Introduction to the Graphics Processing Unit

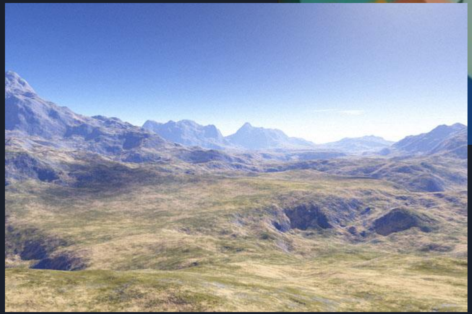
An overview of Computer Graphics, Render Pipelines and APIs

Programming – Computer Graphics

aie
SPECIALIST EDUCATORS IN
GAMES, ANIMATION & FILM VFX

Computer Graphics

- Computer graphics is any sort of visualisation created and displayed by a computer
- Early imagery from the 1950's created vector-based displays using series of lines
- Methods of rasterising bitmap imagery and geometry enabled images with colour and texture variation
- Eventually dedicated hardware was created to calculate the costly methods needed to display complex graphics
  - This created an explosion in visual fidelity



aie
SPECIALIST EDUCATORS IN
GAMES, ANIMATION & FILM VFX

The Graphics Processing Unit (GPU)

- Initially developed to increase visual performance when drawing basic shapes and speed up bitmap manipulations

- Processors capable of 3D manipulation and Hardware Texture & Lighting (T&L) were eventually released

- Each manufacturer implemented their own way for rendering graphics
  - This caused problems for PC game developers dealing with the vast array of hardware configurations

aie
SPECIALIST EDUCATORS IN
GAMES, ANIMATION & FILM VFX

Introduction of Standards

- The need for standards was agreed upon and a set render pipeline was formalised with API developers leading the charge

- Multiple APIs capable of manipulating these units
  - OpenGL
  - Direct3D (part of the DirectX set of APIs)
  - Vulkan
  - Metal

aie
SPECIALIST EDUCATORS IN
GAMES, ANIMATION & FILM VFX

APIs

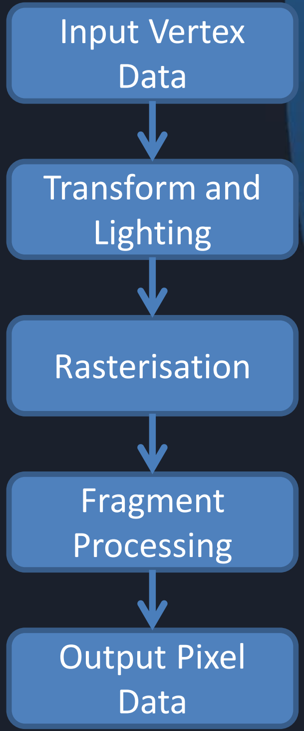
- The various APIs are a way for programmers to interface with the GPU drivers without having to write code specific for each device driver
  - The API developers and the hardware manufacturers make sure they work for us

- APIs do have limitations though
  - Direct3D only applies to Microsoft operating systems
  - OpenGL is cross-platform across PC hardware
  - OpenGL ES is commonly used on mobile devices
  - Metal only applies to Apple operating systems

aie
SPECIALIST EDUCATORS IN
GAMES, ANIMATION & FILM VFX

The Early Render Pipeline

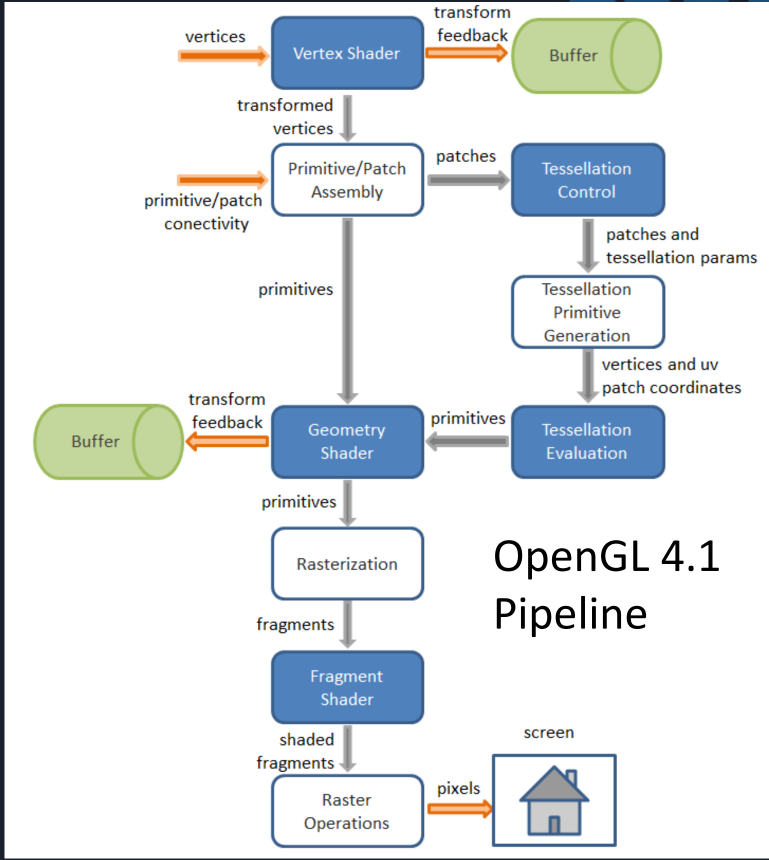
- A render pipeline refers to the method and stages of rasterisation for graphics hardware
  - Rasterisation refers to converting vector-based graphic images into a raster image consisting of pixels
  - Geometric data is input into the pipeline, and coloured pixels are drawn as output
- In the beginning the amount of customisation was limited
  - An artist could create a mesh and textures, choose colours for a light or two, but that was all that could be done to differentiate the look of their game from other games



2000-2012 Render Pipeline

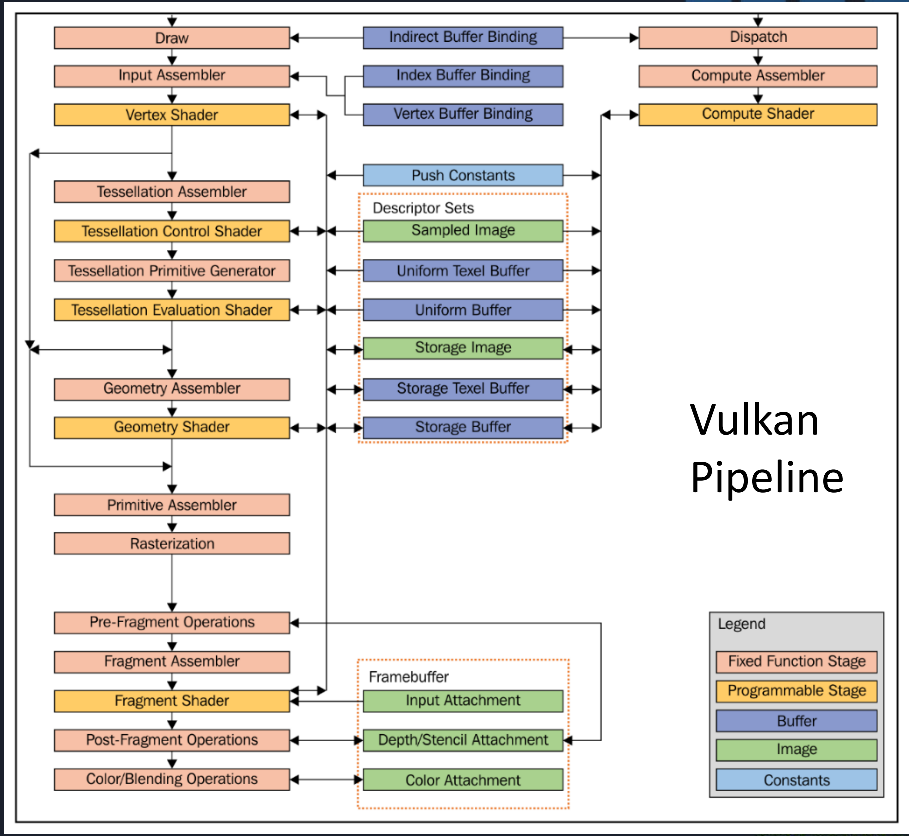
- GPU hardware improved greatly, with new features allowing for more complex render pipelines
  - Increased GPU speed
  - Increased GPU RAM
  - Increased register and operation counts
  - Programmable and Fixed stages

- Game developers could now completely customise how their products look



Modern Render Pipelines

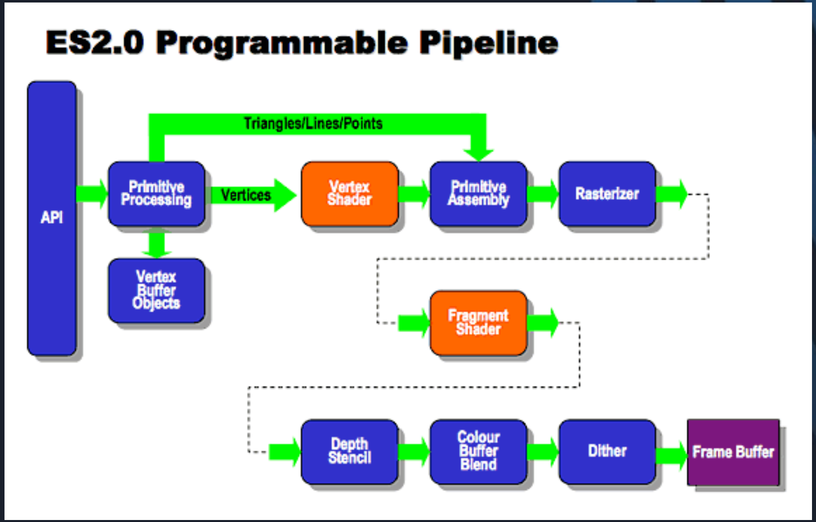
- Current generation pipelines are complex with many programmable stages
- Data can now flow both ways between the GPU and CPU at multiple points in the pipeline via buffers
  - Many inputs and many outputs, not just input geometry and output pixels
- Compute stages allow custom rendering or advanced computation of almost any type of data



Offshoot Render Pipelines

- Other pipelines exist
  - OpenGL ES (Embedded Systems)
  - WebGL for web sites (resembles OpenGL ES)

- These are usually cut-down versions of standard pipelines and contain less stages due to hardware limitations on target devices



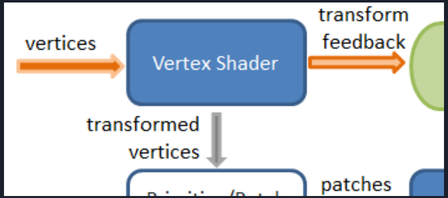
Buffer Objects

- The inputs and outputs of a typical Render Pipeline are called Buffer Objects
- There are different types of buffers, usually they are simply arrays of data
  - Vertex Buffer Object, containing the points that make up geometry for a mesh
  - Index Buffer Object, containing topology information for the geometry, i.e. triangles
  - Various image buffers, usually containing image pixel data, which can include data representing the screen being displayed or textures mapped onto geometry
- Buffers can be used as input to various stages
  - Vertex and Index buffers are used as input to the pipeline
  - Image buffers can be accessed at different points through the pipeline
- Some stages in a render pipeline may also output buffers
  - For example, the final output of the pipeline usually goes into an image buffer

aie
SPECIALIST EDUCATORS IN GAMES, ANIMATION & FILM VFX

The Vertex Shader Stage

- Once the buffers have been setup and sent to the GPU the first stage they usually enter when we make a render call is the Vertex Shader stage



- This is a programmable stage that processes each vertex in a vertex buffer individually and separate from each other
  - Multiple vertices are processed at once through simple functions

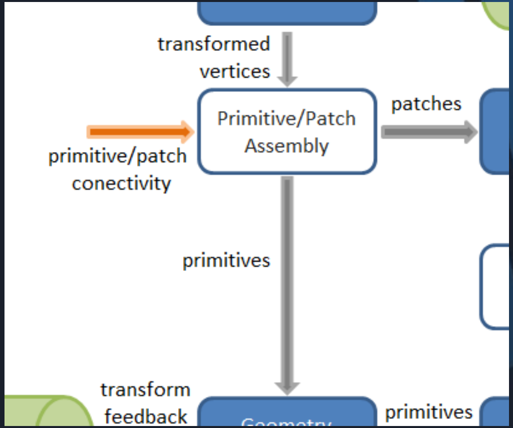
- Typically the task of the Vertex Shader is to transform vertices from object space to screen space for later stages in the pipeline

aie
SPECIALIST EDUCATORS IN
GAMES, ANIMATION & FILM VFX

Primitive / Patch Assembly Stage

- The next stage is a fixed stage, and it is responsible for arranging the vertex data into primitives for the later stages
  - Makes use of vertex and index buffers

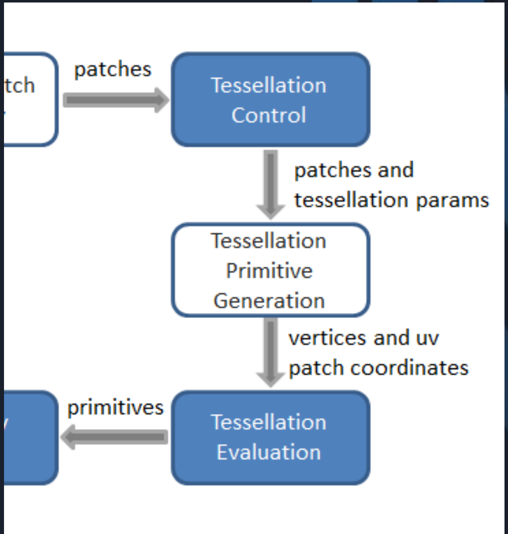
- This stage is usually controlled simply by specifying the primitive type when rendering
  - Triangles, lines, points, patches



- Outputs to either an optional Tessellation stage or to the Geometry Shader stage

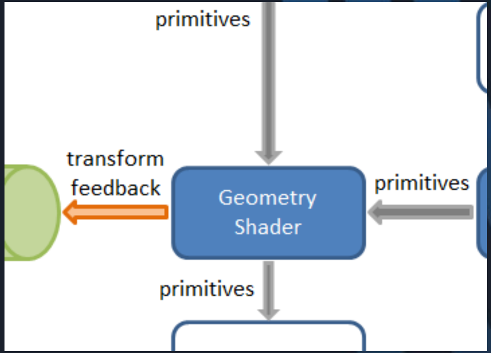
Tessellation Stages

- These are an optional set of three stages
  - **Tessellation Control** Stage (Programmable)
  - **Tessellation Primitive Generation** Stage (Fixed)
  - **Tessellation Evaluation** Stage (Programmable)
- The tessellator takes in patch primitives and splits them into multiple primitives, increasing the primitive count in a mesh
  - A patch primitive can have 1 or more vertices (32-max usually)
  - The Control stage defines how many times to split up the input patch
  - The Generation stage then generates the new primitives
  - The Evaluation stage receives these new primitives and then sends them through to the next stage in the render pipeline



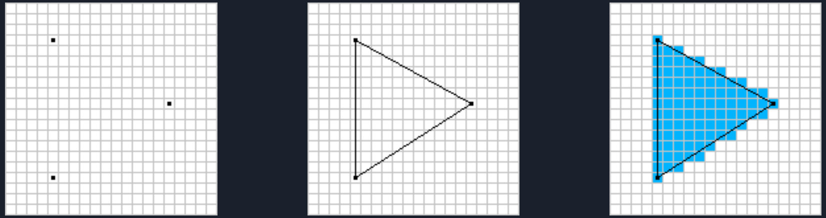
Geometry Shader Stage

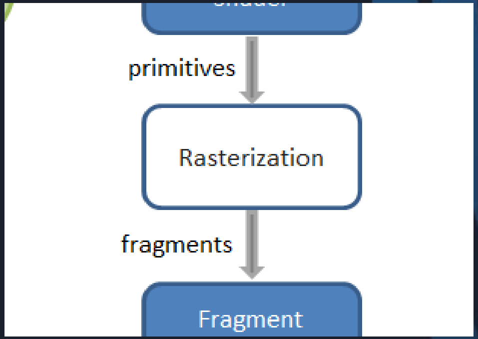
- An optional programmable stage
  - If not set then it passes primitives straight to the Rasterisation stage
- Receives all the information needed for a primitive
  - For example, 3 vertices in the case of a triangle, 2 for a line
- Can perform any last minute processing on the primitive



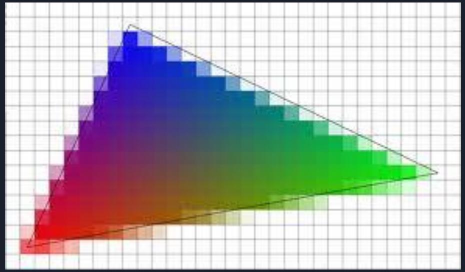
Rasterisation & Interpolation Stage

- This fixed stage receives vector-based primitives and plots out the pixels covered by the shape





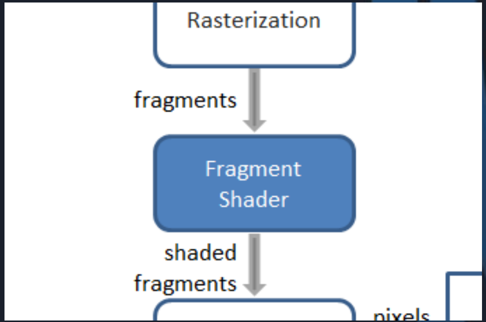
- It then interpolates the data from each vertex across the pixels covered, passing the interpolated data at each pixel to the next stage in the pipeline



SPECIALIST EDUCATORS IN GAMES, ANIMATION & FILM VFX

Fragment Shader Stage

- This is the final <u>programmable</u> stage in a typical Render Pipeline, but not the last stage



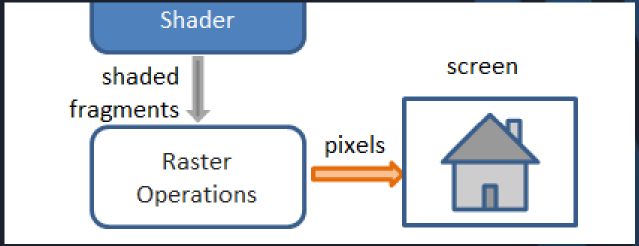
- Also called the **Pixel Shader**, as it receives the interpolated pixel data from the rasteriser

- Typically its job is to output a desired colour at the specified pixel, based on the input data, and by sampling other buffers usually containing texture information
  - Can output more than one pixel to more than one output buffer!

- However the data returned is not necessarily what appears on screen!

Raster Operations Stage

- The final stage in the pipeline is a fixed stage
- Receives data from the **Fragment Shader**
- Using flags, it blends the pixel with any existing pixel at that location within the output buffer



Final Output Data

- The render pipeline's job is to output all the pixels represented by geometry
  - The pixels of all geometry in a level can be combined into a final image buffer
- Typically the output buffer represents the screen but it can also output to other locations
  - Data can be rendered into an image to be used as a texture when rendering other geometry

aie
SPECIALIST EDUCATORS IN GAMES, ANIMATION & FILM VFX

Summary

- The render pipeline has many parts to it, and people have found numerous different ways of using it to achieve visuals of all types
  - Some of its most recent uses aren't even for visuals!

- The best way to understand it is to just tinker with it and see what types of visuals and effects you can come up with!

aie
SPECIALIST EDUCATORS IN
GAMES, ANIMATION & FILM VFX

Further Reading

- Akenine-Möller, T, Haines, E & Hoffman, N, 2008, *Real-Time Rendering*, 3rd Ed, CRC Press

- Haemel, N, Sellers, G & Wright, R, 2014, *OpenGL SuperBible*, 6th Ed, Addison Wesley

SPECIALIST EDUCATORS IN GAMES, ANIMATION & FILM VFX