Optimisation for COMP280

Joshua Thompson

November 19, 2020

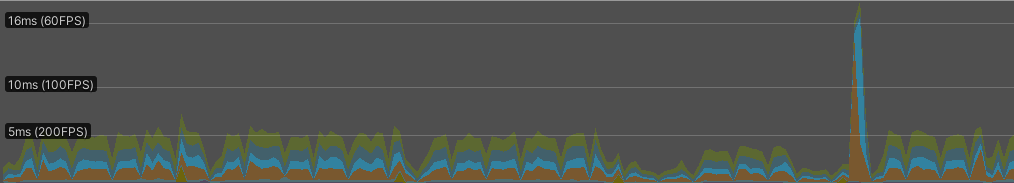
# Introduction

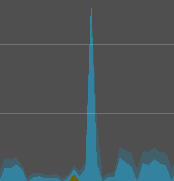
In this Journal I will record my findings and process’ taken to optimise my GAM220 project. The Github repository for which is referenced below.

**https://github.com/Alphon1/UFO-Mafia-Game**

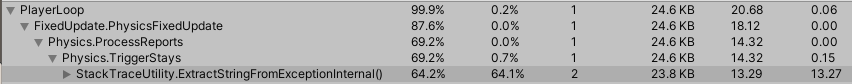
# An assessment of the current performance

I firstly needed to identify where any bottlenecks or hotspots were present within the game; to achieve this I used Unity’s in-built profiler on a built version of the game. Before I did this however, I wanted to familiarise myself with the profiler and turned to Unity’s official resources[7]. After some testing I noticed some significant spikes in CPU usage in relation to both physics and scripts.





To further identify the cause of this I assessed the hierarchy and found “Physics,TriggerStays” to be contributing largely to this however I was unsure as to what “StackTraceUtility” meant and decide to conduct some research. I quickly found that this was related to error messages and tracking the process’ through the stack that caused them[8]. This made a lot of sense as after a brief glance at the console there had been several error messages present.



# Resolving the errors

The most obvious error can be seen below:



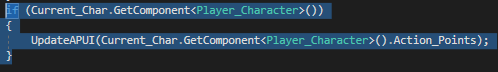
This was being caused by a script that was attached to both the enemy and the player called “Cover.cs” which should have only been utilised by the player, so I quickly removed this from the enemies prefab.

The next error I had to resolve as shown below:



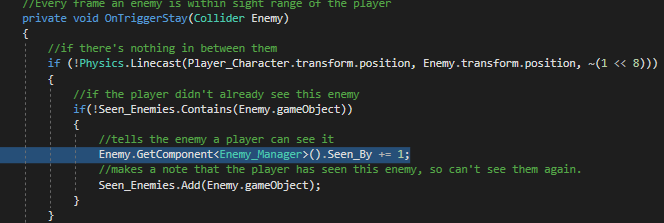


The problem appeared to be occurring when the action points UI was being updated on an enemies turn however they do not have a “Player\_Character” component so I added a check to make sure this was the case before the UI is updated. This is shown below:

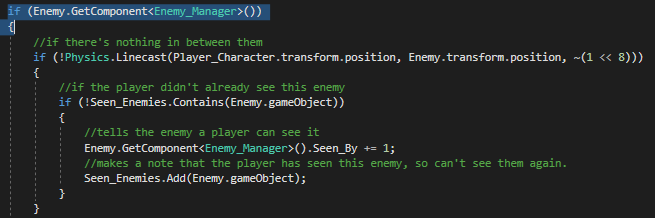


Lastly, I had a similar issue this time coming from “Player\_Vision.cs” I have displayed this below:



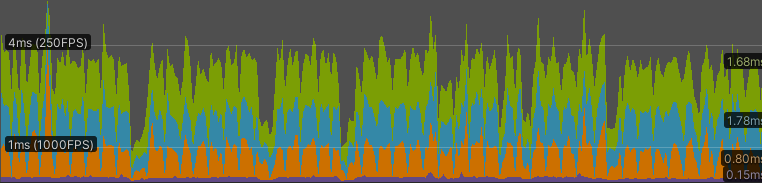


Once again there is no check to make sure that the “Enemy\_Manager” component is present on “Enemy” however unlike last time I thought it best to encapsulate the linecast as well as there was no reason to draw this without the component being present. You can see my implementation of this below:

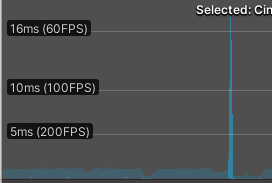


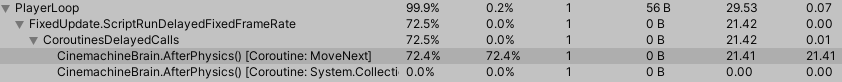
# Reassessing performance

As you can see from the graph below although there are still some inconsistencies the significantly larger spikes I identified previously are no longer present.



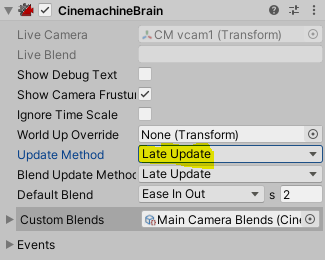
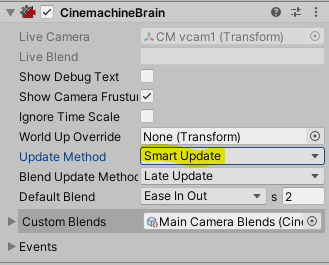
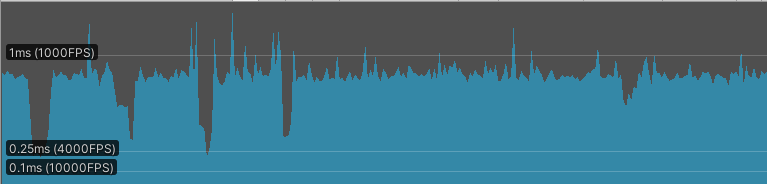
Upon further testing I found another spike of similar caliber to the one found previously. This time the cause was very clear as “Cinemachine” a tool for improved camera control was using an intensive coroutine.





# Addressing Cinemachine

After some research I found a thread on the unity forums where many others had encountered the same if not similar issues[9]. From what I had read either turning off unused cameras or changing the brain update mode to “LateUpdate” would resolve the spikes. As the latter was a quicker solution to test I opted to try this first.

This appeared to resovle the spiking as I had hoped and thus I didn’t see a need to turn off cameras that weren’t being used as this could create unforseen bugs.

# Conclusion

At this point I felt that a sustainable and more than adaquete framrate had been achieved thanks to the optimisations I had already made. Considering the target platform for the project is PC I tested on a below average PC for those that play lots of games and as such saw no need to optimise further for platforms with less processing power such as mobile or select consoles.

# References

1. Edsger W. Dijkstra. Go to statement considered harmful. *Communications of the ACM*, 11(3):147–148, 1968.
2. E. G. Gilbert, D. W. Johnson, and S. S. Keerthi. A fast procedure for computing the distance between complex objects in three-dimensional space. *IEEE Journal on Robotics and Automation*, 4(2):193–203, 1988.
3. Donald E. Knuth and Ronald W. Moore. An analysis of alpha-beta pruning.

*Artificial Intelligence*, 6:293–326, 1975.

1. Bui Tuong Phong. Ilumination for computer generated pictures. *Communi- cations of the ACM*, 18(6):311–317, 1975.
2. R. L. Rivest, A. Shamir, and L. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*, 21(2):120–126, 1978.
3. Alan M. Turing. Computing machinery and intelligence. *Mind*, 59:433–460, 1950.
4. Unity. Introduction to profiling in Unity Unite Now 2020 https://www.youtube.com/watch?v=uXRURWwabF4&ab\_channel=Unity, 24/5/2020
5. Erin Baez. Stack Trace: What is it and how does it help you debug? https://www.scalyr.com/blog/stack-trace-what-is-it-and-how-does-it-help-you-debug, 7/7/2020
6. ibbyn Cinemachine.AfterPhysics() [Coroutine: MoveNext] Performance. https://forum.unity.com/threads/cinemachinebrain-afterphysics-coroutine-movenext-performance.515069/, 31/1/2018