COMP280 Optimisation

Daniel Neale

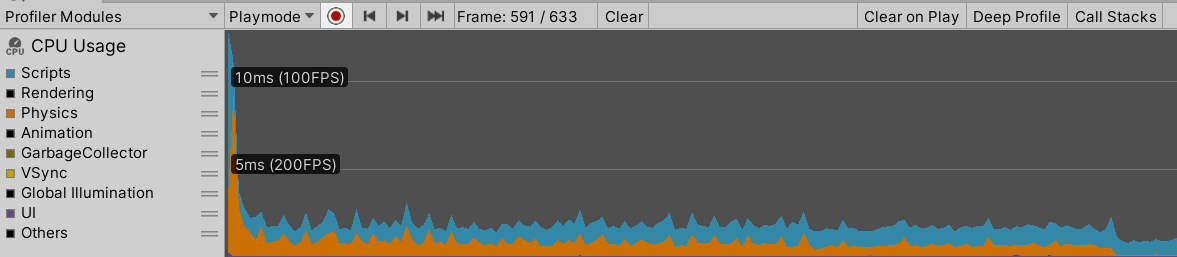
# Introduction

For this worksheet I’ve decided to optimise a game I helped make for a game jam. In the game you play as a tumble weed who has to pass through various western themed levels. The overall goal is to get to the end of the level without being killed by any of the resident cacti and the player wins once they have successfully passed through all of the levels.

As I worked on this with other people I am only going to be focussing on my contributions to the project. The specific systems I worked on are the player controller, camera controller, wind system and timer system as well as other minor systems however I should only have to focus on the main systems.

The game files can be found through [this link](https://github.com/Mistiare/PGJ-Untouchable).

# Baseline

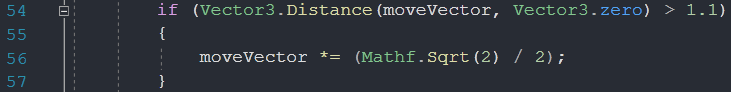


This is a baseline for the project and at first look it’s already fairly optimised so shouldn’t need any major reworks. The only major spike is when loading into a scene which is to be expected however I will still look into start processes to see if anything is majorly affecting it.

My plan going forward is to look at each individual system and evaluate any issues that could be occurring with them. From there I can patch them out and run the profiler again to see if performance has improved. I doubt I’ll be able to improve it by any noticeable amount however it’s still worth carrying on to see how efficient I can get the systems.

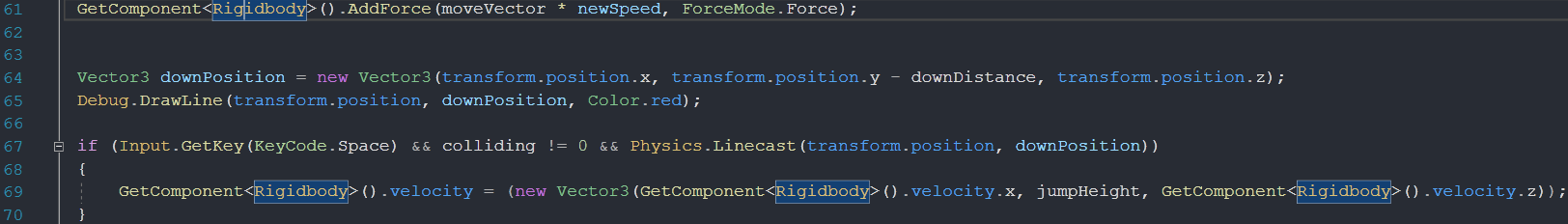
# Player Controller

The player controller system is fairly basic and only consists of movement as the player only needs to get to the end of the level also considering you play as tumbleweed we didn’t want to make the player too complex. Thusly the player controller is just in charge of moving forward and backward, strafing left and right and jumping.

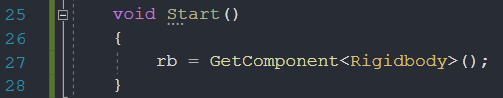


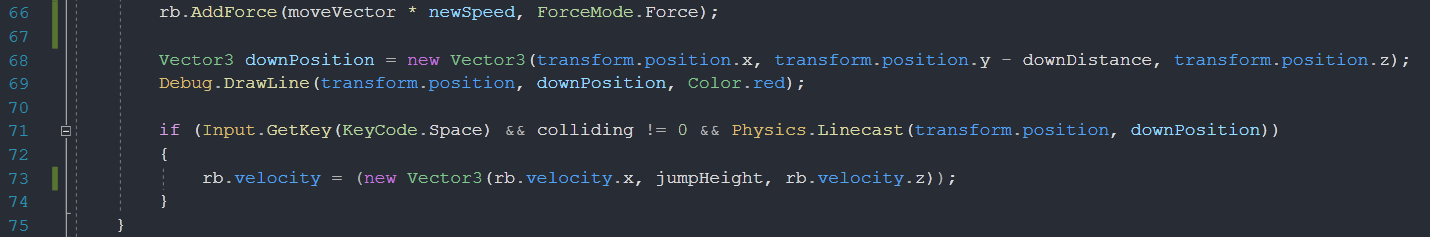
The first issue I’ve identified is this check to see if the player is moving at an angle. I did this check so I could then normalise the movement vector. On reflection, Unity has a normalise function so I can just run this instead of the check which should be slightly more efficient and will be much easier to read.





The next issue with the use of GetComponent in update as well as in other areas of the code which are commonly run. By removing these I should make the code a lot more efficient as I’ll on longer be getting the component several times a frame.

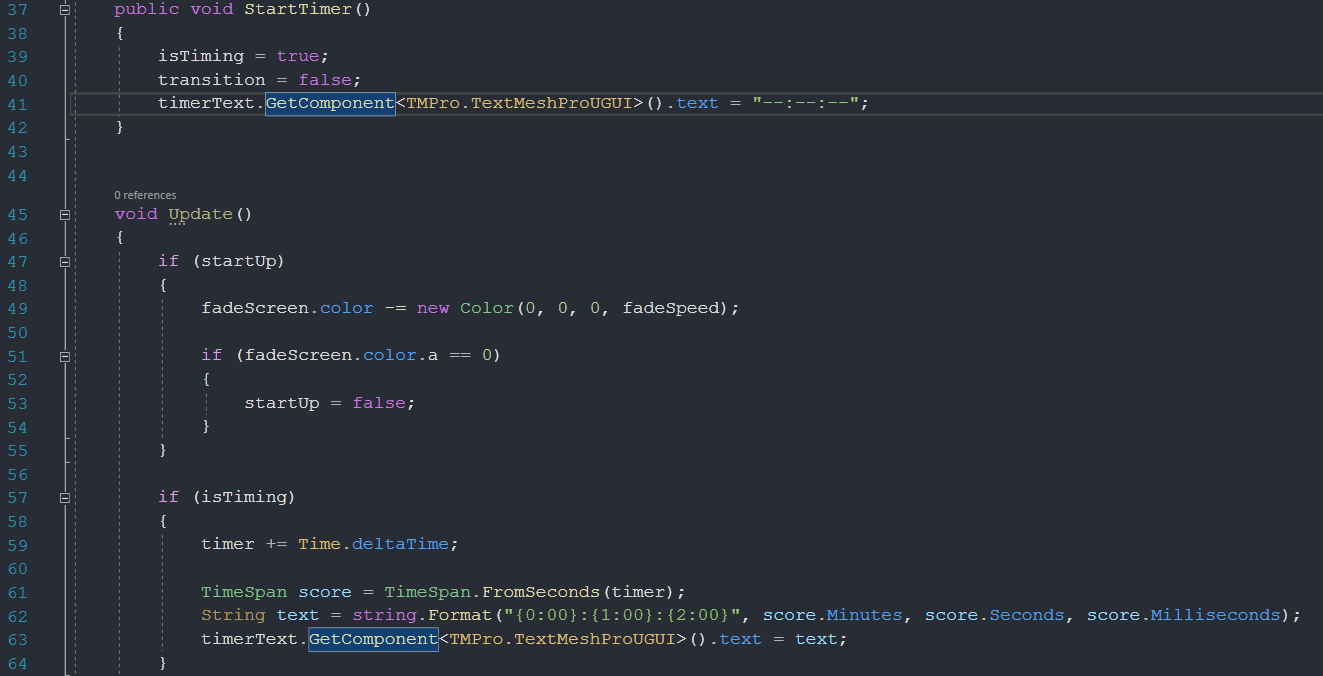




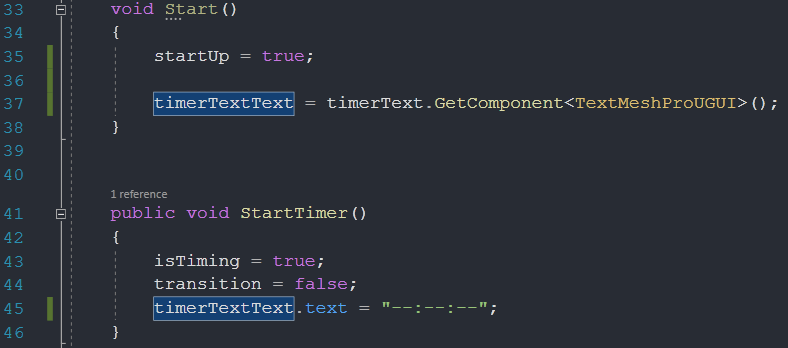
I’ve added a reference to the rigidbody that the script grabs on start that it then uses elsewhere in the script. This, as far as I’m aware, was the main issue with the script and so it should be running better, however I doubt it will make much of a noticeable difference.

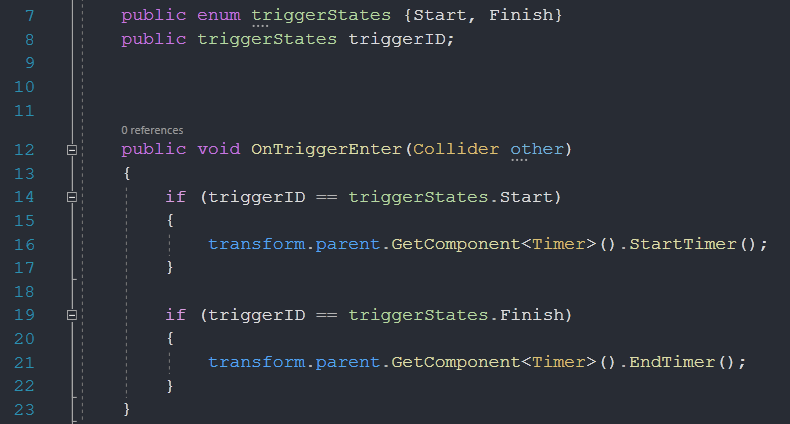
# Timer Controller

The timer controller is a very simple timer system that times the player between the start gate and end gate of a level. The timer is displayed to the player in the corner of the screen and then saved to player prefs at the end of the level.

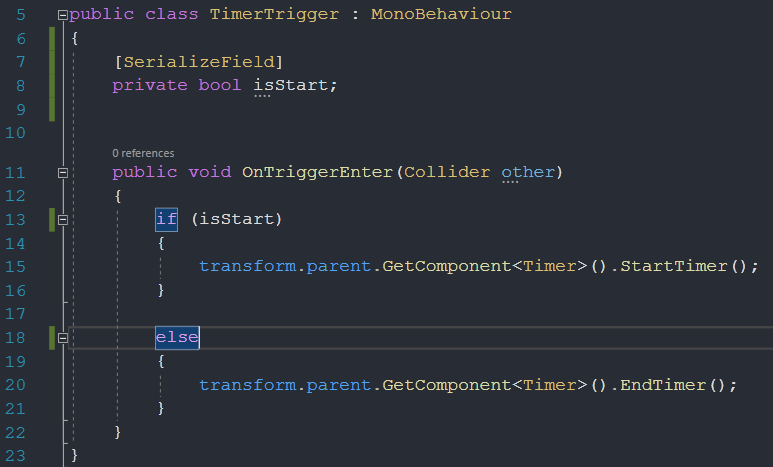


The main issue with this script is the use of GetComponant in update again which can be fixed by using a variable to store it which I’ll declare in start. This like before should make the script slightly more efficient however shouldn’t incur any noticeable change.



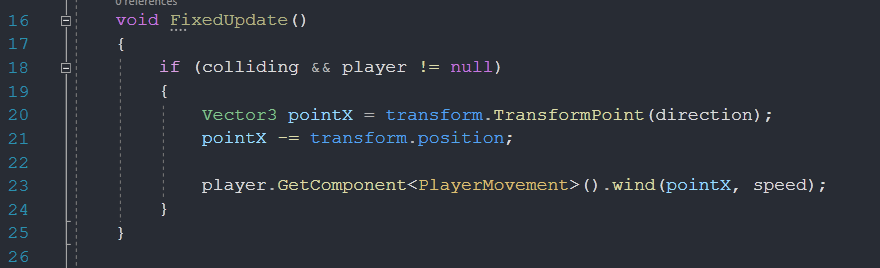


This is the system for detecting when the player goes through either of the gates to start and stop the timer. The only issue with this system is that I’m using enums when I could be using a Boolean which should save on ram and make the system slightly more efficient.



# Wind Controller

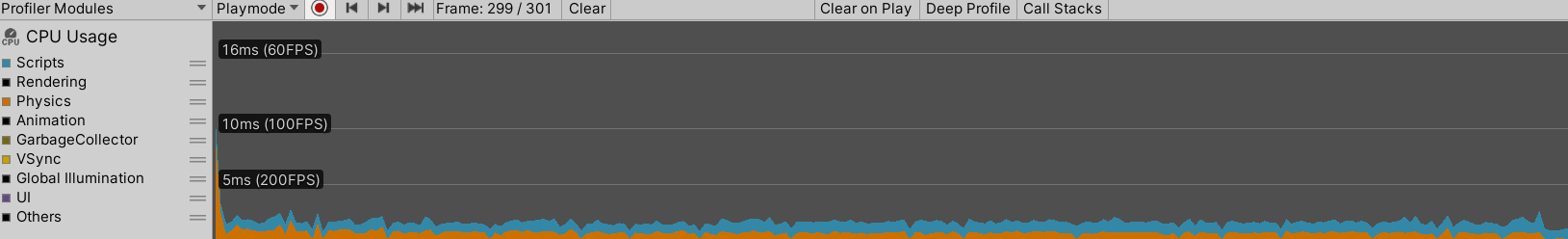
The final major system I created for the project was the wind controller which creates sections of directional wind placed around the level. The script is attached to a collider which when the player intercepts, sends the direction and strength of the wind in the form of a Vector3 to the player which then adds the appropriate force.



The only issue with this script again is with getting components in update so I will declare a new variable in start to then use.



# Conclusion



This is the profiler run after all of the changes above have been made. To my surprise there has actually been a noticeable difference in the profiler to what the readings were before. This still hasn’t translated to any noticeable difference in game however there wasn’t any lag beforehand so it wouldn’t be noticed anyway. Overall it seems the average frame time has gone down by about 0.5 milliseconds and the frame time when loading scenes has gone down by about 3 milliseconds.