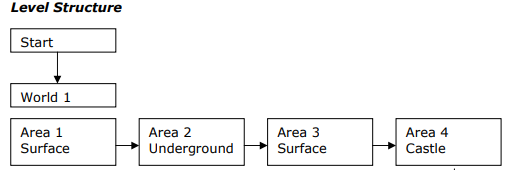


Flying turtle are usually found moving in a similar way to normal turtle, however they have the ability to move vertically as well as horizontally. They also have 2 health points so Jumpman has to jump on them twice.

**King Turtle**



King turtle dies when the mechanism is activated and he drops into the lava

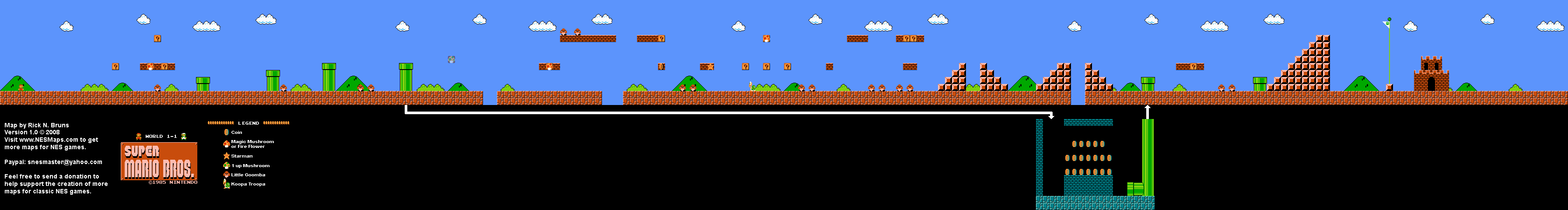


**Game Setup:**

Level Design

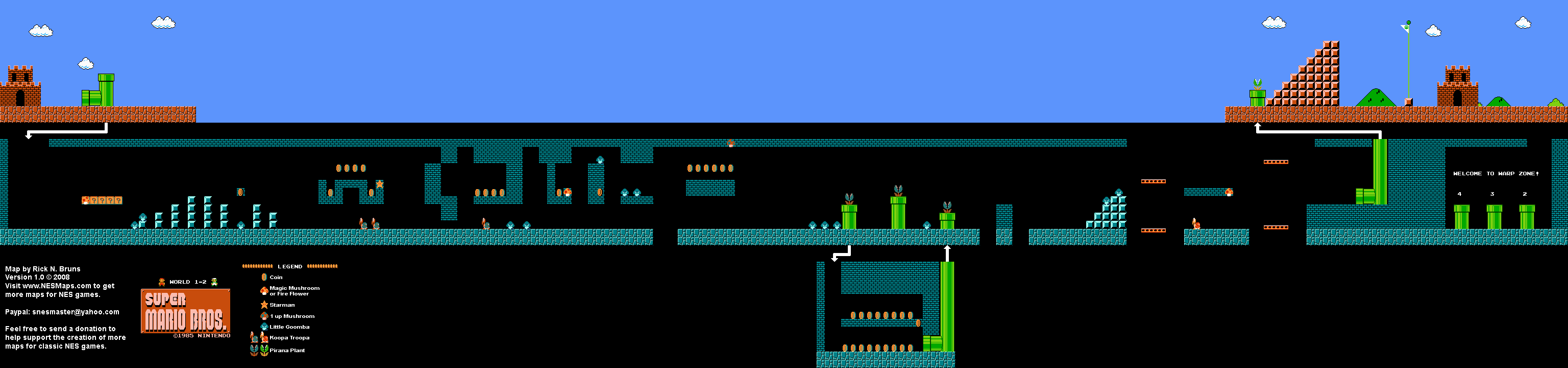
Currently there are 4 levels comprised of the following:  
i) Tutorial

The tutorial level serve as a basic area for the players to familiarize with their controls:



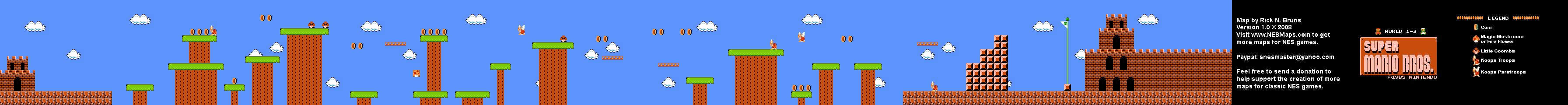
ii) Dungeon

A dark underground level filled with deadly traps to test the player’s skill:



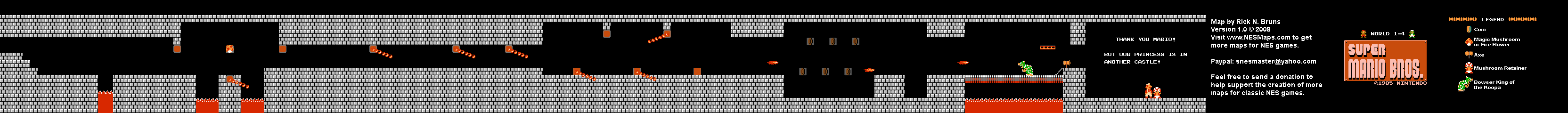
iii) Sky High

An open area with many platforms to jump onto to avoid certain death:



iv) Boss Fight

A simple boss fight with the titular nemesis of the game and saving the princess:



Character Control and Animations:

Jumpman



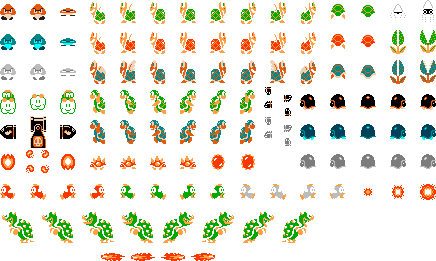
Character death system

The level ends if the player runs out of 3 lives or the timer reaches it’s limit



Enemies and their respective animation:

The enemy will walk around to harm the player

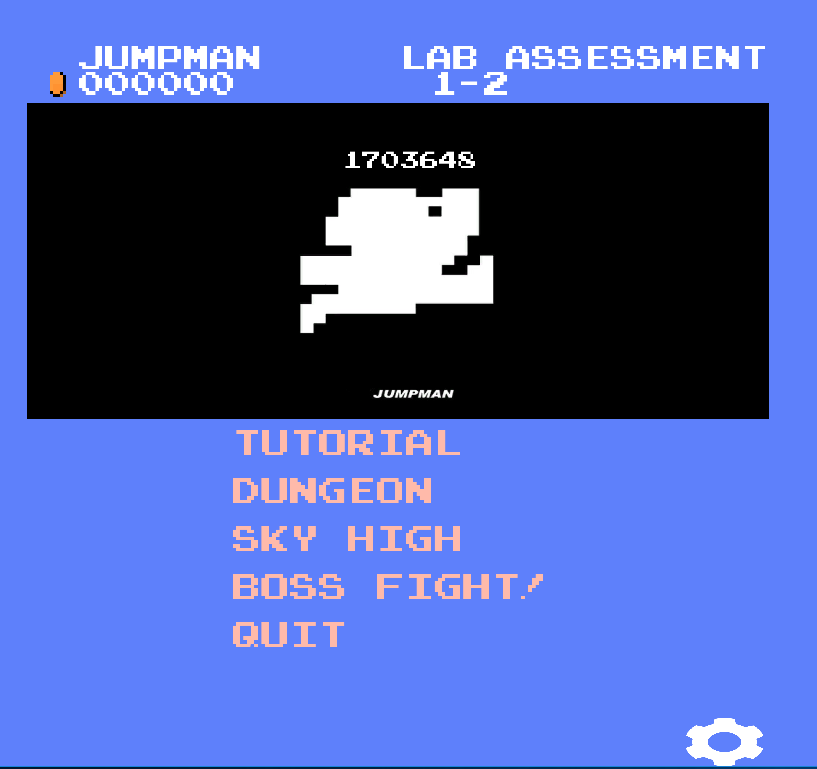
****

Collisions

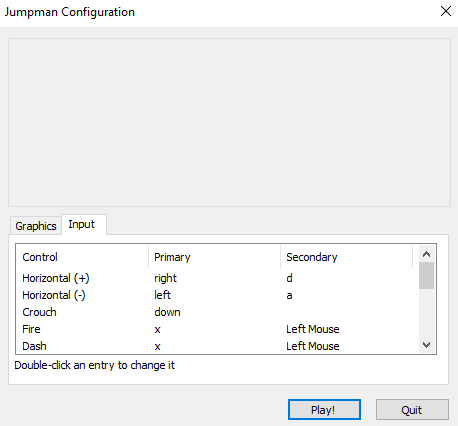
Audio

The audio uses a remix version of the classic theme and can be found here:  
<https://www.youtube.com/watch?v=T_WUvxO0lnk>

Menu



Input:  
  
Unity allows the binding of controls thus it is very accessible



Engine & Architecture:  
  
The choice of game engine is Unity due to it’s beginner friendly nature.

Coding & Usage:  
  
i) using UnityEngine; // invoke libraries of unity engine  
ii) Animator was used to control the Mecanim animation system.  
iii) Vector2 used in some places to represent 2D positions and vectors (e.g. texture coordinates in a [Mesh](https://docs.unity3d.com/ScriptReference/Mesh.html) or texture offsets in [Material](https://docs.unity3d.com/ScriptReference/Material.html)).  
iv)Transform used to store and manipulate the position, rotation and scale of the object.  
v)SpriteRender used to render sprite  
vi) BoxCollider2D ( a box shaped primitive collider)  
vii) Rigidbody2D (physics component for 2D sprites)

Coding Conventions:  
-Uses C# Layout Convention

// Sample code of what happens when a coin block is being hit

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

public class Coin : MonoBehaviour {

private LevelManager t\_LevelManager;

// Use this for initialization

void Start () {

t\_LevelManager = FindObjectOfType<LevelManager> ();

}

// When hitting an object, a coin is added

void OnTriggerEnter2D(Collider2D other) {

if (other.gameObject.tag == "Player") {

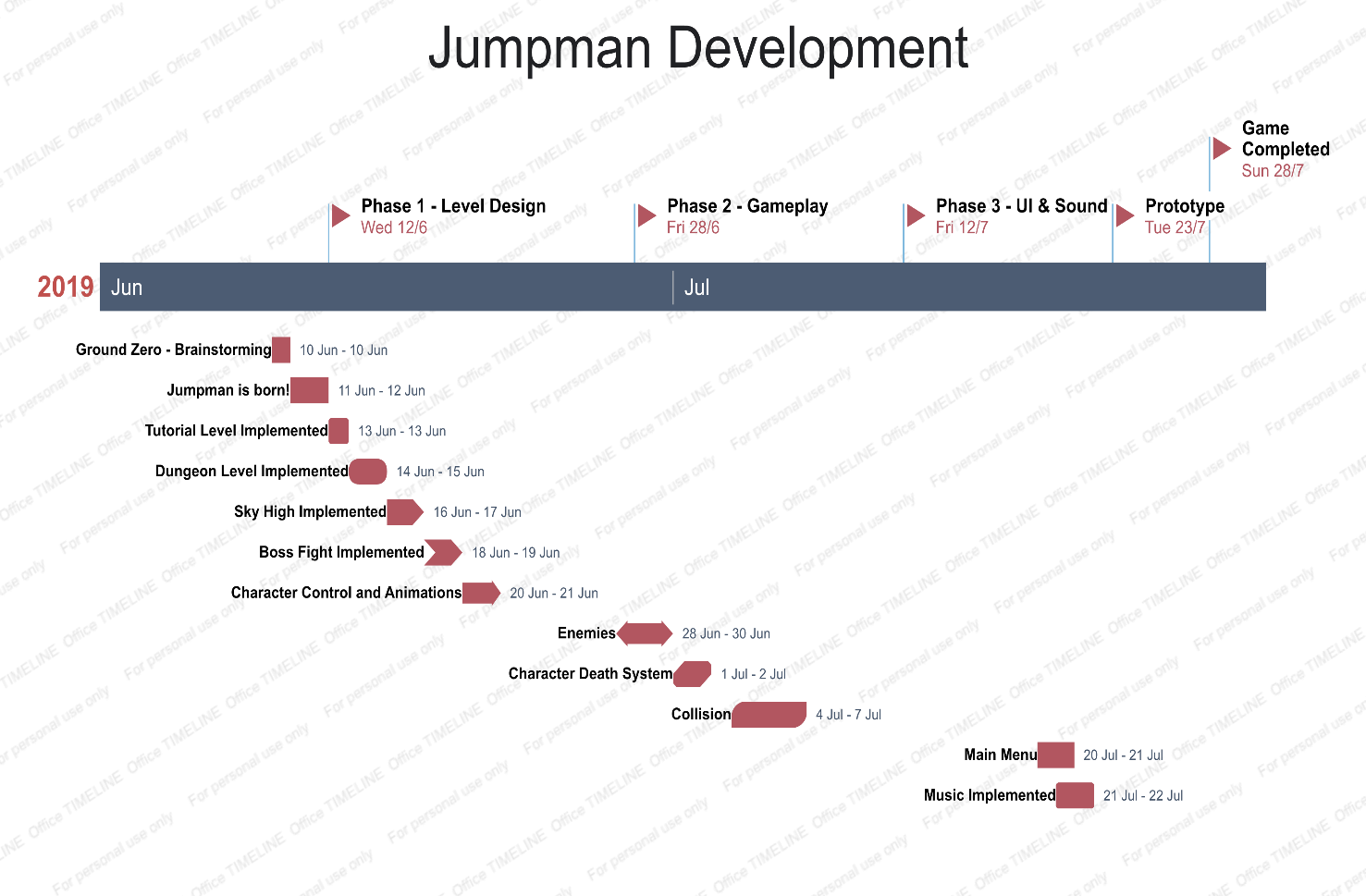
t\_LevelManager.AddCoin ();

Destroy (gameObject);

}

}

}



Credits:

<https://www.youtube.com/watch?v=xYRA3uAola4&list=PLiRrp7UEG13Zsh4-Ir54fFoF7ATm540SL>

<https://www.youtube.com/watch?v=M-1-XxO-Fyo&list=PLZ1b66Z1KFKgm4QrzZ11jfaeHVaWHSHW7>

<https://www.youtube.com/watch?v=BdlL5bwbCiI>

<https://www.nesmaps.com/maps/SuperMarioBrothers/SuperMarioBrothers.html>

<http://www.mariouniverse.com/sprites-nes-smb/>