Shader

Let us paint a World

Shader

a shader is a programm run on a gpu

it uses the mesh data for creating a picture

it renders normally directly on to the back buffer

Shader Languages

High Level Shading Language (HLSL)

OpenGL Shading Language (GLSL)

C for Graphics Effects (CgFx)

Kind of Shaders

Vertex Shader

Pixel Shader

Geometry Shader (since Version 4)

Hull Shader (since Version 5)

Domain Shader (since Version 5)

Compute Shader

Kind of Shaders

Vertex Shader

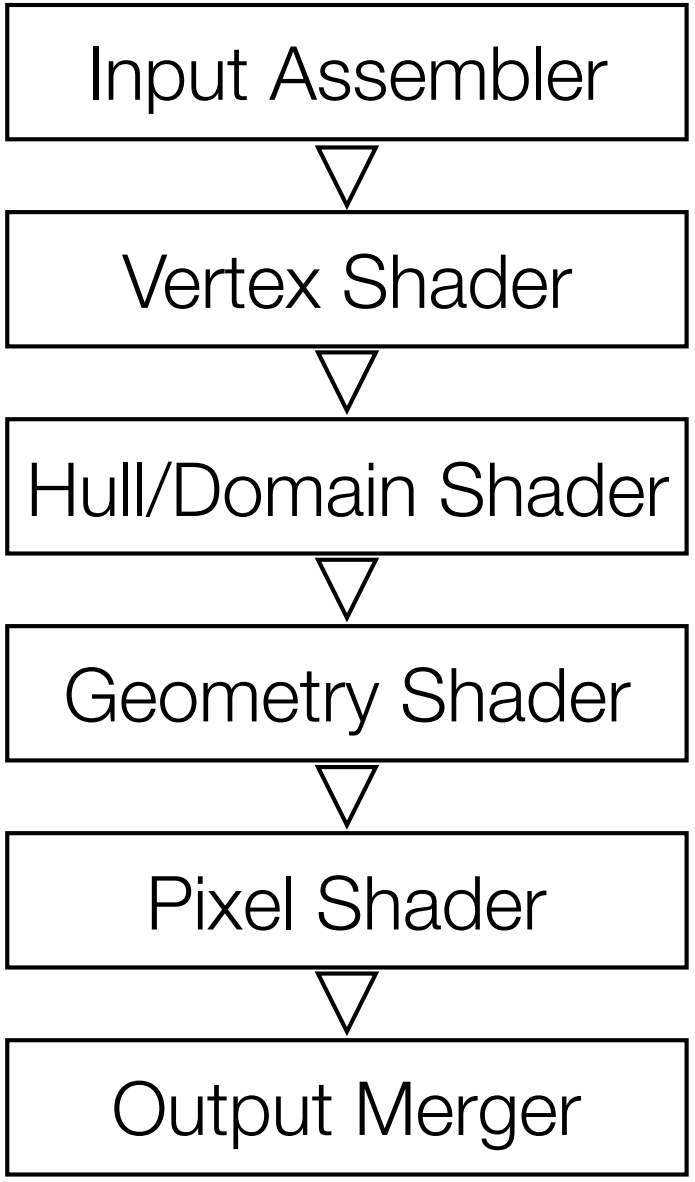
Pixel Shader

Geometry Shader (since Version 4)

Hull Shader (since Version 5)

Domain Shader (since Version 5)

Compute Shader



Vertex Shader

first shader in pipeline

directly after input assembler

performs operations on every single vertex

Pixel Shader

last shader in pipeline

calculate final pixel color and write it to the back buffer

can calculate a depth value and write it to the depth buffer

Geometry Shader

optional shader

deals with whole primitives like triangles, lines or points

can discard or create one or more primitives

e.g. for creating point sprites or volumes

Hull/Domain Shader

optional

subdivides meshes into smaller primitives (Tessellation)

normally based on mathematical functions, e.g. camera distance or edge length

Compute Shader

optional

used for additional calculation

not limited to graphics pipeline

used for heavy float calculation

Direct 3D

every shader has to be created for its own (e.g. ID3D11VertexShader)

every shader will be separately set and provide with data through device context (e.g. VSSetShader())

additional data can be provided to every shader, e.g.

- Textures (ID3D11ShaderResourceView)
- Samplers (ID3D11SamplerState)
- Constant Buffers (ID3D11Buffer)

Coding Time

Let’s rendering a mesh