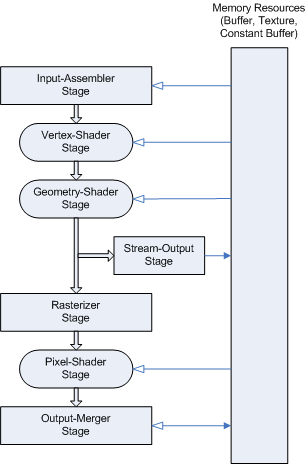
**Report**

Computer shaders are programmes used to render scenes and they run on the GPU. With the help of shaders, the stages of the graphics pipeline can be controlled. They can be programmed in assembly language or shading languages, like HLSL. Graphics APIs, like DirectX and OpenGL support shaders. Before shaders there was a fixed function pipeline, which wasn’t possible to program. It was based on user provided configuration and lacked flexibility. (Heergarden, 2011; OpenGL)

A screenshot of a cell phone

Description automatically generated

Figure 1: Fixed function pipeline (Bailey and Cunningham, 2009)

Figure 2: Programmable pipeline (Microsoft, 20218)

There are three main types of shaders: vertex shader, pixel shader and geometry shader. (Heergarden, 2011)

Vertex shaders operate on each vertex. It is used to transform the vertex’s 3D position into 2D viewport space. Normal mapping, manipulating textures and the position of the vertices can be done in the vertex shader too. (OpenGL, Bailey and Cunningham, 2009, Heergarden, 2011)

Pixel shaders (or fragment shaders) operate on each pixel. It can calculate the colour and transparency of each pixel, calculate lighting, shadows, and depth of field. (Bailey and Cunningham, 2009)

Geometry shaders can generate new primitives from the primitives that were sent to it. It is optional. (OpenGL) It was introduced in DirectX 10. (Anguelov, 2011)

**Techniques used in my shaders:**

**Lighting**

There are 5 lights.

**Changing texture colours**

Two textures are fading from one to the other on a cube.

Two diffuse maps have to be passed to the shader. In the pixel shader the lerp function blends the textures. Both textures have to be used in the lighting calculation.

A screen shot of a person

Description automatically generated

**Wiggling Sphere**

A sphere model is performing a wiggling motion while the texture is rotating.

“*Very trippy.*”

* a friend

The vertices of the model are manipulated in the vertex shader using sin and cosine to achieve the pulsating motion. The wiggle variable is used to control the speed.

A screenshot of a cell phone

Description automatically generated

The texture coordinates are scrolled in the pixel shader to make the sphere appear rotating.

A close up of a clock

Description automatically generated

**Blending**

Blending combines the viewport pixels with the source pixels.

**Additive**

Additive blending can be used for a lightening effect. Can be used for lights, fire etc.

The equation: ***Final.Red = Bitmap.Red + Viewport.Red***

A screen shot of a computer

Description automatically generatedI implemented the additive blending state in State.cpp.

Then I set the blending state before rendering the model in Scene.ccp.



**Multiplicative**

Multiplicative blending can be used for a darkening effect, good for glass, smoke and shadow.

The equation: ***Final.Red = Source.Red \* Destination.Red***

Implementation in State.cpp

A screen shot of a person

Description automatically generated

And setting the blending state in Scene.cpp



**Alpha Testing**

It allows us to check the alpha value of polygons before rendering. A cut out effect can be achieved. Sorting problems don’t affect it as they do alpha blending.

Alpha testing can be done in the pixel shader.

A screenshot of a cell phone

Description automatically generated

Before rendering the right pixel shader has to be set.

**Normal Mapping**

This is a light effect that creates the illusion of bumpiness on a model. It requires a normal map in addition to the base texture.

A screen shot of a computer

Description automatically generatedThe normals have to be transformed on to the mesh. The calculations are done in the pixel shader:

Both the normal map and the diffuse specular map have to be sent to the shader before rendering and the right shaders have to be set.

A close up of a sign

Description automatically generated

**Parallax Mapping**

This effect creates bumpiness but also depth on the model. It needs a height map which is stored in the alpha channel of the normal map. The normal map map has to be sent to the shader.

The calculations are done in the pixel shader:

A screenshot of a cell phone

Description automatically generated

A picture containing photo, black, red, sign

Description automatically generatedThe right shaders also have to be set before rendering.

**Normal mapping and changing texture colours combined**

In this case two diffuse maps and a normal map has to be passed to the shader, and we need to make sure that they are in the right slot.

The same calculations are done in the pixel shader as with a *normal* normal map. The two textures have to be blended with the lerp function.

In the light calculation both textures need to be used.

A screen shot of a person

Description automatically generated

**References**

Anguelov, B. (2011*) DirectX10 Tutorial 9: The Geometry Shader*. Available at: <https://takinginitiative.wordpress.com/2011/01/12/directx10-tutorial-9-the-geometry-shader/>. [Accessed: 28 Apr 2020]

Bailey, M., Cunningham, S. (2009) *Graphics Shaders: Theory and Practice.* CRC Press. Boca Raton

Diard, F. (2007) GPU Gems 3 ch. 41.

Heergarden, M. (2011). *Graphics shaders*. Available at: <https://www.cs.vu.nl/~eliens/download/literatuur-shaders.pdf>. [Accessed: 28 Apr 2020]

OpenGL (n.d.) *Fixed Function Pipeline.* Available at: <https://www.khronos.org/opengl/wiki/Fixed_Function_Pipeline>. [Accessed: 28 Apr 2020]

OpenGL (n.d.) *Geometry Shader.* Available at: <https://www.khronos.org/opengl/wiki/Geometry_Shader>. [Accessed: 28 Apr 2020]

OpenGL (n.d.) *Vertex Shader.* Available at: <https://www.khronos.org/opengl/wiki/Vertex_Shader> [Accessed: 28 Apr 2020]

Microsoft (2018) *Pipeline Stages (Direct3D 10).* Available at: <https://docs.microsoft.com/en-gb/windows/win32/direct3d10/d3d10-graphics-programming-guide-pipeline-stages?redirectedfrom=MSDN>. [Accessed: 28 Apr 2020]