Spike: 20
Title: Measuring Performance

Author: Ford Killeen, 9731822

Goals / deliverables:
Goals this spike aims to achieve:

- Add 100 lights (dynamic/static) to an existing scene
- Measure performance difference between dynamic and static lights
- Write a short report on how to measure performance in UE4

Deliverables required:

- Short report on measuring performance in UE4
- Proof of performance measuring and the difference between the 100 dynamic lights and 100 static lights
- Spike report

Technologies, Tools, and Resources used:
The following is required to complete this spike:

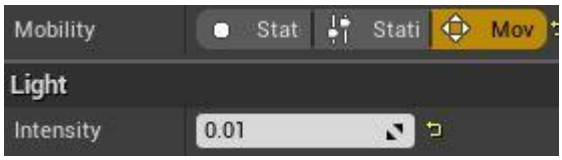
- Unreal Engine 4 (ver. 4.13.2)
- Unreal Engine Performance Profiler
- Pre-existing UE4 scene to perform tests in

Tasks undertaken:
The list below details the steps taken to complete this spike.

- The first thing I did was write a short report detailing how someone can record and measure the performance of their project in the Unreal Engine. This is attached at the end of this spike report.
- Next I opened my existing project from spike 20.
- I then selected the *Light Source* directional light in the scene, adjusted its *Mobility* setting to *Static*, making it a static, baked light with no dynamic properties. I also changed the *Intensity* to 0.01 as with 100 standard lights it was far too bright.

- I then duplicated the light (*Ctrl + W*) so that there were 100 instances of that same static light within the scene. The project was then rebuilt.

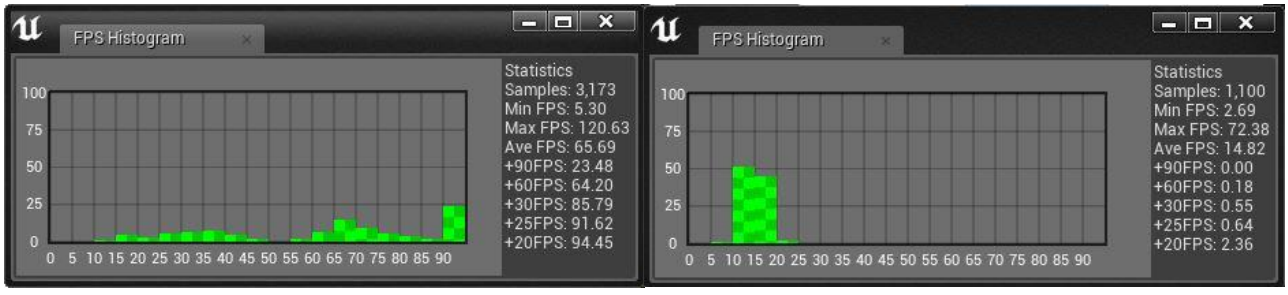
- The performance profiler was opened (see short report below for detailed steps) and the play session was recorded.
- Back in the editor I selected all 100 lights and changed them to be *Moveable*, which makes them completely dynamic lights. The project was then rebuilt.



- A new performance profiler session was opened and recorded using the new dynamic setup.
- The results of both sessions were then compared and are discussed further in the *What we found out* section below.

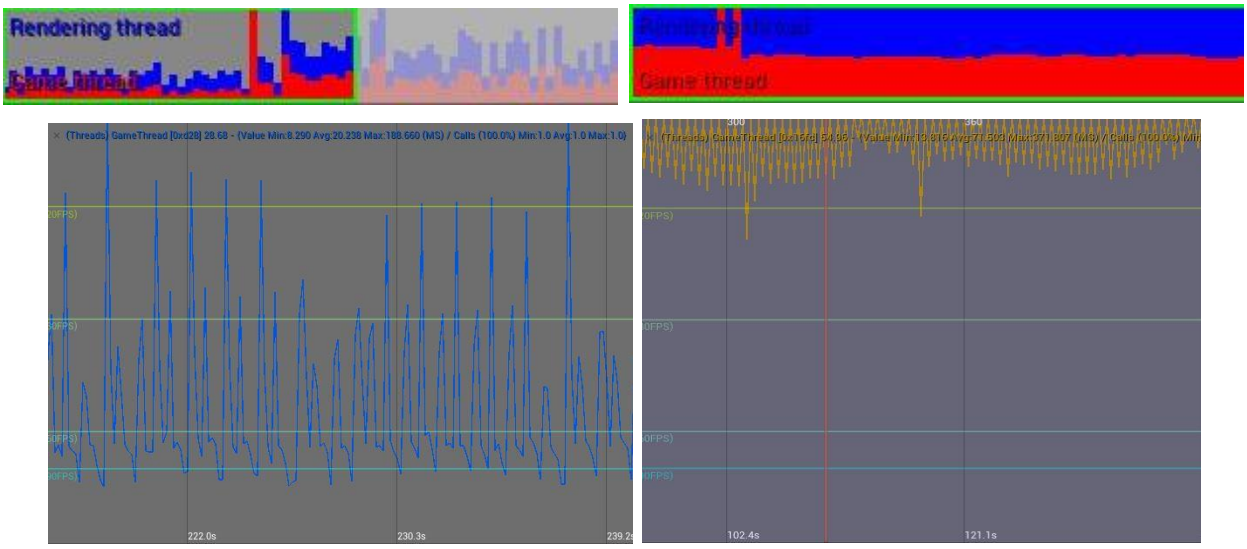
What we found out:
By completing this spike we found out how to monitor and record the performance of an Unreal Engine 4 project. This is important for games development as performance and reliability is a big part of games. That is, if a game requires a supercomputer to run or doesn't run smoothly in certain scenes, then players will get frustrated and possibly stop playing altogether. It is also important as developers will be given certain resource limits to stick by so that the overall game can function at its best.

Results:
The test consisted of me playing my scene with both 100 static/dynamic lights, running around and shooting the cubes in the level. The results can be seen below and the main cause of the drastic performance change is that the static/baked lights are computed once and then never again during play, whereas the dynamic lights are constantly computed and calculated throughout play causing a much larger number of calculations per tick, especially with 100 different dynamic lights.



Static FPS vs. Dynamic FPS

As we can see in the above two pictures, the left picture shows a much higher average frame rate with the largest bulk of frames being at 90+ frames per second. The right picture with the dynamic lights however, shows the average frame rate drop significantly and shows the majority of frames between 10 and 20 frames per second.



Static Graphs vs. Dynamic Graphs

Here we can see the graphs from the performance profiler, with the left images being the static lighting and the right images being the dynamic lighting. With the bottom two images lined up accurately using the frames per second (FPS) lines, we can clearly see that the response times on the left are much quicker than those on the right, and that the FPS numbers are much lower (closer to the top) on the right.

Unreal Engine Performance Profiler – How To:

The below steps will detail how to open and use the UE4 performance profiler for basic performance monitoring.

- With your UE4 project open, go to *Window -> Developer Tools -> Session Frontend* from the toolbar to open the profiler.
- With the *Session Frontend* window open, ensure your device is selected in the top left side of the window so that you are receiving data from your device/editor session.
- Looking to the top centre of the window, navigate to the *Profiler* tab.
- With the *Profiler* tab selected, you can now click the *Data Preview* and *Live Preview* buttons to kick off the graph view for some basic information.
- To add more values to the graph, in the left side of the *Profiler* tab you will see a list of values that can be added. These are categorised and you will need to expand the group to see the actual values available. Double click on a value for it to be visible in the graph view.
- To save the recorded data, click the *Data Capture* button along the top row within the *Profiler* tab and keep this on until you want to stop. When ready to stop unselect it and it will ask if you want to save the session to your local machine. Click yes so that you can come back and analyse this data at any time in the future.