Some Optimization Techniques Using Unreal Engine

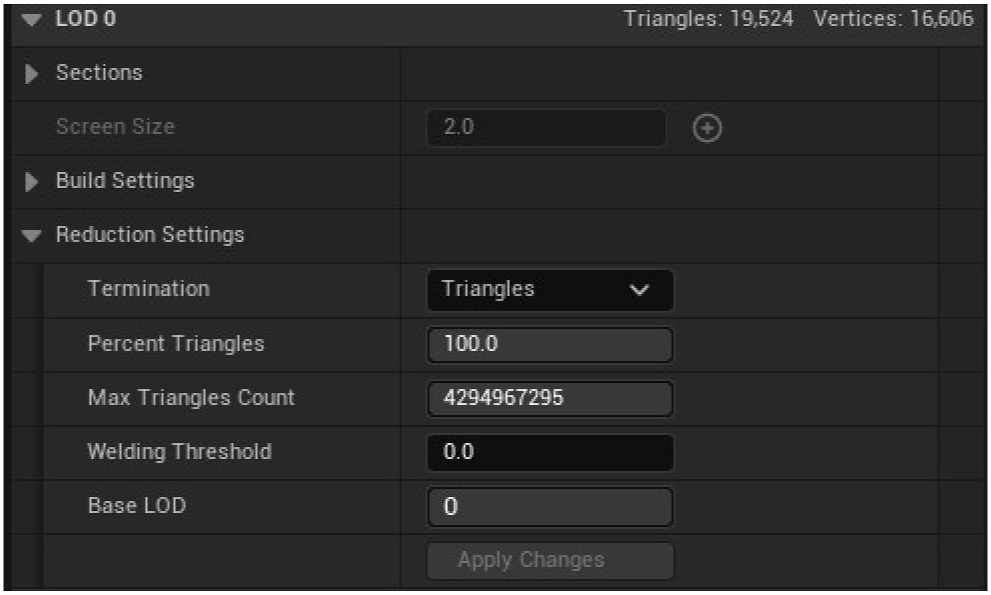
Summary

LODs

https://dev.epicgames.com/documentation/en-us/unreal-engine/creating-and-using-lods-in-unreal-engine

Using a default tyre that comes with Unreal. "SM_Offroad_Tire".

In the details panel, the maximum number of polygons for a mesh asset can be adjusted.
The total number of polygons on the asset can also be reduced by a certain percentage.



LOD 0 — Triangles: 19,524 Vertices: 16,606

Sections

Screen Size: 2.0

Build Settings

Reduction Settings

Termination: Triangles

Percent Triangles: 100.0

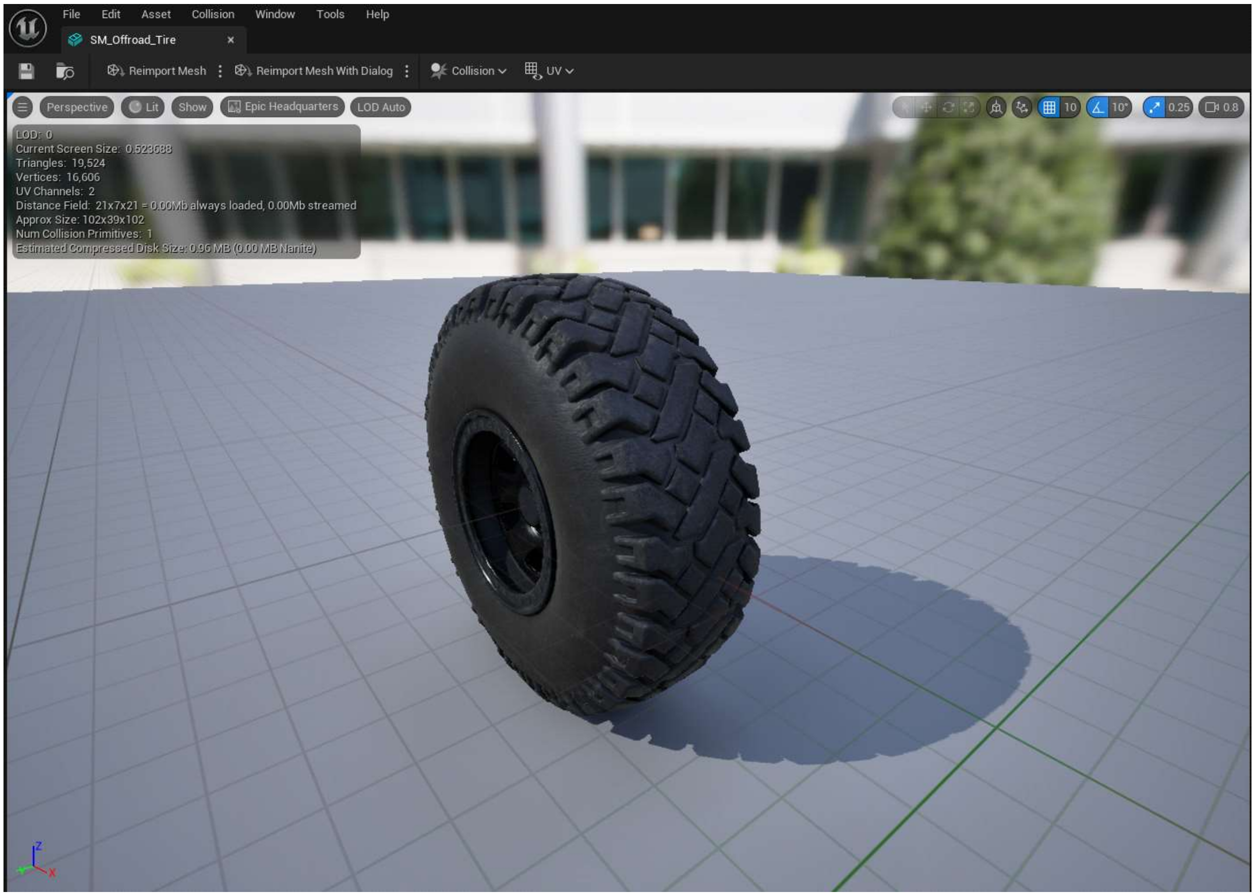
Max Triangles Count: 4294967295

Welding Threshold: 0.0

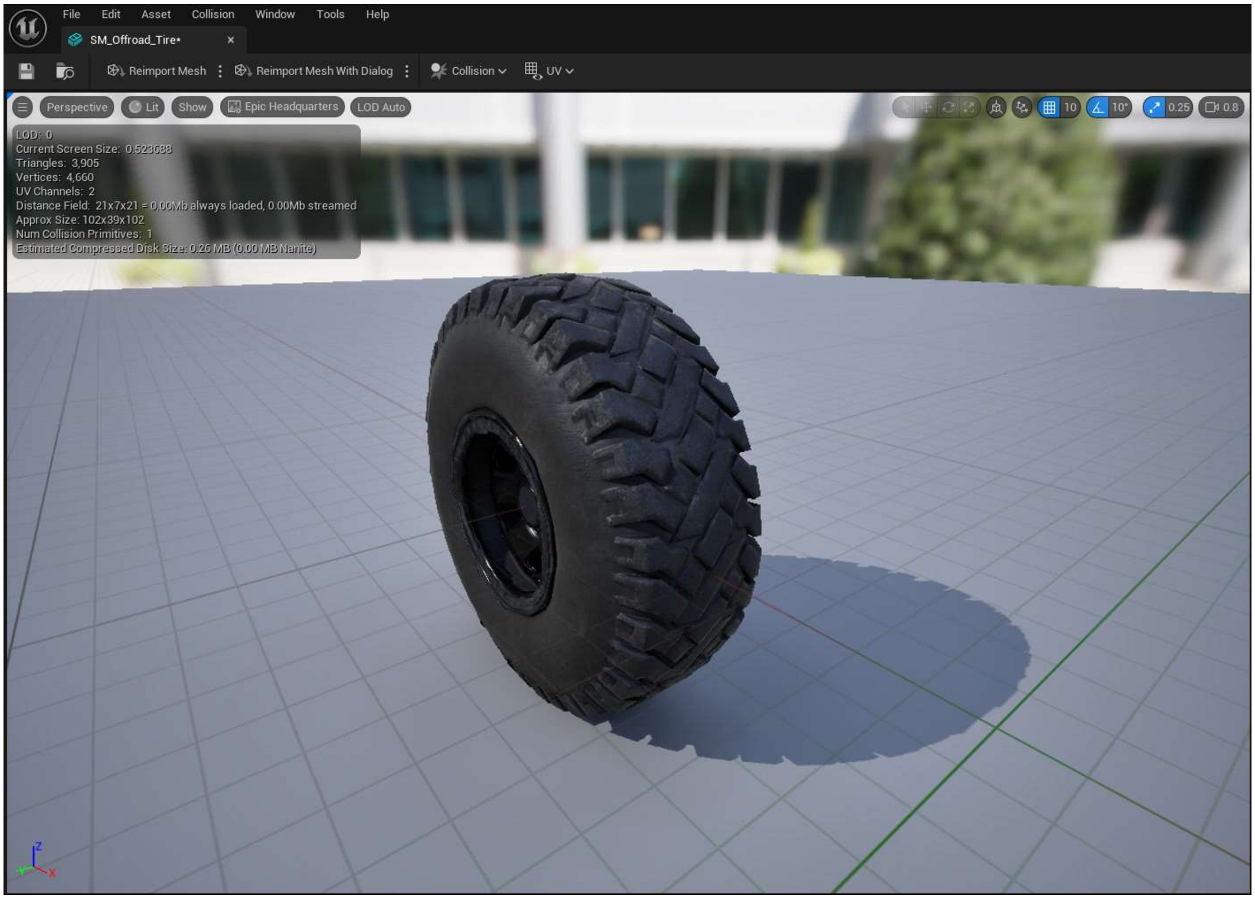
Base LOD: 0

This will not deteriorate the initial mesh copy, which is a worry I had.
Unreal creates (what they call) an 'imported' copy of the mesh. Every mesh has this copy by default (assigned the default name "LOD 0").
A mesh asset can have more than one copy. Unreal will create multiple copies and whichever one is showed on screen will depend on how much space that asset takes up on the screen. (So assets which are more visible will have higher detail etc).

I'm going to continue testing with one LOD for now.



```
LOD: 0
Current Screen Size: 0.523688
Triangles: 19,524
Vertices: 16,606
UV Channels: 2
Distance Field: 21x7x21 = 0.00Mb always loaded, 0.00Mb streamed
Approx Size: 102x39x102
Num Collision Primitives: 1
Estimated Compressed Disk Size: 0.96 MB (0.00 MB Nanite)
```



LOD: 0
Current Screen Size: 0.523688
Triangles: 3,905
Vertices: 4,660
UV Channels: 2
Distance Field: 21x7x21 = 0.00Mb always loaded, 0.00Mb streamed
Approx Size: 102x39x102
Num Collision Primitives: 1
Estimated Compressed Disk Size: 0.26 MB (0.00 MB Nanite)

This second image here is the same tyre with 20 percent of the total polygons.
The underlying function, which Unreal provides, does it's best to determine which nodes and vertices can be removed, while changing the visual representation as little as possible.

I think it does incredibly well, although, there is some noticeable 'jaggedness' around the rim.
After some basic testing on the chassis and tyre meshes, I believe a 50 percent reduction offers the best trade-off when lowering the number of polygons while creating minimal visual change..
I imagine the ideal solution would require seperating sections of a mesh and assigning individual LOD settings to each section. I believe this requires a dynamic mesh.

(Next steps)
Convert static mesh to dynamic mesh,
Adjust Settings,
Convert back to Static Mesh

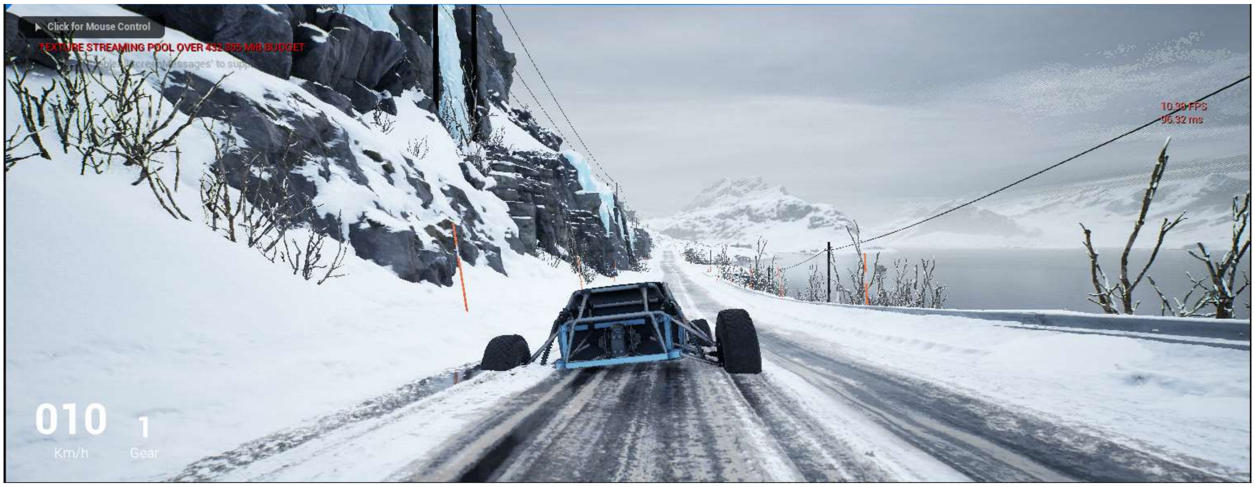
(I struggled with this but hope to come back to it in future. Seems like it would be the ideal solution).

HLOD's:

Unreal HLOD's (Hierarchical LOD):
https://dev.epicgames.com/documentation/en-us/unreal-engine/hierarchical-level-of-detail?application_version=4.27

Using Unreal asset "Automotive Winter Scene".

Hierarchical LODs are the easiest way to lower polygon count at runtime. They will essentially merge meshes into a single asset, which can drastically reduce poly count.
Adding an HLOD into a complex scene reduced the triangle count by 23% (12 million triangles to 2.8 million). However, this required a large amount of extra computation which slowed the scene down (Lower fps, Higher CPU Utilization), and resulted in some collision issues..



TEXTURE STREAMING POOL OVER 432.355 MiB BUDGET

'DisableAllScreenMessages' to supp

10.38 FPS
96.32 ms

This supports the conclusion that it would be better to reduce the asset quality before runtime.

Defeaturing:

Unreal Defeaturing:
https://dev.epicgames.com/documentation/en-us/unreal-engine/simplifying-static-meshes-by-defeaturing?application_version=4.27

Deprecated. The required plugins are no longer free.
There are still some features in the Unreal 'modeling mode' which can be useful. But, once more, these features are adjusted for each individual mesh, outside of runtime.

I have tested some of these features on a simple car asset. They did not make a noticeable difference for an asset that is already fairly low res. But they offer useful features which would greatly benefit certain assets, such as - Filling holes and joining overlapping edges.
Defeaturing helps to smooth over and remove unecessary polygons.

The modeling mode also allows the ability to fraction the total number of triangles, with additional functionality. Firstly, the simplification will save the simplified model (Unlike LODs which simplify at runtime). Secondly, a simplification algorithm can be chosen. Each has it's benefits, and the one used will depend on the asset. The default QEM (Quadratic Error Model) works fine for my testing purposes.

Before Simplification and LOD change (approx. 1.9 million triangles)

After Simplification and LOD change (approx. 496,000 triangles)