COMP350-Research Journal

Optimisation of Games

1506530

March 21, 2018

1 Introduction

An optimisation is making the most out of the available resources and is essential to creating a smooth experience for the user. In the games industry, there are many types of optimisation methods and tools, one such tool is the unreal profiling tool [1], which includes a detailed breakdown of everything that runs in the game loop. Amongst the unreal tools are also a Graphics Processing Unit (GPU) profiler, and an asset statistics sheet, there are more ways to get an overview of what is happening in the engine which shall be covered in this journal. The main point of this journal is to provide a foundation for optimising a game within the unreal engine providing both about the tool and how to use it.

2 Optimisation in Unreal

Getting the most Frames per Second (FPS) out of your system is the most important thing to achieve when optimising a game [2], and in Unreal this can be done in several ways, a list of which can be found at the Unreal Documentation. In this journal, an overview of some of the tool will be presented with examples of how to use them.

The first step is to find the bottleneck and see whether it is in the Central Processing Unit (CPU) or GPU or in the Game (Your Code). A bottleneck means at what point in the scene draw is the game slowed, i.e. the GPU could be slower than your CPU at drawing the scene, therefore your bottleneck would be in the GPU. Unreal provides a command "stat.unit" which displays statistics similar to image 1. In this example the Frame and GPU are very close in value, this means that the bottleneck is the GPU. If it were the CPU then the Draw would be close, and if it is your code then the Game will be. The best way to discover which this is is by launching a release build of the game and typing the command, then noting the values. Once you have discovered the bottleneck then the next move is to use the appropriate profiler to fix the issue.

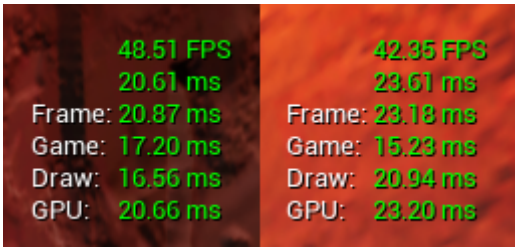

Figure 1: A before and after picture showing the stat.unit command in Unreal

3 CPU Profiling

Being CPU bound means that there are too many draw calls, this is an issue that artists can help overcome by combining multiple assets into one larger asset. Having multiple instances of the same material can increase the draw time and can be resolved by using a material instance. The CPU render thread also handles processing objects, GPU commands and graphics drivers [3].

3.1 Code Optimisation

Another way you can be CPU bound is in the game thread in this instance using the session frontend profiler will give you an idea of where the CPU is begin slowed. To begin finding this a use of the command "stat startfile" when running the game and then "stat stopfile" when the game has been run for a period of time. This action will record and save data from the game which can then be opened in the session frontend window and inspected. This is related to the game's code, which in itself can be a maze to find the issue, one common pitfall is the "FTickFunctionTask" which is the tick for every actor and component with one. Reducing the number of actors that tick or introducing a delay to tick triggers can help reduce this issue [4].

4 GPU Profiling

The GPU profiler shows the user how much each rendered object uses during runtime, this can allow the user to quickly identify any cost for the various passes. There are two types of GPU profilers in Unreal, one text version and one with a mouse-based UI. Both have their merits and both have limitations
Mouse Based UI:

- You can see a greater level of information for the high-level costs, for example, you can see exactly what is causing the PrePass to be of a high cost by clicking on it.
- Unfortunately this does not give a real-time running cost view of the game, the information is given from data taken from the frame it is called on.

- Some drivers can optimise shaders before the GPU profiler uses the data so it can produce false numbers.

Text:

- Unlike the mouse-based UI, you can not see much information passed the high-level names of the pass.
- The text can give a real-time running cost of the game allowing for greater accuracy and finding trouble areas.
- The text can be used during the release build.

To access the GPU text profiler just type the command "stat.gpu" and a wall of text will appear in-game as seen in image 2. Using the same "stat start/stopfile" command you can load the in-game GPU information into the frontend tool for reviewing [5].

Figure 2: A before and after picture showing the stat.gpu command in Unreal

5 Content Optimisation

Simply put content optimisation is the process of optimising the objects that are rendered in-game, including the meshes, animations and lighting. Though lighting can be dealt with using the GPU profiler there are other ways to identify the issue, though in this instance the meshes are the focus. Unreal provides a Mesh statistics profiler that is a spreadsheet of data that contains various pieces of information that can help to identify any suspect meshes [6]. As can be seen in image 4, there are several meshes referencing 4K texture files when they do not need to, this can be easily rectified by selecting the asset and editing the max resolution.

6 Final Remarks

There are many tools in Unreal that can help developers identify and rectify issues within their game to ensure it runs at maximum efficiency. Though there are many more than those described in this journal, the ones mentioned here cover the basics. When planning on optimising work it is always best to plan ahead and try to optimise during development and not to optimise at the end, as this can reveal larger issues within your program that may require a larger amount of time and effort to fix than you have available [7].

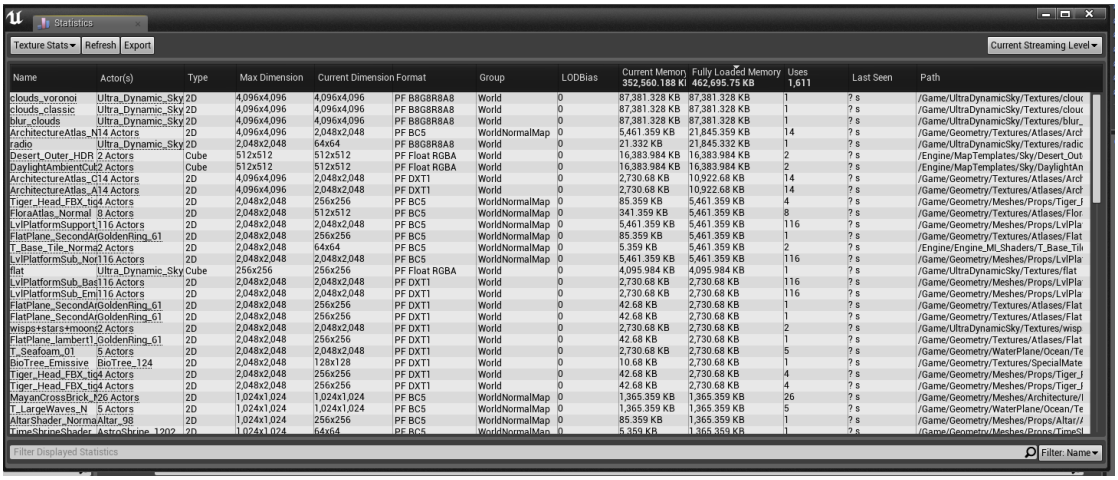

Figure 3: An Image showing the statistics page for assets in Unreal

References

[1] E. M. and J. Rous. Performance and profiling. [Online]. Available: https://docs.unrealengine.com/en-us/Engine/Performance[Accessed18-03-2018]

[2] E. Games. Optimizing your game — live training — unreal engine livestream. [Online]. Available: https://www.youtube.com/watch?v=U0p8EY07_mc[Accessed18-03-2018]

[3] ——. Cpu profilingl. [Online]. Available: https://docs.unrealengine.com/en-us/Engine/Performance/CPU[Accessed18-03-2018]

[4] B. Tellez. Uhow to improve game thread cpu performance in unreal engine. [Online]. Available: https://www.unrealengine.com/en-US/blog/how-to-improve-game-thread-cpu-performance[Accessed18-03-2018]

[5] E. Games. Gpu profiling. [Online]. Available: https://docs.unrealengine.com/en-us/Engine/Performance/GPU[Accessed18-03-2018]

[6] M. Thorzen. The witcher 3: Optimizing content pipelines for open-world games. [Online]. Available: https://www.youtube.com/watch?v=p8CMYD_5gE8[Accessed18-03-2018]

[7] Z. Parrish. Ue4 performance and profiling — unreal dev day montreal 2017 — unreal engine. [Online]. Available: https://www.youtube.com/watch?v=hcxetY8g_fs[Accessed18-03-2018]