COMP350-Research Journal

Optimisation of Games

1506530

March 18, 2018

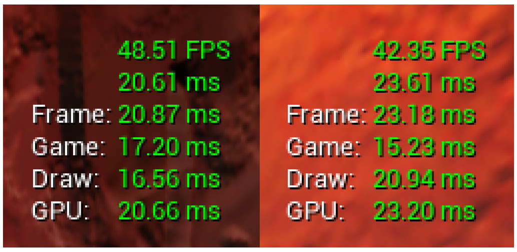
1 Introduction

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 a 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 optimizing a game within the unreal engine providing both about the tool and how to use it.

2 Optimisation in Unreal

Optimisation making the most out of the available resources. In games this is usually attributed to gaining the most Frames per Second (FPS) out of your system [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 on how to use them.

The first step is to find the bottleneck and see whether its in the CPU or GPU or in the Game (Your Code). 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 its 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.



```
      48.51 FPS         42.35 FPS
      20.61 ms          23.61 ms
Frame: 20.87 ms   Frame: 23.18 ms
Game:  17.20 ms   Game:  15.23 ms
Draw:  16.56 ms   Draw:  20.94 ms
GPU:   20.66 ms   GPU:   23.20 ms
```

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 te 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 games 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 thw 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 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 clikcing 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.

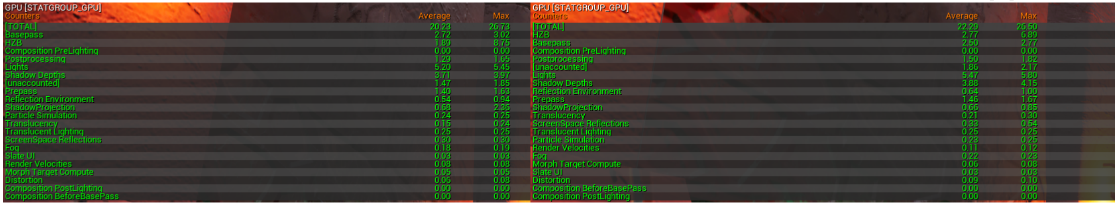
- It can not be used during the release version of the game.

- 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 ingame as seen in image 2. Using the same “stat start/stopfile” command you can load the ingame GPU information into the frontend tool for reviewing [5].

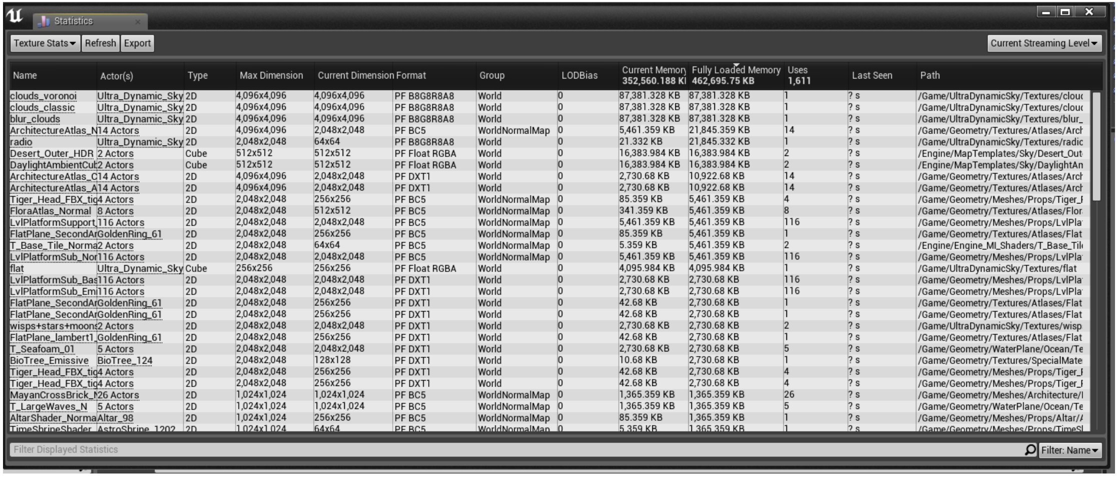


```
GPU [STATGROUP_GPU]
Counters                        Average    Max
[TOTAL]                           20.23  26.73
Basepass                           2.72   3.02
HZB                                1.89   8.75
Composition PreLighting            0.00   0.00
Postprocessing                     1.29   1.65
Lights                             5.20   5.45
Shadow Depths                      3.71   3.97
[unaccounted]                      1.47   1.85
Prepass                            1.40   1.63
Reflection Environment             0.54   0.94
ShadowProjection                   0.68   2.36
Particle Simulation                0.24   0.25
Translucency                       0.15   0.24
Translucent Lighting               0.25   0.25
ScreenSpace Reflections            0.30   0.30
Fog                                0.18   0.19
Slate UI                           0.03   0.03
Render Velocities                  0.08   0.08
Morph Target Compute               0.05   0.05
Distortion                         0.06   0.08
Composition PostLighting           0.00   0.00
Composition BeforeBasePass         0.00   0.00
```

```
GPU [STATGROUP_GPU]
Counters                        Average    Max
[TOTAL]                           22.29  26.50
HZB                                2.77   6.89
Basepass                           2.50   2.77
Composition PreLighting            0.00   0.00
Postprocessing                     1.50   1.82
[unaccounted]                      1.86   2.17
Lights                             5.47   5.80
Shadow Depths                      3.88   4.15
Reflection Environment             0.64   1.00
Prepass                            1.46   1.67
ShadowProjection                   0.66   0.85
Translucency                       0.21   0.30
ScreenSpace Reflections            0.33   0.54
Translucent Lighting               0.25   0.25
Particle Simulation                0.23   0.25
Render Velocities                  0.11   0.12
Fog                                0.22   0.23
Morph Target Compute               0.06   0.08
Slate UI                           0.03   0.03
Distortion                         0.09   0.10
Composition BeforeBasePass         0.00   0.00
Composition PostLighting           0.00   0.00
```

5 Content Optimisation

Simply put content optimisation is the process of optimising the objects that are rendered ingame, 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 a 4K texture files when they do not need to, this can be easily rectified by selecting the asset and editing the max resolution.



Name	Actor(s)	Type	Max Dimension	Current Dimension	Format	Group	LODBias	Current Memory 352,560.188 KI	Fully Loaded Memory 462,695.75 KB	Uses 1,611	Last Seen	Path
clouds_voronoi	Ultra_Dynamic_Sky	2D	4,096x4,096	4,096x4,096	PF B8G8R8A8	World	0	87,381.328 KB	87,381.328 KB	1	? s	/Game/UltraDynamicSky/Textures/clouc
clouds_classic	Ultra_Dynamic_Sky	2D	4,096x4,096	4,096x4,096	PF B8G8R8A8	World	0	87,381.328 KB	87,381.328 KB	1	? s	/Game/UltraDynamicSky/Textures/clouc
blur_clouds	Ultra_Dynamic_Sky	2D	4,096x4,096	4,096x4,096	PF B8G8R8A8	World	0	87,381.328 KB	87,381.328 KB	1	? s	/Game/UltraDynamicSky/Textures/blur_
ArchitectureAtlas_N	14 Actors	2D	4,096x4,096	2,048x2,048	PF BC5	WorldNormalMap	0	5,461.359 KB	21,845.359 KB	14	? s	/Game/Geometry/Textures/Atlases/Arch
radio	Ultra_Dynamic_Sky	2D	2,048x2,048	64x64	PF B8G8R8A8	World	0	21.332 KB	21,845.332 KB	1	? s	/Game/UltraDynamicSky/Textures/radic
Desert_Outer_HDR	2 Actors	Cube	512x512	512x512	PF Float RGBA	World	0	16,383.984 KB	16,383.984 KB	2	? s	/Engine/MapTemplates/Sky/Desert_Out
DaylightAmbientCub	2 Actors	Cube	512x512	512x512	PF Float RGBA	World	0	16,383.984 KB	16,383.984 KB	2	? s	/Engine/MapTemplates/Sky/DaylightAn
ArchitectureAtlas_C	14 Actors	2D	4,096x4,096	2,048x2,048	PF DXT1	World	0	2,730.68 KB	10,922.68 KB	14	? s	/Game/Geometry/Textures/Atlases/Arch
ArchitectureAtlas_A	14 Actors	2D	4,096x4,096	2,048x2,048	PF DXT1	World	0	2,730.68 KB	10,922.68 KB	14	? s	/Game/Geometry/Textures/Atlases/Arch
Tiger_Head_FBX_tig	4 Actors	2D	2,048x2,048	256x256	PF BC5	WorldNormalMap	0	85.359 KB	5,461.359 KB	4	? s	/Game/Geometry/Meshes/Props/Tiger_I
FloraAtlas_Normal	8 Actors	2D	2,048x2,048	512x512	PF BC5	WorldNormalMap	0	341.359 KB	5,461.359 KB	8	? s	/Game/Geometry/Textures/Atlases/Flor
LvlPlatformSupport_	116 Actors	2D	2,048x2,048	2,048x2,048	PF BC5	WorldNormalMap	0	5,461.359 KB	5,461.359 KB	116	? s	/Game/Geometry/Meshes/Props/LvlPla
FlatPlane_SecondAr	GoldenRing_61	2D	2,048x2,048	256x256	PF BC5	WorldNormalMap	0	85.359 KB	5,461.359 KB	1	? s	/Game/Geometry/Textures/Atlases/Flat
T_Base_Tile_Norma	2 Actors	2D	2,048x2,048	64x64	PF BC5	WorldNormalMap	0	5.359 KB	5,461.359 KB	2	? s	/Engine/Engine_MI_Shaders/T_Base_Til
LvlPlatformSub_Nor	116 Actors	2D	2,048x2,048	2,048x2,048	PF BC5	WorldNormalMap	0	5,461.359 KB	5,461.359 KB	116	? s	/Game/Geometry/Meshes/Props/LvlPla
flat	Ultra_Dynamic_Sky	Cube	256x256	256x256	PF Float RGBA	World	0	4,095.984 KB	4,095.984 KB	1	? s	/Game/UltraDynamicSky/Textures/flat
LvlPlatformSub_Bas	116 Actors	2D	2,048x2,048	2,048x2,048	PF DXT1	World	0	2,730.68 KB	2,730.68 KB	116	? s	/Game/Geometry/Meshes/Props/LvlPla
LvlPlatformSub_Emi	116 Actors	2D	2,048x2,048	2,048x2,048	PF DXT1	World	0	2,730.68 KB	2,730.68 KB	116	? s	/Game/Geometry/Meshes/Props/LvlPla
FlatPlane_SecondAr	GoldenRing_61	2D	2,048x2,048	256x256	PF DXT1	World	0	42.68 KB	2,730.68 KB	1	? s	/Game/Geometry/Textures/Atlases/Flat
FlatPlane_SecondAr	GoldenRing_61	2D	2,048x2,048	256x256	PF DXT1	World	0	42.68 KB	2,730.68 KB	1	? s	/Game/Geometry/Textures/Atlases/Flat
wisps+stars+moons	2 Actors	2D	2,048x2,048	2,048x2,048	PF DXT1	World	0	2,730.68 KB	2,730.68 KB	2	? s	/Game/UltraDynamicSky/Textures/wisp
FlatPlane_lambert1_	GoldenRing_61	2D	2,048x2,048	256x256	PF DXT1	World	0	42.68 KB	2,730.68 KB	1	? s	/Game/Geometry/Textures/Atlases/Flat
T_Seafoam_01	5 Actors	2D	2,048x2,048	2,048x2,048	PF DXT1	World	0	2,730.68 KB	2,730.68 KB	5	? s	/Game/Geometry/WaterPlane/Ocean/Te
BioTree_Emissive	BioTree_124	2D	2,048x2,048	128x128	PF DXT1	World	0	10.68 KB	2,730.68 KB	1	? s	/Game/Geometry/Textures/SpecialMate
Tiger_Head_FBX_tig	4 Actors	2D	2,048x2,048	256x256	PF DXT1	World	0	42.68 KB	2,730.68 KB	4	? s	/Game/Geometry/Meshes/Props/Tiger_I
Tiger_Head_FBX_tig	4 Actors	2D	2,048x2,048	256x256	PF DXT1	World	0	42.68 KB	2,730.68 KB	4	? s	/Game/Geometry/Meshes/Props/Tiger_I
MayanCrossBrick_N	26 Actors	2D	1,024x1,024	1,024x1,024	PF BC5	WorldNormalMap	0	1,365.359 KB	1,365.359 KB	26	? s	/Game/Geometry/Meshes/Architecture/I
T_LargeWaves_N	5 Actors	2D	1,024x1,024	1,024x1,024	PF BC5	WorldNormalMap	0	1,365.359 KB	1,365.359 KB	5	? s	/Game/Geometry/WaterPlane/Ocean/Te
AltarShader_Norma	Altar_98	2D	1,024x1,024	256x256	PF BC5	WorldNormalMap	0	85.359 KB	1,365.359 KB	1	? s	/Game/Geometry/Meshes/Props/Altar/A
TimeShrineShader	AstroShrine_1202	2D	1,024x1,024	64x64	PF BC5	WorldNormalMap	0	5.359 KB	1,365.359 KB	1	? s	/Game/Geometry/Meshes/Props/TimeSl

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 optimizing 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 is available [7].

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]