Optimisation research Journal

1505536

I. INTRODUCTION

Optimisation plays a vital role within development for video games. When producing a game developers aim to achieve the best performance they can. This is so important for making games feel responsive and smooth. Unity offers built in profiler tools which can be utilised to find issues and then techniques applied to fix them. There are several techniques which can be used specifically by programmers within the unity framework to get as much performance out of a game as possible. Knowing these techniques is not vital for development especially on a small scale game but as a game becomes larger they become more important. Depending on the work style adopted optimising can ether by utilised throughout the development of a game or at the end of a games development.

II. OPTIMISATION

Optimisation within games is the process making a software more efficient so that it runs smoothly and getting the best frame rate. This is important as it effects the users experience issues like low frame rate can make a game seem unresponsive. This usually occurs under 30 frames per second.Carrying out optimisation is important for the users experience and optimisations can be done within all areas of a game from the assets in a scene to the games code. Within video games there are several tools and methods for achieving this. One of the primary tools are profilers which monitor the games perfromance while it runs. These tools are often built into engines such as Unity and Unreals profilers. These are so important because they allow a user to find areas which can be optimised.

III. UNITYS PROFILER

Built into the unity engine there is a profiler window which users can use and has been used to find optimisations within COMP350s optimisation assignment. This tool is so important for optimisation because it shows areas which need to be optimised. It is important to know where to optimise as especially in big projects it can be a waste of time and resources to manually look for improvements.An issue may be CPU bound or GPU bounds for example and not knowing which may mean time spent making optimisations that do not help a game. When these changes are made the profiler is needed to see if a change to the game has in fact improved the performance or not.An important methodology in profiling is to run the profiler, analyses the results, make a change and then run the profiler again.The profiler runs along side the game and provides feedback on it in real time with graphs showing the performance of different components.Spikes in the graph show areas which are more costly for performance

then others. When using one of these profilers it is important to aim to replicate real world situations to get accurate results. The software being optimised for COMP350 is designed for computers so optimising it on a laptop is suffecient but for a mobile game it would be important to profile it on that device. Doing this is important so when the profiler runs you can see the performance as users will experience it.

IV. LOOP UNROLLING

One technique for optimising code is loop unrolling. Loop unrolling is where instead of contantly using loops that have to update the loop state each iteration manually calling functions. With static loop unrolling a user knows the length of an array and so can manuually call the functions they want easily. With an array of unknown size a just in time compiler can decide if a program uses a normal loop or make a sequence of individual instructions. With loop unrolling it is best for simple peices of code code that has lots of branches is also not suitable for this as it can make it even slower.Within the optimisation project this can be applied to the loop statement which spawns a certain amount of enemies. The amount of enemies which is spawned is a known amount so a static unlooping could be applied. This would detract from the readability of code so it would require comments to say what is happening but this trade off would be worth it for performance.

V. FREQUENCY OF FUNCTION CALLS

Calling functions takes up resources and can slow down a game causing frame rate issues.When calling functions one way to help optimise is look at how often a functions needs to be called and only call it that many times. This way the user wont notice a difference in game play and the computers resources are not being used on unnecessary functions. One example is calling a line tracer which takes up lots of memory because it is doing calculation every frame of the game. Within the COMP350 game there is a function which calls for enemies to pick a random location to walk to on the map. This was being called every second when it was not necessary, and as a result was wasting resources. Putting delays on how often functions are called can be done ij Unity by using the invoke keyword and selecting whatever time ou want.

VI. GARBAGE COLLECTION

Garbage collection is where data which is no longer required is removed from the memory meaning it is able to be used.Calling Garbage collection every frame is a waste of resources as it creates to much work for the device to do. It also means the impact of garbage collection is not as vital as it is only clearing with small amounts of memory. Without garbage collection however it can create a memory leak which

results in a game crashing so it is important. Currently in the COMP350 optimisation module my garbage collection is happening way to often which is dmaging its perfromance. In order to try and optimise this element to the game one method is to reduce the amount of heap allocations and deallocations. The heap is where long term memory is stored like objects you create.Object pooling can be used to reduce allocations and deallocations because it means destroying and spawning enemeis can be avoided. Currently the build of the game had debug logs in and these call in every version of Unity so these need to be removed in order to reduce unnecasery heap allocation. This is because strings are reference types not value types because it is an objects of a class.

VII. MEMORY POOLING

Memory pooling is the process of allocating memory ahead of time so that when memory is needed any unused space from tha allocated memory can be used. When the memory is no longer needed the memory space is marked as unused so it is free to be used by another device. Using memory pooling means you can allocate large amounts of memory to the heap which means when you need the memory you are just referencing the one instance unlike the stack where the memory is duplicated.By allocating large amounts of memory to the heap it leaves the stack free for computations which can make that program run faster.With the stack it keeps track of what is currently going on within the code so it uses the function at the top of the stack. If large amount of memory is on the top of the stack it can slow it down massively.Within my game this could mean loading the general map on the heap and elements that I am interacting with on the stack as they are involved in computations.

VIII. CONCLUSION

There are many more techniques then those focused on in this paper but these tackle important areas of optimisation. Implementing these techniques among others are very important for developers of any scale from indie to AAA. Within Unity knowledge of the profile itself as a tool is an expectation of industry companies and knowing these techniques combines with this knowledge making the optimisation process. This can make improve the players experience during the game and allow for more features to be in the game.