GPU shaders for advanced visualisation

Visualisation Report

M.Sc. Software Engineering for Technical Computing

Aman Makkar

S321839

Abstract

This report has been compiled to provide a summary account of the Shader technology for advanced visualisation in the computational engineering. The purpose is to provide an understanding of how shader technology has changed over the years and how improvements in GPUs is helping the industry grow, this includes the gaming industry as well as films, Science and space technology.

1. Introduction

Shader technology provides the capability of running a piece of code directly on the Graphics processor unit, it gives the programmer the freedom to be able to manipulate images before they are displayed on the screen. Shaders technology initially started in the early 1980s in the filming industry, at that time it was specifically used for special affects (Hergaarden, 2011). Shader technology was only limited to pixel shaders at the time however, other shaders were introduced to the industry such as vertex and geometry shaders. Carrying out GPU programming in the early 1980s was a difficult task as every computer needed to programmed individually with no portable GPU API available. Therefore, in the early 1980s, two such APIs were released, and these are the OpenGL and DirectX APIs.

In the early 90s when the two APIs were released, there were still constraints on the technology in terms of what can be achieved. As there was no industry support for the shader technology, only the fixed-function pipeline (FFP) was available, although it allowed customisation to the rendering process, it was not possible to add custom algorithms to the shaders, leading the program to be rather rigid (Hergaarden, 2011). This changed in early 2000s when the shader technology started gaining industry wide support, and this has led to improvements in GPU programming over the years.

2. GPU Shaders

There are different kind of shaders available such as pixel/fragment shaders, vertex sharers and geometry shaders.

2.1. Vertex Shaders

Vertex shader has the freedom to be able to change the vertices of a given shape however, it has one important responsibility and that is to set the final position of the shape on to the screen. This is an important task as it has to be able to show 3d shapes on a 2d screen (*Aerotwist - An Introduction to Shaders - Part 1*, n.d.). Furthermore, vertex shaders can also remove vertices from a given shape however, they cannot be added back by the vertex shader, that work is carried out by the geometry shader (Hergaarden, 2011).

2.2. Fragment Shaders

Fragment shader has the capability of providing texturing and lighting to a given shape however, the fragment shader's important responsibility is to set the final colour of the shape on to the screen. Fragment shaders uses the data provided by the vertices to draw the pixels within the given vertices of a triangle (*Aerotwist - An Introduction to Shaders - Part 1*, n.d.).

2.3. Geometry Shaders

Geometry shaders have the capability to carry out operations on complete geometry of a shape which are represented by vertices (*Introduction to Shaders - Arction*, n.d.). Geometry shaders are relatively new and not so well supported by the industry yet however, this is growing (Hergaarden, 2011).

3. Uses of Shader Technology

3.1. Shaders in Astronomy

As the field of Astronomy carries on growing, with the increasing use of radio telescopes and radio interferometers, the amount of data generated that needs to be evaluated has increased, specifically the need to evaluate images of distant planets and galaxies (Vohl et al., 2017).

Astronomical images show range of densities, temperatures and dimensions with density over several magnitudes and temperatures that range from 10^4 and 10^8K (Kaufman et al., n.d.) . There are various issues that are faced by scientists when using visualisation for Astronomical objects and one of the biggest one is when the small-scale features are blurred out in the image. Although there are a few solutions to this issue, such as using higher-resolution images and renderings or zoom into the domain. Zooming into the images can cause its own problems such as large domains getting too large that can lead to misinterpretations (Kaufman et al., n.d.).

To improve the visualisation of the images and read the particle data the use of OpenGL is one of the most used APIs. The points provided fit well into the pipeline architecture of OpenGL and they can provide highly detailed images of Astronomical data (Kaufman et al., n.d.). Due to the flexibility of OpenGL custom shaders have been developed that can be specific to the user needs. A such shader was developed to be able to model Saturn and it's rings based on the data returned by the Voyager and Cassini spacecrafts. In order for this shader to be developed, various things had to be accounted for, such as Saturn's self-shadowing, the shadows cast by the Saturn's rings on Saturn, the Shadow cast by Saturn on to its rings and so on (Fajardo Hernandez & Pomarède, n.d.). As can be seen in figure 1 below, the flexibility of OpenGL was very useful in providing a high-resolution image of Saturn and one of its moons.

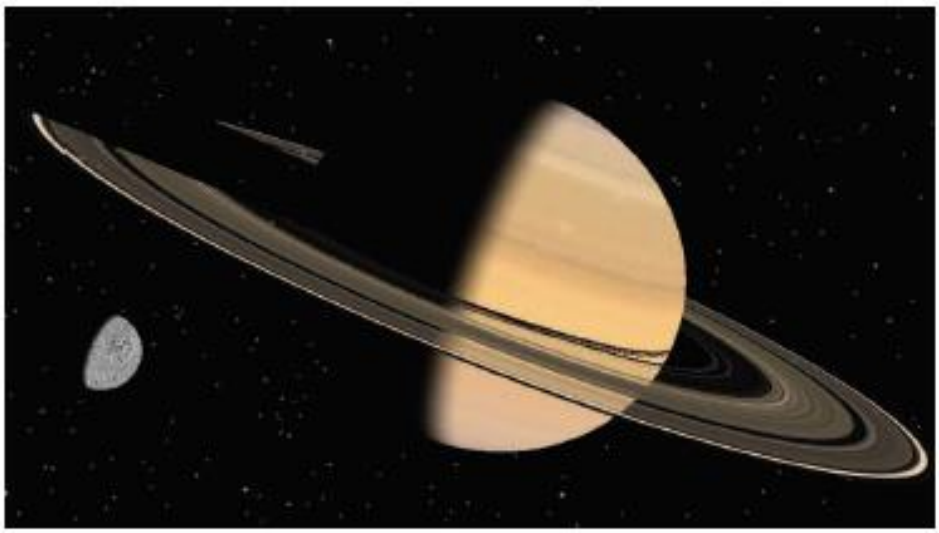

Fig 1: Shadow effects of Saturn using GLSL (Fajardo Hernandez & Pomarède, n.d.)

3.2. Shaders in Gaming

Shaders in the gaming industry are used for laying out details of shadows, lighting and texture. The flexibility of openGL Shading Language (GLSL) is very useful and it is based on the C programming language, due to this flexibility custom Shaders can be created that are specific for game development. As GLSL only works in conjunction with OpenGL rather than independently it provides an API for pixel shaders (Howlett et al., n.d.). Figure 2 below shows the difference shaders make to a game.

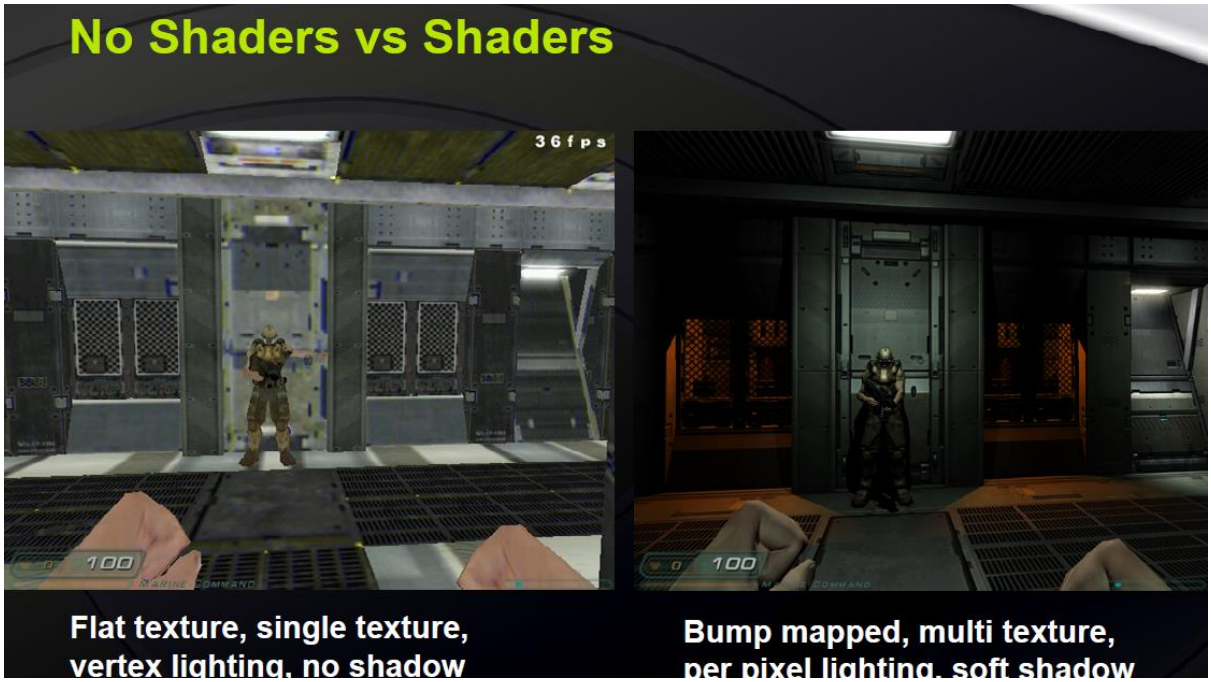

Fig 2: The pic on the left without shaders and on the right with shaders (*Integrating Shaders into Your Game Engine Bryan Dudash NVIDIA Developer Technology*, n.d.).

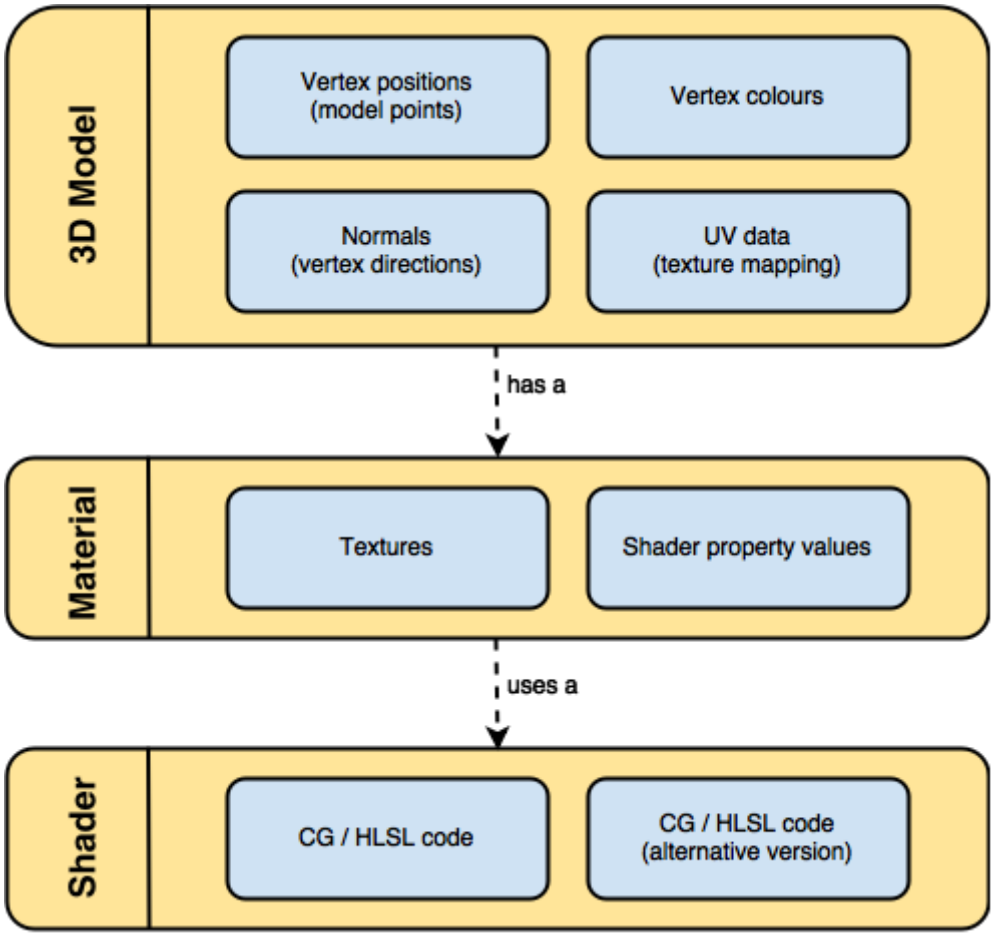

Fig 3: Shows what is needed for graphics in a video game (*Meet the Shaders : Vertices, Polygons and Meshes | by Binigya Dahal | Medium*, n.d.)

In the gaming industry, shaders are used to render an object, which includes the code and mathematical calculations that are required for lights, viewing angle and so on, these parameters can also be modified for maps and colours (*Meet the Shaders : Vertices, Polygons and Meshes | by Binigya Dahal | Medium*, n.d.). The job of a shader is to give instructions to the GPU on how to display an image on the screen based on the calculations provided in the shader.

4. Conclusion

Above is a brief breakdown of how the shader technology is used in Astrophysics and the gaming industry. In the Space industry, shaders are used for images that have been returned by various spacecrafts that may be orbiting a distant planet or looking out at far out galaxies. This data can be very large and to see the images clearly to understand the environment in space, it is essential that the images that are sent back can be evaluated correctly, and this is where the shader technology is very useful. In the gaming industry, before the use of shader technology, most of the games looked and felt almost the same, leaving the environment and the feel of the game feeling lifeless however, since the shader technology made its way into the gaming industry in the early 2000s, the gaming industry has transformed with every game feeling different to any of the other games that are available in the market. There are obviously various other industries where shader technology is essential such as films, medical imaging and so on.

5. Bibliography

- *Aerotwist - An Introduction to Shaders - Part 1*. (n.d.). Retrieved March 4, 2021, from https://aerotwist.com/tutorials/an-introduction-to-shaders-part-1/
- Fajardo Hernandez, E. M., & Pomarède, D. (n.d.). *Interactive Visualization of Shadow Effects in the Planetary System of Saturn, its Rings and its Moons using an OpenGL Shader in IDL* (Vol. 461).
- Hergaarden, M. (2011). Graphics Shaders. In *Graphics Shaders* (Issue January). https://doi.org/10.1201/b11316
- Howlett, A., Colton, S., & Browne, C. (n.d.). *Evolving Pixel Shaders for the Prototype Video Game Subversion*. Retrieved March 4, 2021, from www.introversion.co.uk/subversion
- *Integrating Shaders into Your Game Engine Bryan Dudash NVIDIA Developer Technology*. (n.d.).
- *Introduction to Shaders - Arction*. (n.d.). Retrieved March 4, 2021, from https://www.arction.com/articles/introduction-to-shaders/
- Kaufman, A., Fan, Z., & Petkov, K. (n.d.). *Visualization needs and techniques for astrophysical simulations Related content Implementing the lattice Boltzmann model on commodity graphics hardware*. https://doi.org/10.1088/1367-2630/10/12/125008
- *Meet the Shaders : Vertices, Polygons and Meshes | by Binigya Dahal | Medium*. (n.d.). Retrieved March 4, 2021, from https://medium.com/@darkdreamday/meet-the-shaders-vertices-polygons-and-meshes-1dde115c4bc6
- Vohl, D., Fluke, C. J., Barnes, D. G., & Hassan, A. H. (2017). Real-time colouring and filtering with graphics shaders. *MNRAS*, *471*, 3323–3346. https://doi.org/10.1093/mnras/stx1676

Appendix 1. GPU Class Diagram

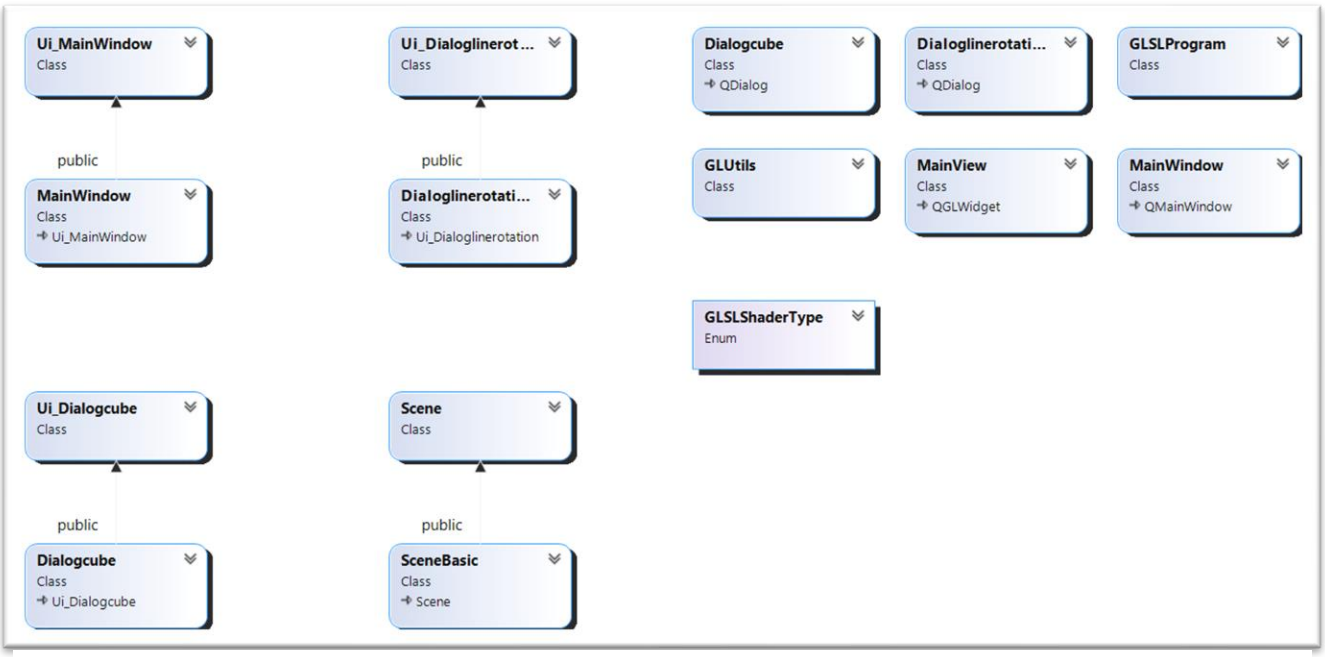

Figure 4 GPU Class Diagram