Optimisation research Journal

1505536

I. Optimisation

Optimisation within games is the process making a software more effecient 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 unresponse. 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.

II. 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 expcially in big projects it can be a waste of time and resources to manually look for improvments.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 methododolgy in profiling is to run the profiler, analiyse 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 perfomrnace 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 optiised 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 perfromance as users will experience it.

III. 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 thats size is unknown 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 ertain amount fo enemies. The amount of enemeis 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.

IV. Frequency of function calls

Calling functions takes up resources and can slwo down a game causing frame rate issues.When calling functions one way to help optimise is look at how oftena 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 unneccasary functions. One example is calling a line tracer which takes up lots of memory because it is doing calculation every frame of the gmae. 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 neccassary, and as a result was wasting resources. Putting delays on how often functions are called can be done ij Unity by usuing the invoke keyword and selecting whatever time ou want.

V. Assets

In a scene the amount of assets present and complexity of assets can cause perfromance issues.When looking at the profiler duruing run time comparing it tot he amount of objects ina scene can help to identify a problem. Within the COMP350 project enemies were spawning constantly in set locations. Having enemies constantly being created meant that there were constant spikes in the physics and it constantly became strenious on the computers resources. One way to fix this is to spawn all of the enemies at the start of the level and have them invisible with no physics active until they are needed. Having a limit on the amount of enemies that spawned also prevents to many from beinf present. Within this perticular project constantly spawning enemies was unnessary anyway to all enemies could just be added to the level manually.

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 howeve 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 poooling is the process of alocating memory ahead of time so that when memory is needed any unsused space from tha alocated memory can be used. When the memory is no longer needed the memory space is marked as unsused 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 computaions which can make th progra 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 im interacting with on the stack as they are involved in computations.