# Real-time Adaptive Use of Surface Tessellation & Subdivision

## Background

Surface tessellation has been used in graphics engines for an extremely long time. It is generally used to boost the fidelity of surfaces on higher-power machines. There are 2 main methods of tessellating in current graphics environments: "Loop" and "Catmull-Clark". These methods are both used throughout the industry, with "Loop" being favoured by graphics rendering engines due to it working well with triangle-based meshes and "Catmull-Clark" being favoured for modelling due to it using quad-based meshes as shown in Figure 1 and 2 respectively.

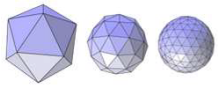

Figure 1: Loop Surface Subdivision (Fuhrmann, S. 2009

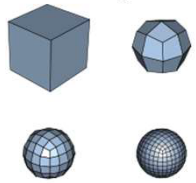

Figure 2: Catmull-Clark Surface Subdivision (Romainbehar, 2006)

## Motivations

The current tessellation in rendering engines tends to be done in a static manner. The tessellation value set by the user tends to only affect the max tessellation done by the graphics system when close to the user. However, doing this means losing efficiency in low detail environments as the user will wish to keep stable framerates in higher detail environments.

My objectives are to:

- Develop a rendering pipeline within a current highly used rendering engine i.e. Direct3D, Vulkan etc.
- Create and implement an optimised tessellation system that changes the tessellation of object meshes in real-time to allow for greater fidelity whilst keeping performance.
- Potentially optimise this system to work on the GPU instead of strictly CPU.

Doing these should hopefully fill a gap in rendering development to allow for more flexible use of surface tessellation and subdivision.

## Discussion

A literature review was used to understand the current ways that tessellation is used in rendering engines as well as potential solutions to some of their flaws. The review focused on two section; tessellation techniques and subdivision techniques.

The review shows that most of the work on surface subdivision being done is to combine the current subdivision schemes to allow for a more varied selection of models to be used as shown in Figure 3.

Figure 3: Output from paper on Composite Subdivisions for 3D Quad-Triangle Meshes. (Kok-Why, S. W. Andi and Khairil Imran, 2012)

The review into the tessellation techniques show more into optimising the tessellation system, such as using a mesh shader pipeline and a view dependant tessellation metric as shown in Figure 4.

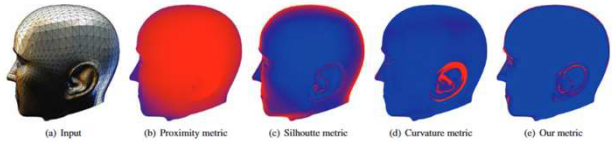


Figure 4: Adaptive Tessellation Metrics from A View-dependent Adaptivity Metric For Real Time Mesh Tessellation. (Boubekeur, A view-dependent adaptivity metric for real time mesh tessellation, 2010)

Neural subdivision is also being investigated by these papers, where machine learning is used to generate the subdivided meshes, these meshes seem to have a higher detail than the current standard techniques as in Figure 5.


Figure 5: Neural Subdivision. (Youwei, Qingping, & Lamei, 2007)

## Methodology

My methodology will potentially be adapting the "Adaptive Tessellation Matrix" from Figure 2 into a rendering engine using the Vulkan API. I may also use the information and knowledge from other research papers that also work on subdivision in modern rendering engines. However, I will not be using "Neural Subdivision" due to the complexity of using machine learning and my lack of hardware for the task.

I'll be using the "Agile" methodology to split the project into the following chunks to complete the project.

Initially, I will be using the Vulkan Documentation (Vulkan®, n.d.) to create a basic rendering engine.

I will then use this to test performance with current tessellation and subdivision systems by taking readings such as average frames per second, average frame times and 1% lows.

Subsequently, I will be developing a variant of the tessellation and subdivision systems that works in real time and adapts based on the above-mentioned statistics.

I'll then test the performance of the system after these changes have been made and conclude as to whether this system completes its function properly.

## References


- Boubekeur, T. (2010). A view-dependent adaptivity metric for real time mesh tessellation. 2010 IEEE International Conference on Image Processing (p. 4)
- Fuhrmann, S. (2009), Loop Surface Subdivision.
- Kok-Why, N., S. W, A., & Khairil Imran, G. (2012). Composite subdivisions for 3D quad-triangle meshes. 2012 International Conference on Computer & Information Science (ICCIS) (p. 4).
- Liu, H., Kim, V. G., Chaudhuri, S., Aigerman, N., & Jacobson, A. (2020). Neural subdivision. ACM Transaction on Graphics, Vol. 39, No. 4, 16.
- Romainbehar, (2006) Catmull-Clark Surface Subdivision.
- Vulkan®. (n.d.). Khronos Vulkan® Tutorial. Retrieved from Vulkan Documentation