Drowned Path

Optimization document

Maxime GRIMARD—-HAMEL, Gebril BENATTIA, Paul BUSSY

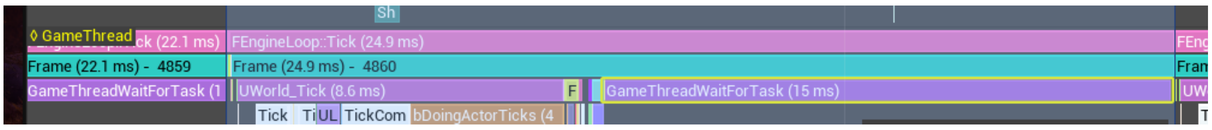
Summary

Introduction

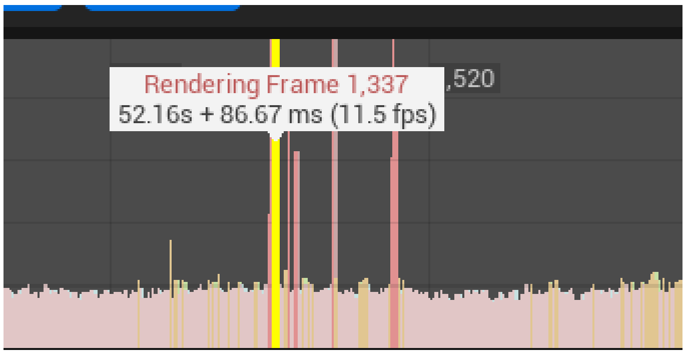
This is a document about what we optimize on this project and also some things we didn't have time to but if we had more time we would have checked.
Firstly, we had a well optimized game since the beginning of the project and for the majority of the development so it was a little tricky to know where we needed to look for performance issues. Though we still checked some things during the project to make sure we kept this good performance along the rest of the project.

Profiling

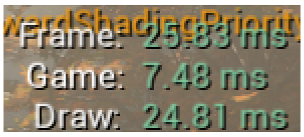
At first we did some profiling along the project, and we saw that one of the longest tasks was the "WaitForTask" function on the CPU, and after further research we learned that this task waits for some action in parallel to be executed. In our case it was the GPU mainly.



We had sometimes some FPS drop related to the rendering frame like this one :



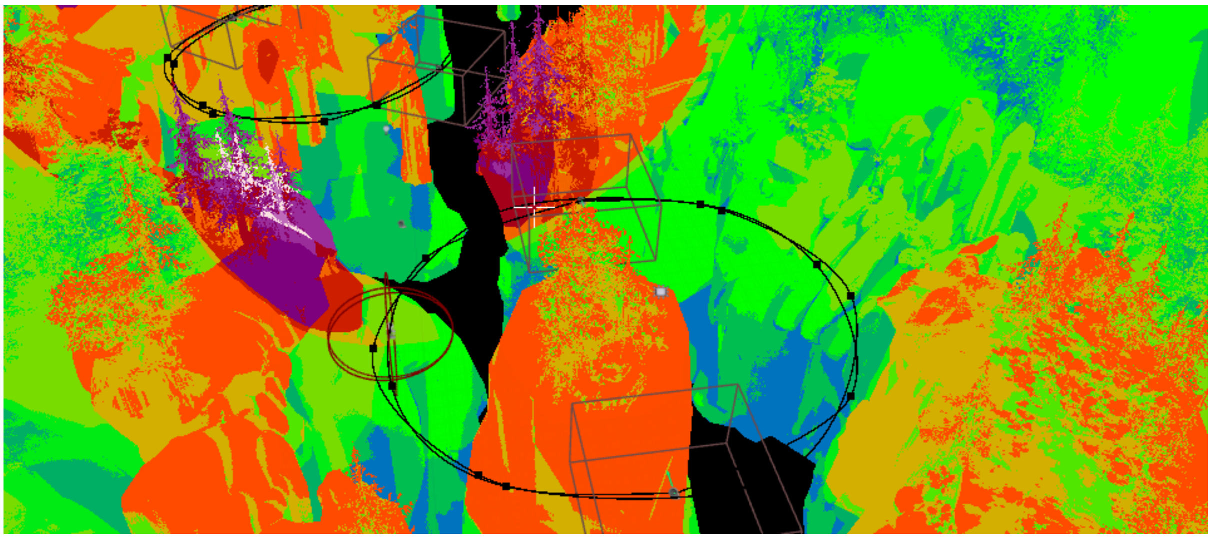
After realising that we decided to launch the game to look closer to the stats in Standalone and when we looked we confirmed that the Draw frame was the longest to process. There were a high difference between the Game and Draw so with the time we had we focused mainly on the Draw and did not really dig to optimize the Game:

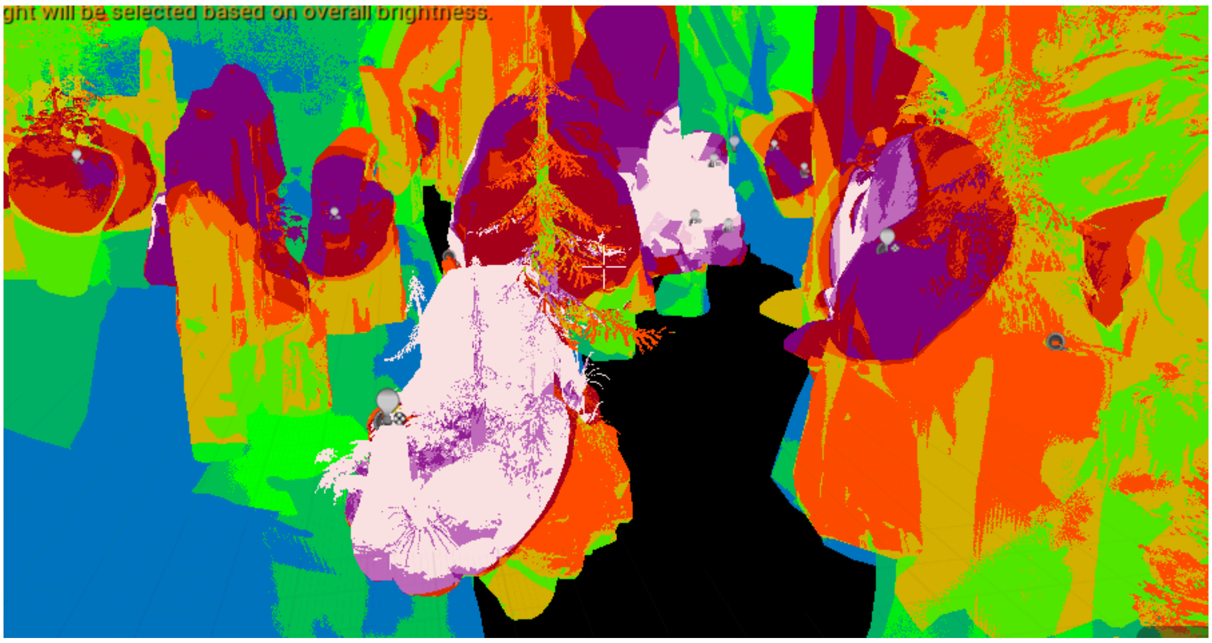


```
Frame: 25.83 ms
 Game: 7.48 ms
 Draw: 24.81 ms
```

Lights

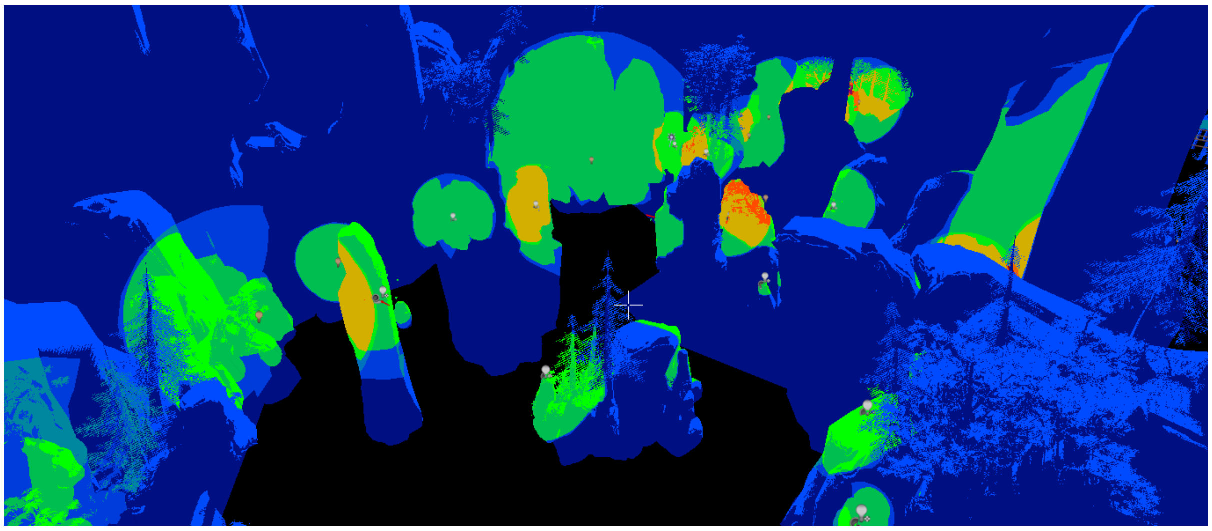
Early in the project we used the optimization viewmodes in Unreal to ensure that the lights were correctly set and if there were not too much complexity, which could have been one of the causes of our lack of FPS. Indeed when we looked specifically at the scene with our boat, we saw that the light complexity was very high even reaching the white color, which is the highest level of complexity.


Light complexity on LD BOAT 2



Light complexity on LD BOAT 1

So, with the help of our artists, we deleted the lights that were too close to each other and just kept the props themselves and set the light with the right properties (movable or stationary). Which was really efficient and really simplified the complexity.

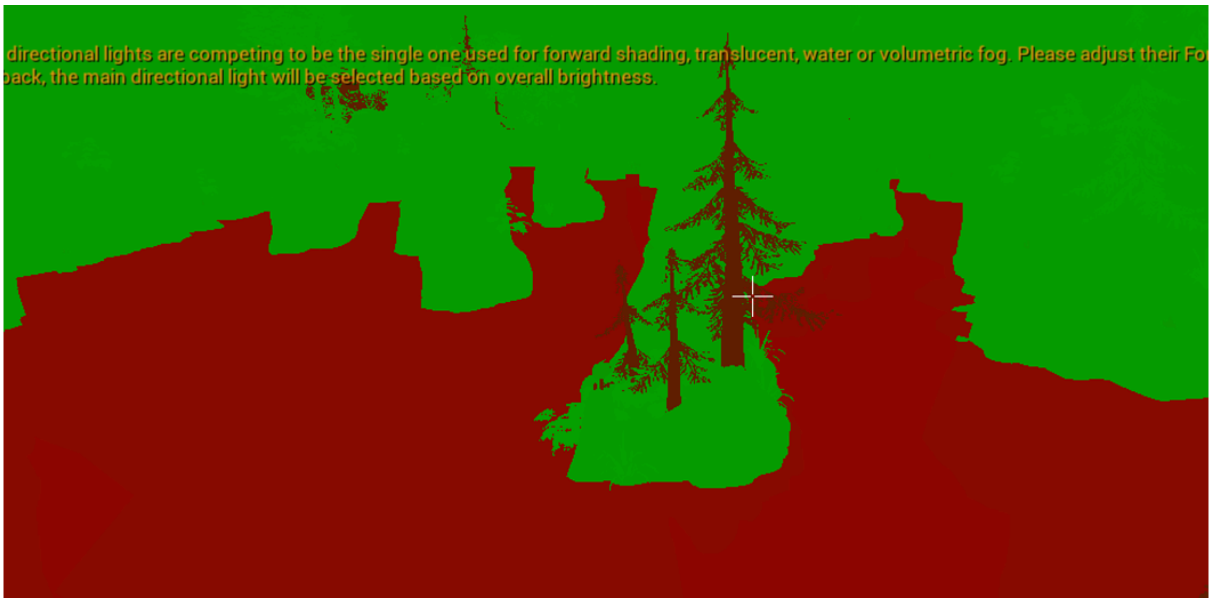


New light complexity on LD BOT 1

Shaders

Water

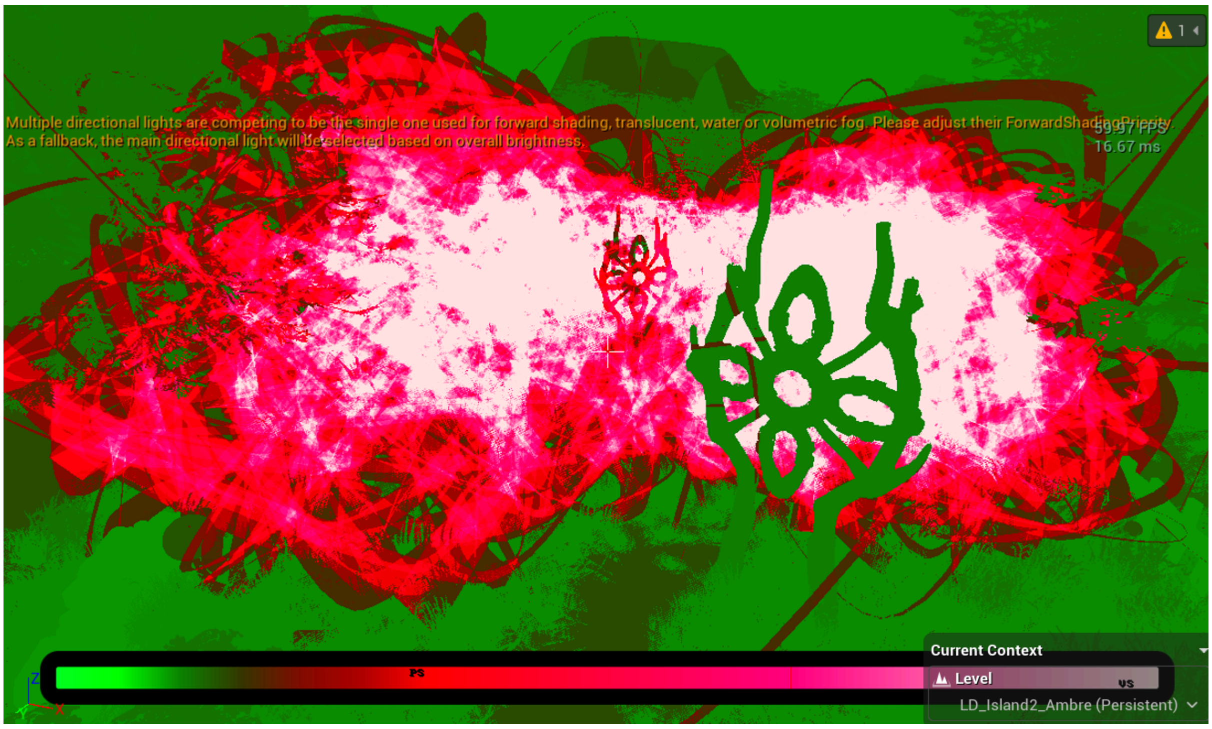
Another thing we saw on the boat LD was that the shader complexity for the water was red, but after talking with the artists we thought that it was not the main thing we had to optimize so we moved on to other things quickly and a teacher told us later that they are things with shader that at one point, will take resources no matter what so it was another reason to not took our time on this one.


directional lights are competing to be the single one used for forward shading, translucent, water or volumetric fog. Please adjust their Fo
back, the main directional light will be selected based on overall brightness.

Shader complexity LD BOAT 1 : red is water

Veil

But we had another shader and this one was really heavy. It was the shader of the veil (our swarm enemy) and the shader complexity for this one was extremely high.


Multiple directional lights are competing to be the single one used for forward shading, translucent, water or volumetric fog. Please adjust their ForwardShadingPriority.
As a fallback, the main directional light will be selected based on overall brightness.
59.97 FPS
16.67 ms

Shader complexity on LD_Island2

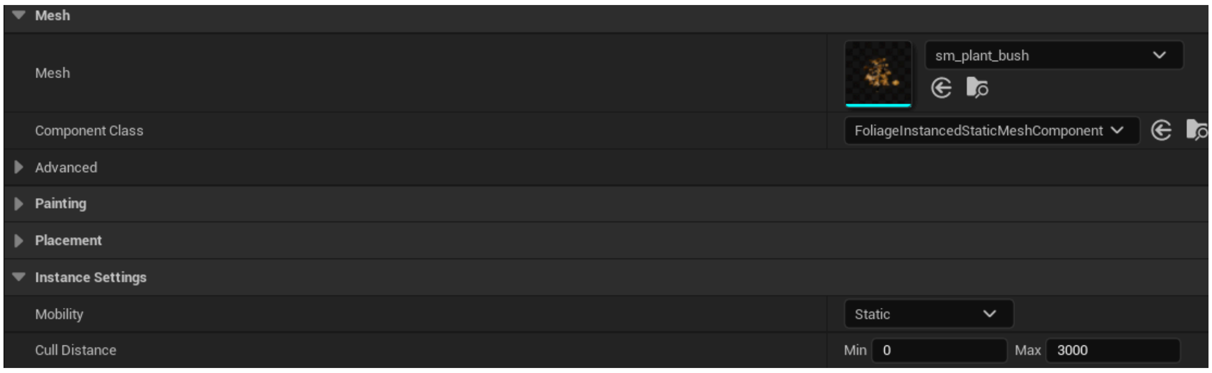
So we did some research and saw some ways of optimizations, but they all needed us to remake the Niagara System. Currently, the Niagara is made with a ribbon between two points, which causes a lot of overdraws in the center. One thing is that we could have done this with a fog particle system instead or by using a fluid simulation with a plugin like Fluid Ninja, but those were really time-consuming and needed the full help of our VFX artist that was not available at the time. One other minor fix we could have done was to reduce the number of ribbons and create them further from the center, but for the reduction, our artist said that reducing the number would make the VFX visually bad.

Level Streaming

Another thing we wanted was to use level streaming for all our levels. But after realizing that they aren't linked in any way (they act like separate rooms), and that we could just load the next level synchronously as they aren't that big, we decided not to use it. Using level streaming would probably have made the playing experience much better, but it would've needed the levels to be adapted accordingly.

Culling Distance

After making some test we realize that the FPS were dropping when we looked at the village, so we tried adding distance culling on the static mesh foliage and on some props on the second island, which was the one that caused the most lag, and we've seen a huge frame rate improvement, almost doubling what we had before and getting more than 100 fps in some situations on our computers.


Mesh
Mesh sm_plant_bush
Component Class FoliageInstancedStaticMeshComponent
Advanced
Painting
Placement
Instance Settings
Mobility Static
Cull Distance Min 0 Max 3000

Additional optimizations not done in the project

Due to a lack of time in the project and also the fact that we already had a game running smoothly with those previous optimizations, we did not explore other things we could have done.

Indeed, one of the things we did not do is to check if all the LOD of the items are correctly set and with the right value for the switch between them. It would permit for some moments in the game where we look at trees or houses from far to draw less vertices to lighten the GPU.

Also we did not check our GDP's BP because our main problem was the GPU usage, but there certainly are some things to optimize like the water and boat system where there is maths and this is something we should compute on a C++ file since it is much more efficient.