# Advanced Real Time Rendering Shadows Report

## Shadow Mapping Implementation

The shadow mapping that I have used in this assignment, primarily revolves around rendering the scene depth (Z-buffer) into a texture from the perspective of a light. The scene is initially rendered to obtain the Z buffer data. Then a 2nd pass is used to perform the main rendering. Each pixel in the shadow map is then compared to the pixels in the scene image and if the pixel selected is closer to the light than the corresponding pixel in the shadow map it is drawn normally. If it is further, then it will be drawn into a shadow. This allows different heights to have the same shadow rendering across from the same object casting the shadow. It also means that ordering of objects is not an issue since the division between an object being in shadow or not is not down to distance or objects in the way but in effect a binary yes/no once the shadow map has been calculated.

## Advantages and Disadvantages of Shadow Mapping

Shadow mapping is used to cast shadows in a 3D scene with varied geometry including curved surfaces and hilly terrain. Shadow volumes in contrast divides the world into two regions, with one region being an area not in shadow and the other area being an area within shadow. Shadow volumes uses the stencil buffer whereas shadow mapping uses the Z buffer with both stored next to each other within RAM. From this point on I will use the shorthand of “mapping” and “volumes” to refer to shadow mapping and shadow volumes respectively.

Mapping is less accurate than volumes but is faster and easier to implement depending on the fill time available on GPU hardware. Consequently, if fill rate is an issue it may not be suitable for real time rendering in a complex scene and other techniques including shadow volumes may be superior for a given project, but it is faster. Resolution can be easily modified within the algorithm, with higher resolution requiring more resources. Mapping does well with objects that are not sorted, that a collection of objects aligned in a sequence from the light source to the final object in the path of a light does not need to be ordered to calculate if it is in shadow or not. This is due to the calculations using the Z buffer pixel by pixel. Calculation complexity therefore is linear and quadratic or worse. General rendering speed will be approximately double the speed of rendering the scene without shadows due to the need to perform a shadow render that includes grabbing scene depth in addition to the main render. The actual speed may be a little lower or higher depending on the complexity of shadows required to be rendered and other graphical effects present within the scene, particularly if said effects are also included within the shadow mapping technique. Mapping tends to be the most commonly used technique according to Sintorn et al. (2011), in great part due to good hardware support, ability to handle arbitrary geometry and low variability in frame times.

Shadow volumes are more accurate but may suffer from some issues with silhouette edges during the construction of a shadow volume. This is often termed self-shadowing where a shadow may project its own shadow forward inappropriately. Front and back end capping used to prevent shadows bleeding out into these artefacts can be used depending on the implementation. This need to correct for potential errors makes the implementation somewhat difficult to achieve when compared to mapping. Optimisation is available to use a viewing frustum to eliminate shadows that are outside the viewing plane.

On the downside with volumes, it can be extremely CPU intensive. This is due in large part to the size of shadow volume polygons, with excessive fill times being an issue for convex shapes. Eye in shadow problems can emerge if the camera lies within the shadow quad generated, which can only be solved when the lights view is aligned with that of the camera plane and the light’s far plane is set to equal the camera’s near plane.

## References

Developer.download.nvidia.com. (2017). [online] Available at: http://developer.download.nvidia.com/SDK/10.5/opengl/src/cascaded\_shadow\_maps/doc/cascaded\_shadow\_maps.pdf [Accessed 15 Dec. 2017].

GameDev.net. (2017). *The Theory of Stencil Shadow Volumes*. [online] Available at: https://www.gamedev.net/articles/programming/graphics/the-theory-of-stencil-shadow-volumes-r1873 [Accessed 15 Dec. 2017].

Sintorn, E., Olsson, O. and Assarsson, U. (2011). An efficient alias-free shadow algorithm for opaque and transparent objects using per-triangle shadow volumes. *ACM Transactions on Graphics*, 30(6), p.1.