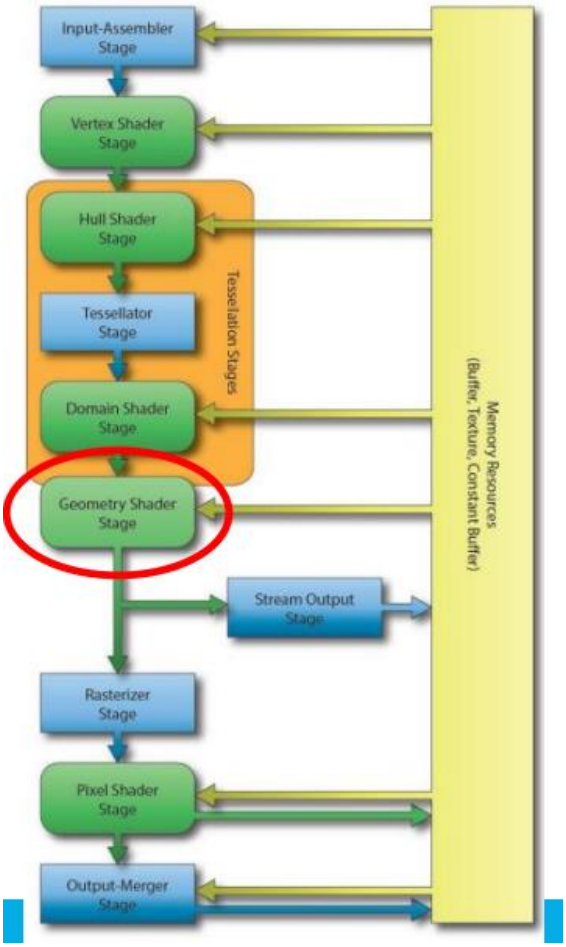
Lecture 3:

Rendering Pipeline (used by OpenGl and Direct3D, we focus on Direct3D):

Allows the programmer to perform many different operations on input data. Improves efficiency.



Input Assembler:

- Building block stage. Reads data from buffers into primitive format the can be used in pipeline. We mainly use triangle lists.

Vertex Shader:

- Shades a polygon based on the properties of vertices such as color. Also performs transformations on primitives.

Tessellation Stages (Hull, tessellator, domain):

- Adding more polygons/vertices and adding more details. Improves graphics quality.

Geometry Shader:

- Operates on entire primitive (triangle). Performs additional processing such as calculating normal, particle systems, shadow volume generation.

Stream Output Stage:

- Allows us to receive data (vertices/primitives) from the geometry shader and feed back into pipeline for processing by another set of shaders. This allows us to use the GPU for processing that would have be done by CPU. Necessary for particles systems, e.g waves/ripples.

Rasterizer:

- Projection of polygons (shapes and primitives) to pixels.
- **Culling:** Determining which polygons get rendered based on whether the vertices are ordered clockwise or anti-clockwise. (clockwise gets rendered)
- Culling is good for efficiency. (Things not facing camera should not be rendered).

Pixel (Fragment) Shader:

- Produces color values for interpolated pixel fragments + per pixel lighting.
- Perform lighting and coloring of pixels.

Output Merger Stage:

- Combines pixel shader output values to produce final image on computer.

Double Buffering:

- When an image isn't rendered you don't want to put it straight on the screen. So you do double buffering to make sure things are completely drawn. Reduces tearing (artifacts in rendering images).
- Can draw images in background while displaying finished images(efficiency).

Framerate: Number of times a screen is refreshed in a second. Number of times the data goes through the rendering pipeline.