**3DGP Final Project Report**

**Game Title:** *Idle Game - Loli Adventure (小蘿莉大冒險)*

**Member List:**

0856610 郭聖弘

0856727 洪冠群

**Introduction:**

The game we design is an 3D idle game. This is a mobile standalone game on Android. Players can choose different combination of skills and equipment to pass the levels. Then, the character will auto-fighting in different stages even if the player closes the game. Thus, you can play the game in a few minutes, and then close the game and put down your cellphone. When you open the game next time, you can get (offline) rewards with stronger levels and more coins. We have two kinds of levels: Forest Level and Dessert Level. Each consist of 3 stages including 3 normal scenes and 3 boss scenes. After you defeat the final boss, you will pass through the game.

**Game Story:**

Loli-Chan accidentally travel in a fantasy world. There are many monsters never seen on Earth. To find out the way home, he started to wander and met some friends. They told Loli-Chan that the way home is in the den of the dragon boss. Therefore Loli-Chan has to defeat the dragon boss. To improve his fighting skills and get more powerful equipment, Loli-Chan embarks on a difficult journey and keeps fighting day and night……

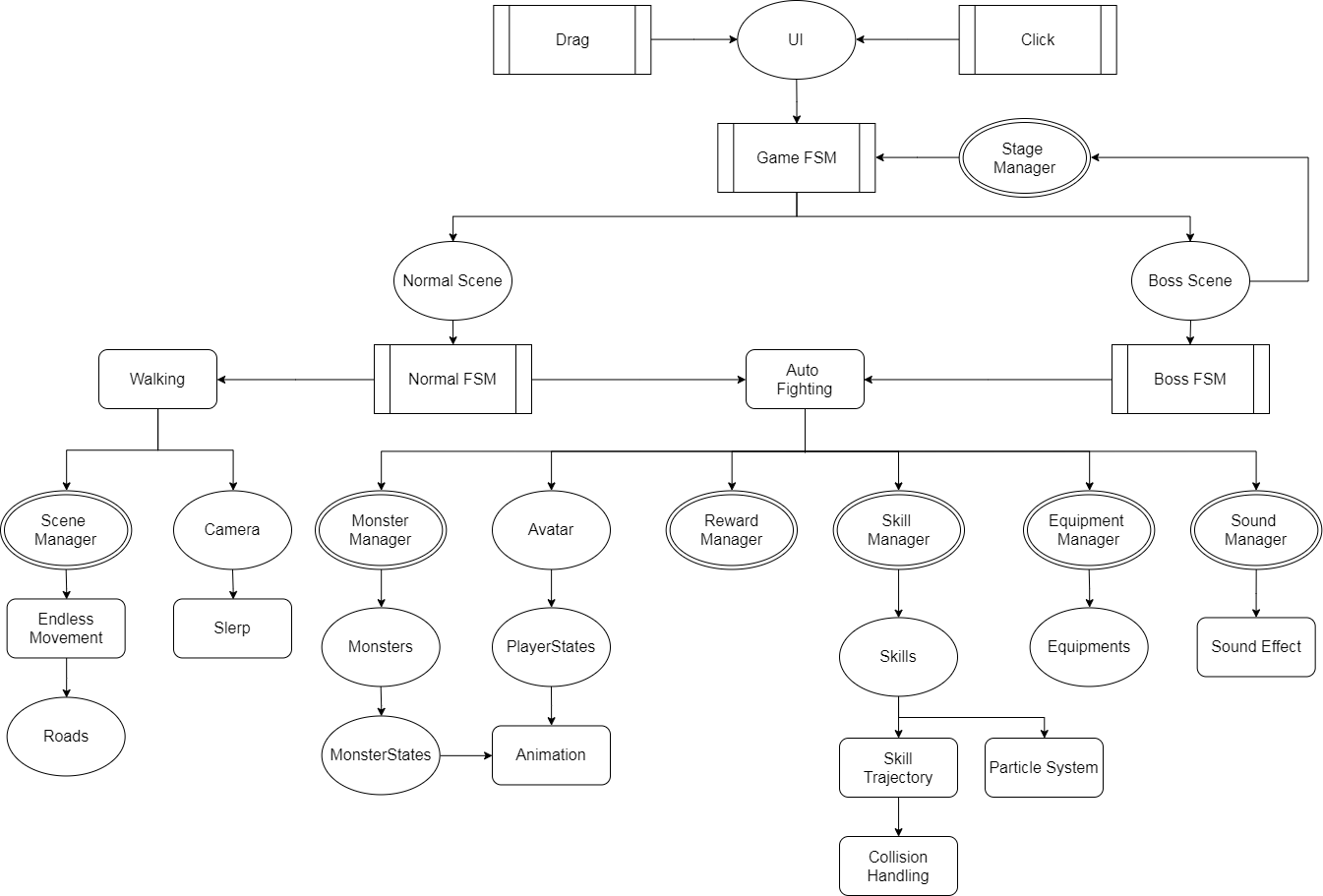
**Game Control:**

Click or Drag on screen.

**System Configuration:**

Android 10.

**Game System Architecture**

****

**Game System component description**

Four types of components:

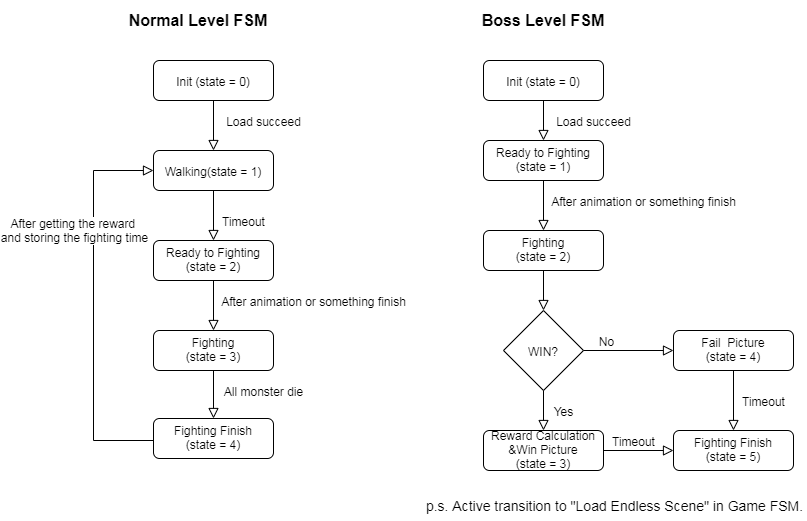
* + 1. *Ellipse: game object.*
    2. *Rounded rectangle: game mechanism.*
    3. *Process rectangle: game control*
    4. *Double ellipse: manager*

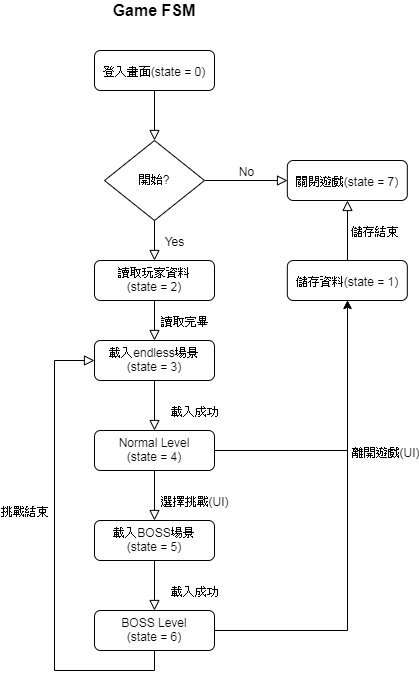
We have two input control: ***Drag*** and ***Click***. By clicking some ***GUIs***, we can control the ***Game FSM*** to load some scenes, according to ***Stage Manager*** which manage player’s current stage. We have two kinds of scene: ***Normal Scene*** and ***Boss Scene***.

In ***Normal Scene***, player will keep fighting against monsters to be stronger. ***Normal FSM*** will control two game progresses: ***Walking*** and ***Auto Fighting***. In ***Walking***, ***Scene Manager*** will keep ***Roads*** move and generate/destroy automatically. Also, the position of ***Camera*** will ***Slerp*** when changing the progress. In ***Auto Fighting*** progress, on the other hand, ***Avatar*** will fight against ***Monsters*** which are managed by ***Monster Manager***. Both will bind with ***PlayerStates*** and ***MonsterStates*** which store the attributes of them. In addition, we have ***Skill Manager*** and ***Equipment Manager*** to handle ***Skills*** and ***Equipment*** which player can choose based on his game strategy. For ***Skills***, we use ***Particle System*** and solve ODE to simulate the ***Particle Trajectory***. We also do ***Collision Handling*** between ***Particle System*** and ***Avatar/Monsters***. What’s more, we have ***Reward Manager*** to manage the reward player obtain, e.g. experience and money. To make the game more interesting, we have used ***Animation*** and ***Sound Effect***.

Next, we can click ***UI*** to change to ***Boss Scene***. In ***Boss Scene***, do something similar as ***Normal Scene*** except for ***Walking***. If player defeat the boss, ***Stage Manager*** will update the current stage.

**Technical Section**

**FSM:**



**Endless Scene**: Create a few roads prefab. And instantiate these prefabs. Then, move the instances. When a instance is out of player’s view, destroy it and generate a new instance.

**Scriptable object:** Write a script inherited from ScriptableObject. It can be seen as an asset in Unity like material or shader. Thus, it can be easily reused and shared between different scenes. To store the data, we also need to save it to disk by writing it as JSON format.

**Skill Journey:** Use acceleration to solve ODE to get the position of skill particles.

**Camera Slerp:** Use linear interpolation to position of camera. And spherical interpolation to quaternion of camera to simulation the motion of it.

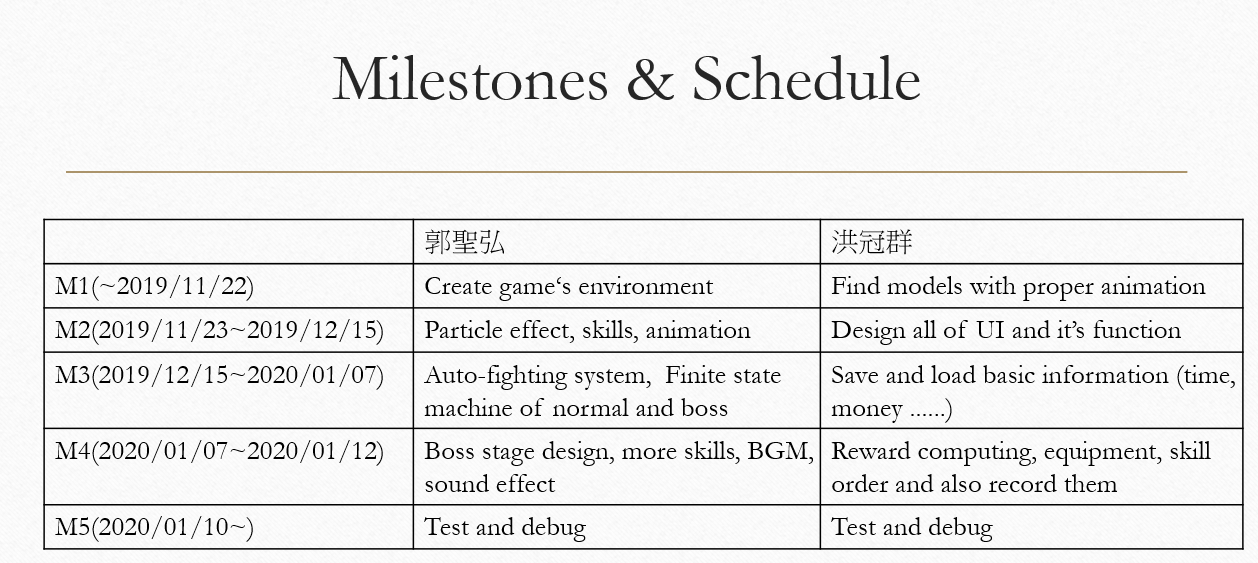
**Physics:** Cast ray up/down to check if hit a terrain. If yes, move the position to hit point. Also, use collider to check the collision on particle object. And compute the damage after hitting the target.

**Skill Change**: We use array to save the information (ex: acquire/not acquire, order) of all skills in scriptable object. When click the “change” in skill panel. First we pass it’s skill ID to skill manager and save it. Then show all of the skills that have acquired but not equipped. After player choosing the skill he/she wants, That skill will also pass it’s skill ID to skill manager and then swap the index in order array(such as [1,0,2,3] means first position is skill numbered 1, second position is skill numbered 0 and so on……). Skill manager also swap the skill record which was attached in player attribute (the order will be used when battle). After dealing with UI and player attribute. Skill manager will save the information into JSON file again.

**Tasks and Contributions of member(s)**

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Tasks** | **Number of Lines** | **Contribution** |
| 郭聖弘 | Endless Scene.  Particle Effect.  Sound Effect.  Finite State Machines.  Skills.  Player Attribute. | 1117 (57.6%) | 55% |
| 洪冠群 | UI panel.  Equipment.  Skill order.  Reward Computing.  Save and Load. | 820 (42.4%) | 45% |

**Milestones**



**Discussion and Limitation**

|  |  |
| --- | --- |
| **Strength** | **Weakness** |
| i. Save time  ii. Control easily  iii. Sense of achievement  iv. Various strategies depend on players. | i. Sense of immersion is weak  ii. Degree of freedom isn’t high |
| **Opportunity** | **Threat** |
| i. Modern people are so busy that they only spend a little time playing games.  ii. Mobile game is mainstream. | i. More and more people contribute in developing mobile game so it is competitive. |

**Future Work:**

More scenes and skill effects. Especially, a wide variety of particle trajectory and effects. In addition, we want more animation of monsters and avatar. Also, we hope that player can choose various things to form more and more combinations.

**Conclusion**

We finish an idle game by Unity (Platform: android). Player can keep fighting in endless scene automatically. And we use 3 finite state machines to control the progress of the game. To let this game more interesting, we create skill, equipment and reward. We also use particle effect and sound effect to enhance the feedback and make the game more colorful.