Graphics Coordinates and the Pipeline

Mike Bailey
mjb@cs.oregonstate.edu

This work is licensed under a Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 International License.

The Basic Computer Graphics Pipeline

Vertex, Normal, Color

MC = Model Coordinates
WC = World Coordinates
EC = Eye Coordinates
CC = Clip Coordinates
NDC = Normalized Device Coordinates
SC = Screen Coordinates

The Basic Computer Graphics Pipeline, Shader-style

Vertex Shader

Fragment Shader

gl_Vertex, gl_Normal, gl_Color

Per-vertex in variables

Uniform Variables

gl_ModelViewMatrix, gl_ProjectionMatrix

Vertex Shader

Fragment Shader

gl_Position, Per-vertex out variables

Uniform Variables

gl_FragColor

Per-fragment in variables

Uniform Variables

MC = Model Vertex Coordinates
WC = World Vertex Coordinates
EC = Eye Vertex Coordinates

The Basic Computer Graphics Pipeline, Shader-style

Vertex Shader

Fragment Shader

gl_Vertex, gl_Normal, gl_Color

Per-vertex in variables

Uniform Variables

gl_ModelViewMatrix, gl_ModelViewProjectionMatrix

Vertex Shader

Fragment Shader

gl_Position, Per-vertex out variables

Uniform Variables

gl_FragColor

Per-fragment in variables

Uniform Variables

MC = Model Vertex Coordinates
WC = World Vertex Coordinates
EC = Eye Vertex Coordinates

MC = Model Coordinates
WC = World Coordinates
EC = Eye Coordinates
CC = Clip Coordinates
NDC = Normalized Device Coordinates
SC = Screen Coordinates
The Shaders' View of the Basic Computer Graphics Pipeline

- In general, you want to have a vertex and fragment shader as a minimum.
- A missing stage is OK. The output from one stage becomes the input of the next stage that is there.
- The last stage before the fragment shader feeds its output variables into the rasterizer. The interpolated values then go to the fragment shaders.

**Diagram:**

- **Fixed Function**
- **Programmable**

**Oregon State University**
**Computer Graphics**

mjb – December 31, 2019