

Optimisation

COMP280 Worksheet 3



1904179

## Introduction

For this assignment I decided to optimise my GAM130 project game Lumin.

The original project repository link:

The updated project repository link:

## First Optimisation – Compass Script

The first thing I did after opening my project was start playing the first level. Straight away I noticed there would be a performance spike in the CPU usage whenever the player moved, more specifically when I changed the direction which the character was facing.

When I looked at the main thread the performance spike appeared to be caused by the compass script.

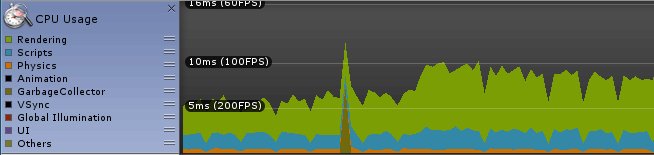


Figure 1 Spike which appeared after changing direction.

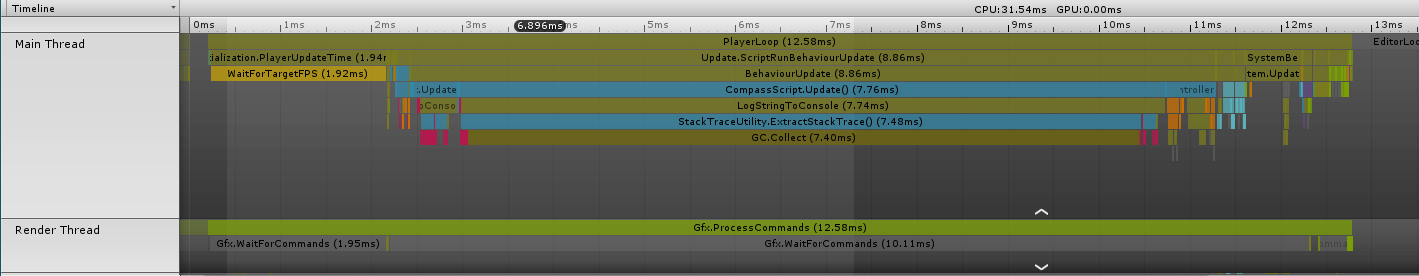


Figure 2 Breakdown of Spike from direction change.

Opening the CompassScript.cs file, I saw that the GetComponent was being being used within the update class. I removed the GetComponent from the update class. I wrote a Start class and new line of code so that the………

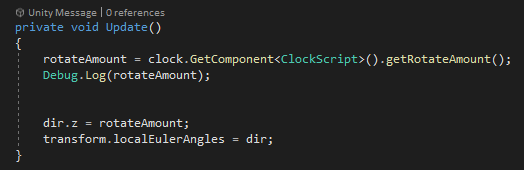


Figure 3 Original code from ClockScript.cs

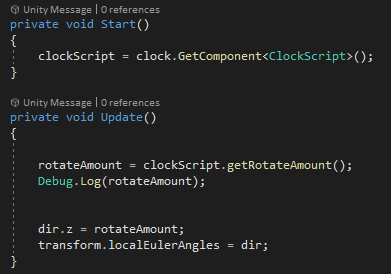


Figure 4 Code after I made changes to ClockScript.cs

* Now that I made changes to the code, I compiled the script and played the game again. This time as I moved around the map, I noticed a spike again however the number had decreased from 7.76 to 7.49.

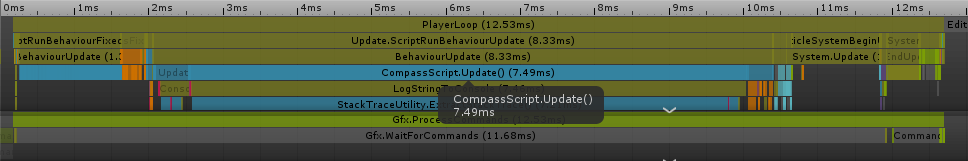


Figure 5 CompassScript.cs now 7.49ms

## Second Optimisation – Clock Script

When I explored further into the game, I noticed that there was a spike which repeated every few seconds. When I looked to see what was causing the spike in data performance, I saw that it was the Compass Script. Every time that the sun on the compass completed a 360 degree turn the spike would appear.

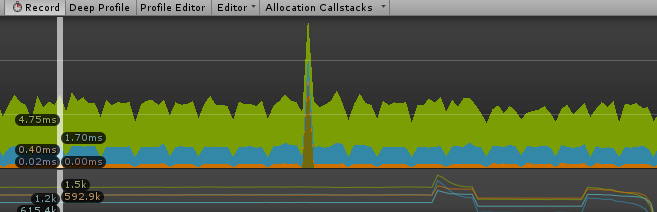


Figure 6 Spike when the compass completes a rotation

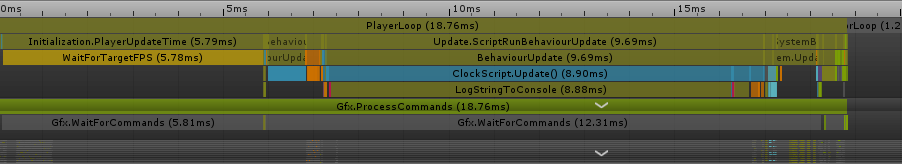


Figure 7 Breakdown of data from compass related spike

Opening the CompassScript.cs file I immediately noticed that a variable ‘daysPassed’ was not being used. I removed the variable and 2 other references of it in the script, compiled my work and then played the game again to see if this made any changes.

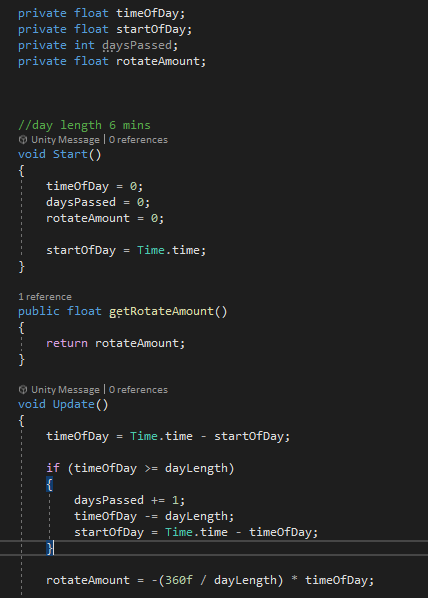


Figure 8 Original Code of ClockScript.cs

When I saw the clock complete it’s first turn and noticed the spike I checked to see if the performance spike had decreased. The spike had been decreased, from 8.90ms to 4.40ms.

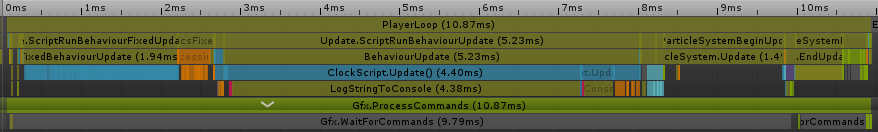


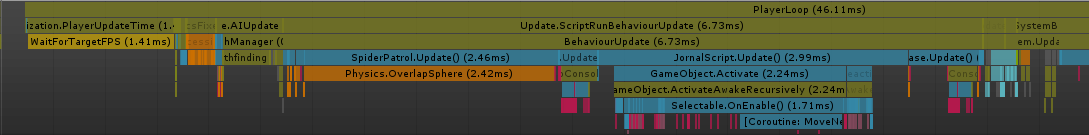
Figure 9 Decrease after removing redundant variable from code

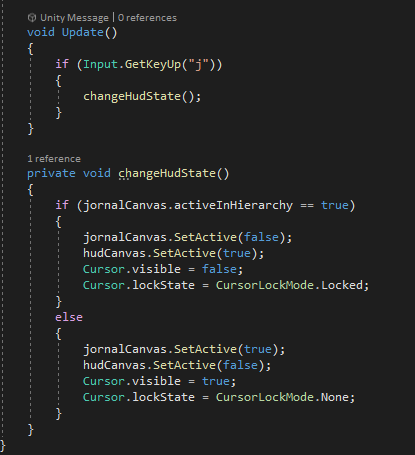
## Opening Journal

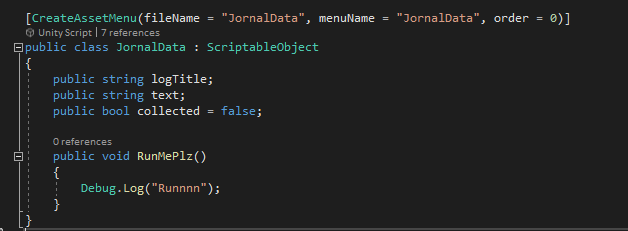
I noticed another spike whenever I would open the journal whilst in game.

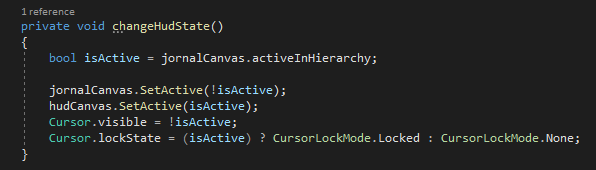


Figure 10 Spike in CPU Performance and Memory from Opening Journal

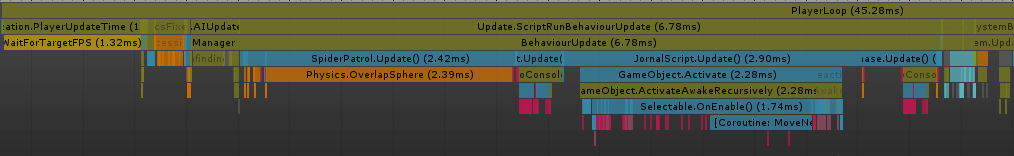






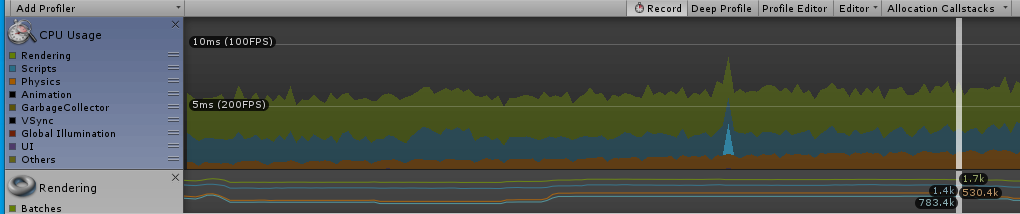


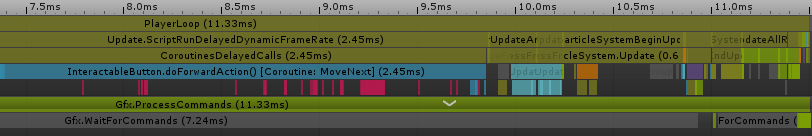
I opened the script which was apparently causing the performance spike and removed the if statements to shorten the code and hopefully make it faster.



## Fourth Optimisation – Interacting with Buttons

Whilst interacting with a button to open doors within the game, a small spike kept appearing.

The spike appeared to be caused by a script called ‘InteractableButton.cs’.



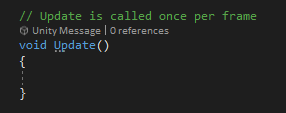
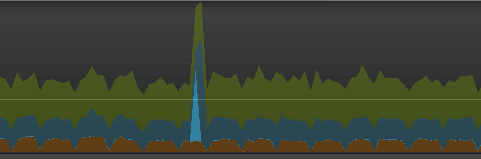
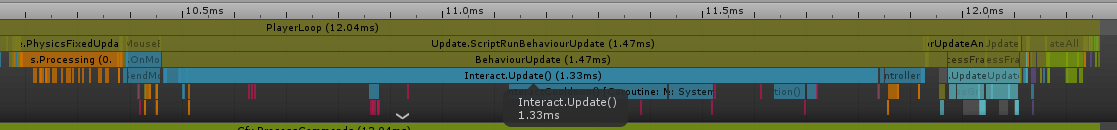


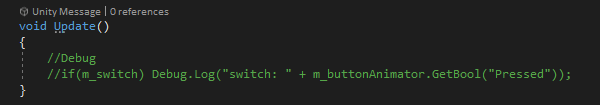
Figure 11 Empty Update Class in InteractableButton.cs File

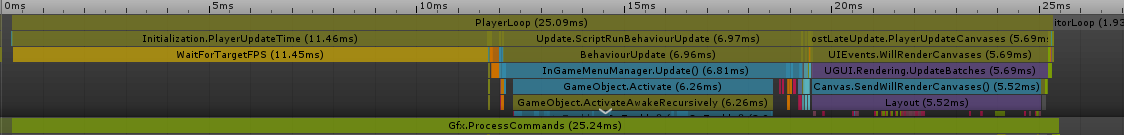
I found an empty update class which I removed as it was redundant, hoping this would fix the issue.

When I went back into the game to see if this had fixed the issue I saw that there were still performance spikes whenever I interacted with buttons. This time however the performance spikes seemed to be caused by another script, ‘Interactable.cs’.



This script also contained an empty update class which held no code. I removed the class, compiled and then played the game again to see if this would amend the issue.



When I interacted with buttons again the performance spike had significantly shortened.

## Fourth Optimisation – Object Pooling

## Fifth Optimisation – Culling