

Optimisation

COMP280 Worksheet 3



1904179

## Introduction

For this assignment I decided to optimise my first year GAM130 project, Lumin.

The original project repository link: https://github.com/kateastrophes/GAM130-Lumin

The updated project repository link: https://github.com/kateastrophes/comp280-worksheet-3

## 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 ‘CompassScript.cs’ file.

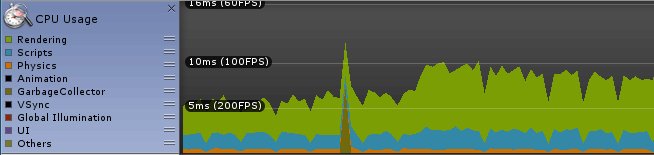


Figure Performance Spike which appeared after changing direction.

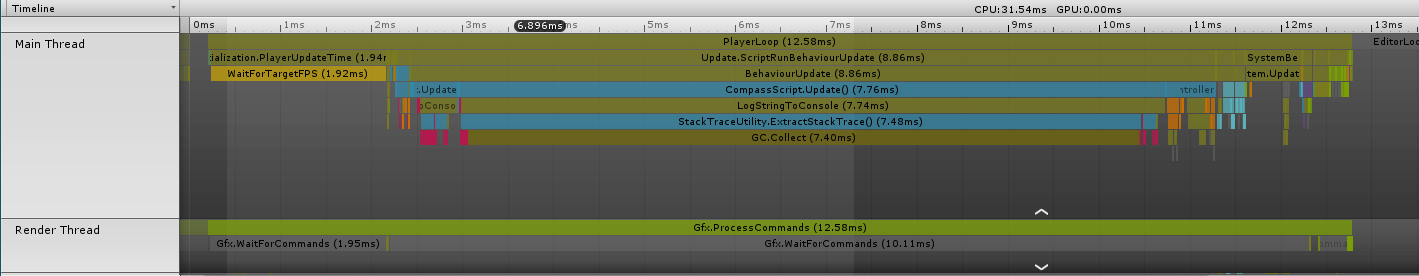


Figure Breakdown of Performance Spike from direction change.

Opening the ‘CompassScript.cs file’, I saw that the ‘GetComponent’ was being used within the update in the class. I removed the ‘GetComponent’ from the update function and wrote a Start function and new line of code so that the ‘GetComponent’ only runs once rather than every frame.

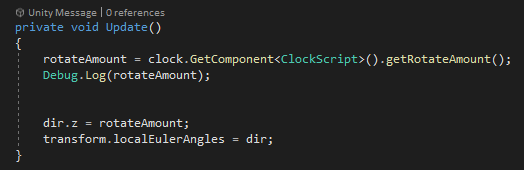


Figure Original code from ClockScript.cs

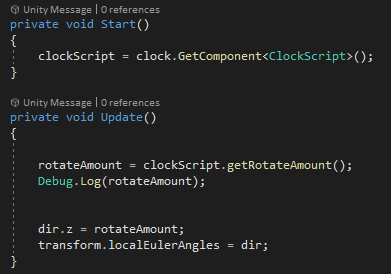


Figure Code after I made changes to ClockScript.cs

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 another spike however the time had decreased from 7.76ms to 7.49ms.

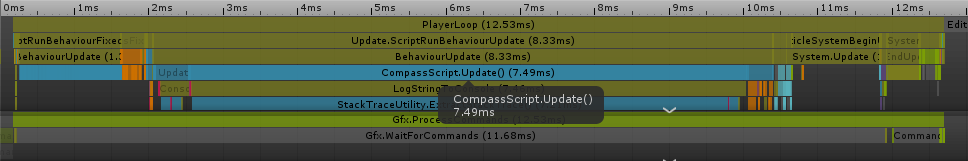


Figure CompassScript.cs now 7.49ms

## Second Optimisation – Clock Script

When I explored further into the game, I noticed that there was a spike which repeated every few seconds. The biggest issue I could see causing the spike in performance was the ‘ClockScript.cs’ file. As I was playing through the game to try and find other issues with the game’s performance, I worked out that the issue was appearing every time the HUD’s day tracker completed a 360 degree turn the spike would appear.

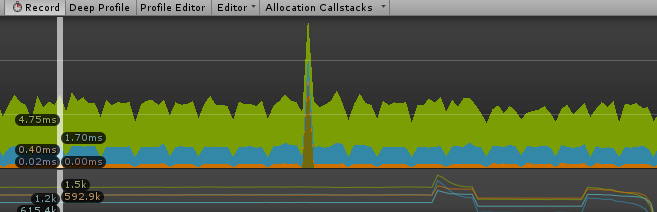


Figure Spike when the compass completes a rotation

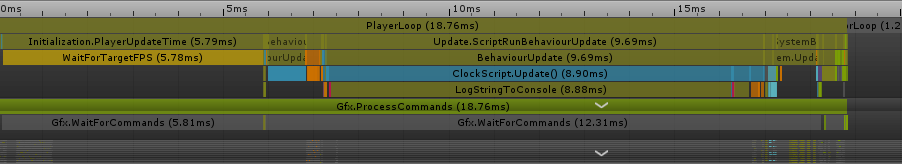


Figure Breakdown of data from Clock Related Performance Spike

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

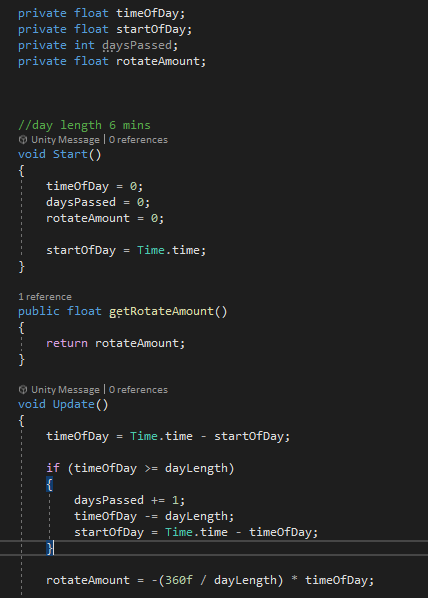


Figure Original Code of ClockScript.cs

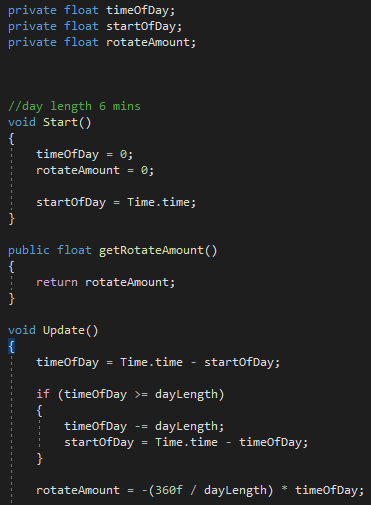


Figure Code After Removing Redundant Variable

When I saw the clock complete its first rotation and noticed the spike I checked to see if the performance spike had decreased. The spike was now significantly smaller and the time had decreased from 8.90ms to 4.40ms.

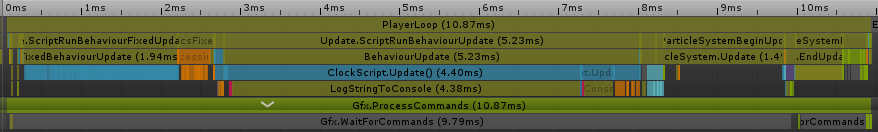


Figure Decreased Time After Removing Redundant Variable from Code

## Opening Journal

The next performance spike I noticed whilst playing the game occurred whenever I would open the in-game journal. Not only would there be a spike in the CPU’s performance, there would also be a spike in the memory usage.



Figure Spike in CPU Performance and Memory from Opening Journal

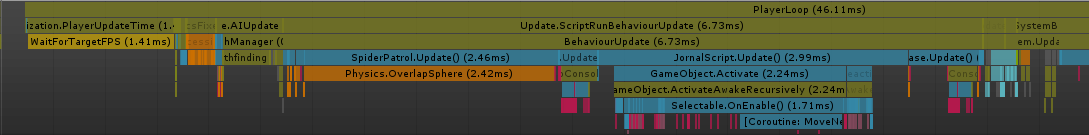


Figure Breakdown of Journal Related Performance Spike

I opened the ‘JournalScript.cs’ file and looked in the update function which was causing the issue. In the script a function called ‘changeHudState()’ was being called every frame. I removed the IF statements from the function to shorten the code and make the script run faster.

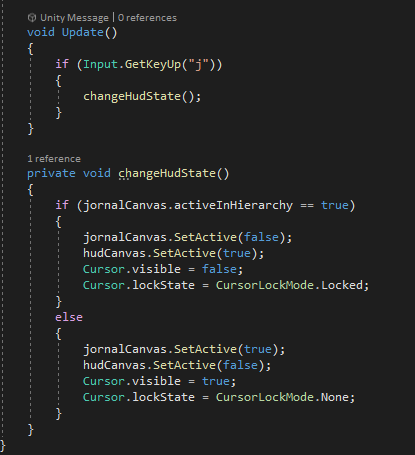


Figure Code Before Removing the IF Statements

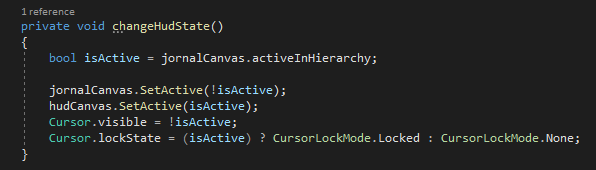
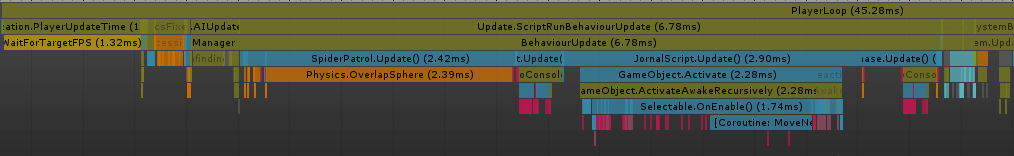


Figure Code After Removing IF Statements



## 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’.

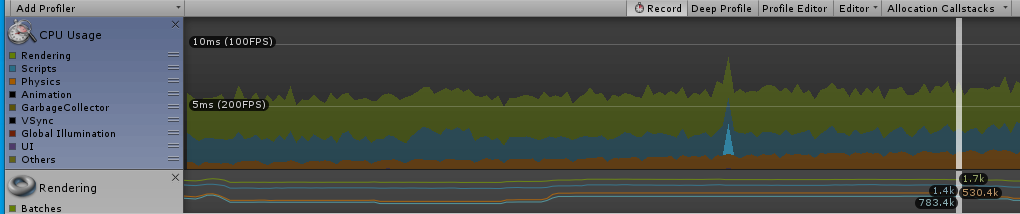


Figure First Performance Spike Noticed When Interacting with Buttons

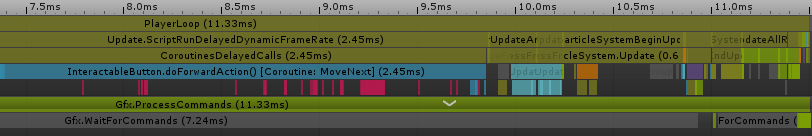


Figure Breakdown of Interactable Button Performance Spike

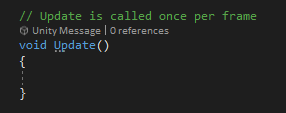
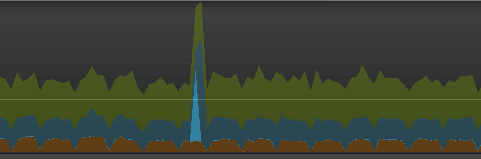


Figure Empty Update Class in InteractableButton.cs File

I found an empty update method within the script which I removed hoping that this would fix the issue and reduce the time it was taking the script to run.

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’.



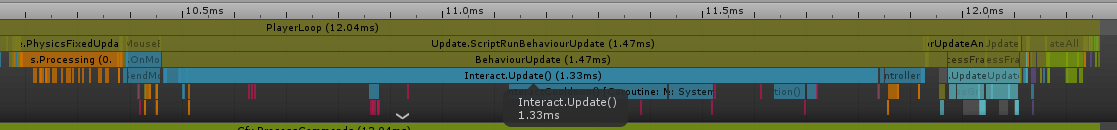
Figure Second Performance Spike from Interacting with Buttons

Figure Breakdown of Second Performance Spike Caused by Interactable Buttons

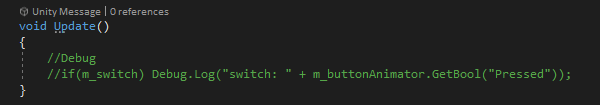


Figure Another Redundant Update Method

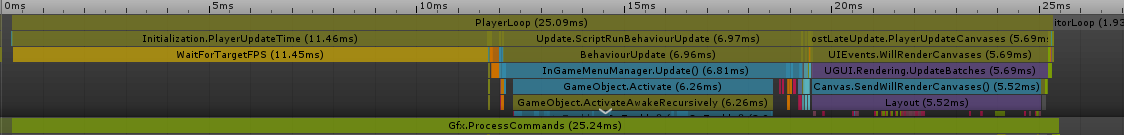
This script also contained and redundant update method. I removed the function and then compiled and then played the game again. When I interacted with buttons again the performance spike had significantly shortened. When looking at the breakdown of the spike I couldn’t find any issue being caused by either script.

Figure No Sign of Issues from Either Interactable Script