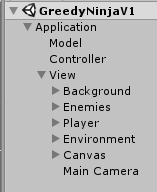
AC31009 Games Programming: Greedy Ninja Report

Adam Manion 140011547

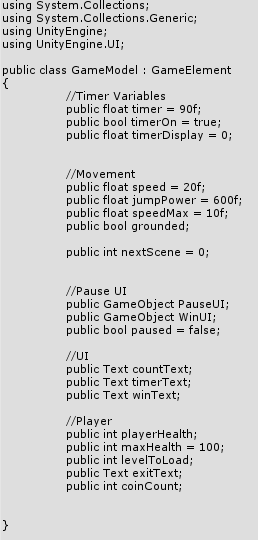
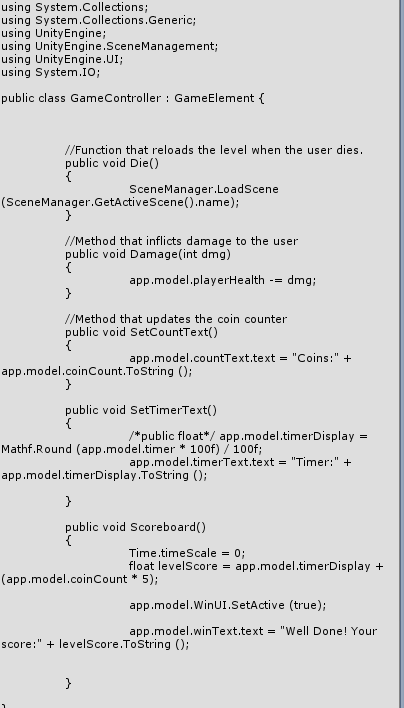
The game I have created for my Games Programming assignment is called Greedy Ninja. Greedy Ninja is a level based 2D platformer. The aim of the game is to collect as many coins as possible to advance to the next level. The game is fast paced and death is frequent if the player is not careful. Keeping the engaged to better their skills.

The game I have created is a lot different from my original plan from the first video where the game title was “Saving Sensei”. I tried to keep my game as close to my original idea as possible so the genre remains somewhat the same and theme is the same as before, you still play as a ninja. I say somewhat the same as my game is still a platform game I have however abandoned the side scroller aspect of the as I found when I was designing my levels they were very boring and dry and not at all fun to play. I have not decided to make a game that take a lot of inspiration game play wise from Super Meat Boy, where it takes skill to complete puzzling levels and I have found this makes for much more enjoyable gameplay. This was one of the main reasons for the switch up.

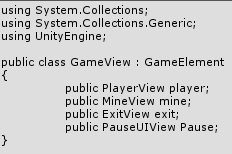
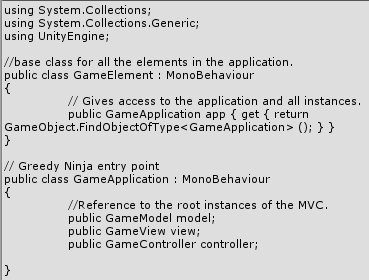
**Design Patterns**

****MVC Design Pattern

I have made use of the MVC design pattern in my game. Each scene in my game is organised with an application layer, model, controller and views. My GameApplication layer is my access point for the three variables Model, Controller and view. This essentially lets each component communicate with each other. The GameModel script contains all the variables and data used in the game. The GameController controls the workflow of the game. The GameView holds instances of every view related to the game.



*GameController.cs GameModel.cs*

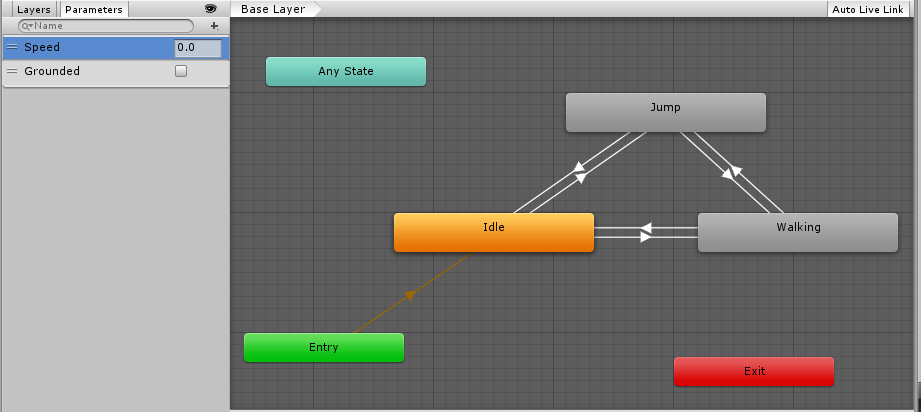


*GameView.cs GameApplication.cs*

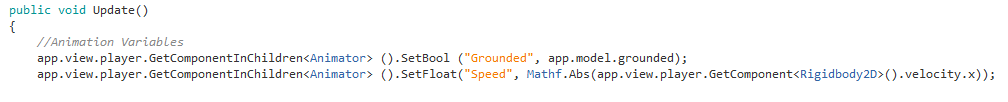
The advantages I found to implementing the MVC design pattern in to my game was that it was a lot easier to interact with variables across different scripts. Also, object referencing was a lot simpler since all the variables were in one place instead of spread out on individual objects. However, when implementing my Drone Movement, I was unsuccessful at coding this enemy as part of my MVC design. This was due to the way I calculated the start and endpoints, these points were two empty objects that’s transforms I referenced for the drone to move to, I did it this way so that I could simply change the start and endpoints freely in the scene. In an MVC design I was only able to create one drone due to the start and endpoint being two single variables in the GameModel.

State Design Pattern

The State Design Pattern is a behavioural design pattern that alters an objects behaviour when its state changes. My game makes use of the State Design Pattern in the animation of my playable character. My Characters animation makes use of a finite state machine and has three states, Idle, Walking and Jump(In air). This implementation removes the dependency on the if/else or switch case conditional logic. Another advantage to the state design is that if I was to go back and add more animations for example a death animation I would simply create a fourth state and add conditions on too my new state without editing a thing in any of my current states.



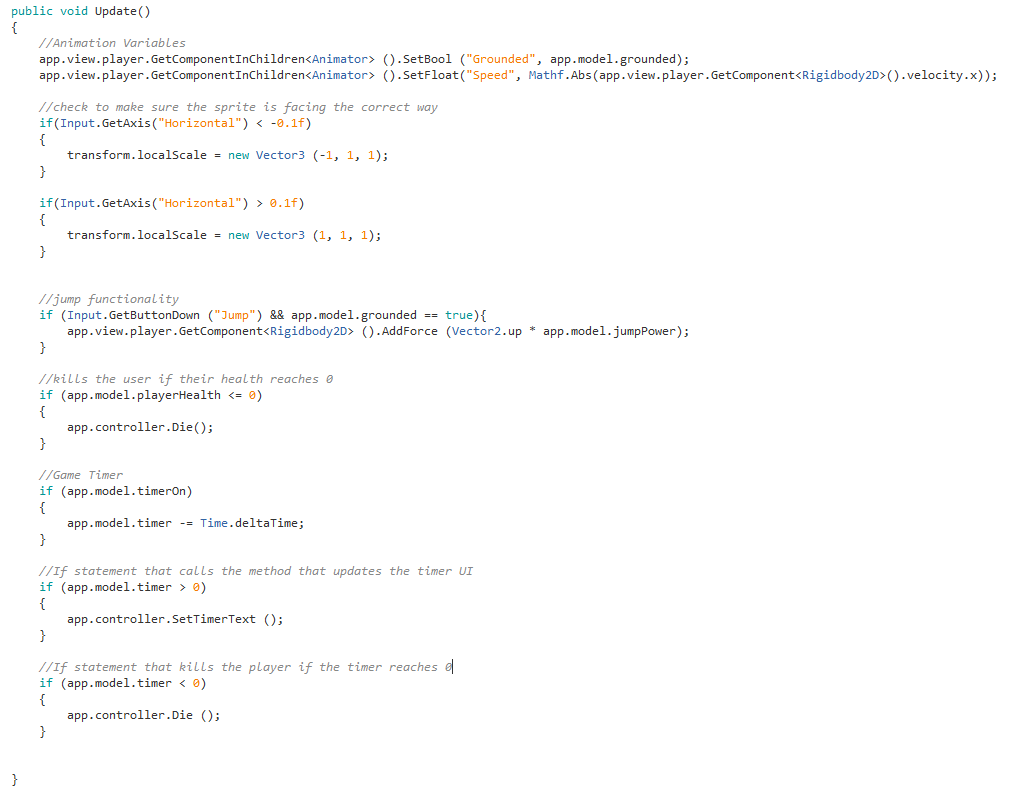
*Visualisation of my finite state machine*



My grounded (bool) and speed (float) variables for my animations. Speed is linked to the players rigidbody2D for smoother more accurate transitions between states.

Game Loop Design

The game loop is the heart of pretty much every game, it’s a controlled infinite loop that keeps the game updated and visible to the player. It will continuously run as the game is played. My game makes use of the game loop design with Unity’s prebuilt methods Update(). In this method I have my players jump functionality, a check for the direction my player sprite is facing, a timer, a check for the timer that kills the user if it reaches zero.

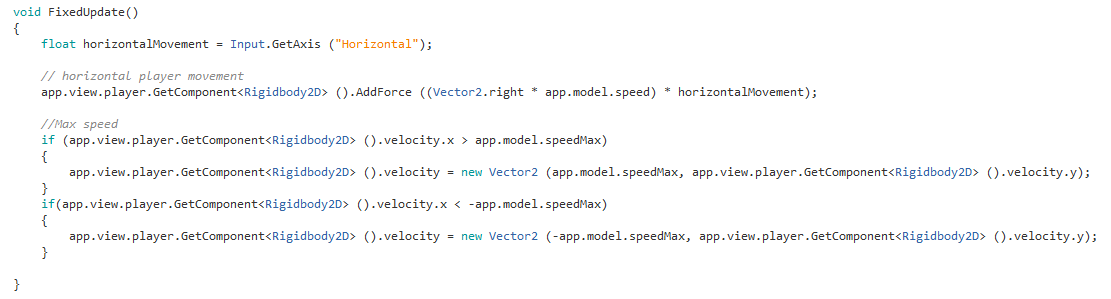


*Update function in the PlayerView.cs*

**Required Aspects**

Physics Engine

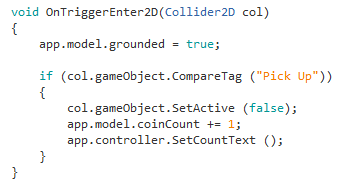
One of the required aspects I decided to implement in to my game was the Physics engine. My playable character makes use of the Rigidbody2D which allows me to apply physics to my player. For player movement, I apply forces to my characters rigidbody to move the character left right and give them the ability to jump. My character is affected by the forces of gravity e.g if he jumps he will come back down.



*My player movement with forces being added to the ridgidbody2D and the max speed IFstatements in the PlayerView.cs*

Collision Detection

The second required aspect implemented in my game is Collision Detection. This was used for multiple parts in my game, enemy attacks and environmental hazard collisions. This is one of the fundamental aspects to my game as without collision detection my player would at the start of the game fall through the floor and the game would essentially be over. Also, collision detection is important in enemy and environmental hazards as the player couldn’t die without it! Collison detection allows the player to “pick up” coins by as shown in the code below.



*Method that detects when the player collides with an object with the tag “Pick Up”*

**Security and Ethics**

Security

When developing my game, I found no areas in which any real need for any extra security in my game. My game is an offline single player game so there is no chance for players to gain an unfair advantage over other players that could ruin their gameplay experience. The source code will also not be released with the game so for those reasons I haven’t implemented anything extra security wise.

Ethics

When creating my game, I took ethics into consideration, I made sure not to stereotype or use any material that could be racist. I have only used free assets and materials and have in no way tried to take credit for anybody else’s work. My game is suitable for all ages as there is no adult themes or visual acts of violence. The player can die in different ways, stepping on mines, jumping into sawblades and falling in to an acid vat to name a few however as I didn’t draw my sprites and the sprite sheet I used didn’t include any death scene I couldn’t implement any visual violence.

**What I would do differently**

If I was to do this exact project again I wouldn’t have spent as much time as I did at the start looking for assets and graphics. Looking back, I now understand that I was too focused on the all the wrong parts of the design of my game. I feel like I would be a lot more comfortable using the development environment the second time around meaning I wouldn’t fall into as many beginner traps that I did this time. However, if I was given this assignment again I don’t think I would pick the same genre of game to make, I think I would either make an adventure RPG type game or I would try my hand at a game in Unity 3D.

**Testing**

When creating my game, I frequently and repeatedly tested my code checking for errors and finding solutions. One of the tools I made use of most when developing the game was the debug.log. To test I would insert messages in methods where errors were occurring and analysing the messages being received to help locate a correct mistake in my code. I also made use of the built-in error messaging and breakpoints to inspect exactly what is happening during runtime.

During the playtesting day people played a demo of my game and gave me feedback based on their experience. One player noticed my pause menu wasn’t active during the game, with further investigation I found I had mapped the PauseUI with a custom hotkey on PC but made the build on the lab computer where the hotkey had not been configured, this has since been addressed. Another player found that you could still control the player on the win screen where they unfortunately hit a trip mine and restarted the level. This issue has also been fixed. Overall the testers found my game to be smooth to play and had no real issues, players did comment on slightly laggy animations which I have improved but I as I didn’t draw the sprites myself I couldn’t perfect them.

**Improvements**

There are some features I wanted to add but unfortunately ran out of time. The first thing I would have liked to add would be more enemies I had ideas for about four or five extra enemies including turrets and chasers. The second and final big improvement I wish I implemented was a scoreboard. Additional features such as sounds and music would also have been nice. I feel a lot of my time issues were due to my change of heart on the direction I decided to take my game. The poor planning at the start left me with a bigger task and I felt I was forever chasing my tail. So my final improvement for this game would be better initial design planning.