Graphics Coordinates and the Pipeline
The Basic Computer Graphics Pipeline

Vertex, Normal, Color

MC -> WC

Model Transform

View Transform

Per-vertex Lighting

Projection Transform

Homogeneous Division

NDC

Viewport Transform

SC

Fragment Processing, Texturing, Per-fragment Lighting

Rasterization

SC

Raster Ops

Framebuffer

SC

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

Oregon State University
Computer Graphics
The Basic Computer Graphics Pipeline, Shader-style

**Vertex Shader**

- **gl_Vertex, gl_Normal, gl_Color**
- **gl_ModelViewMatrix, gl_ProjectionMatrix, gl_ModelViewProjectionMatrix**

**MC = Model Vertex Coordinates**

**WC = World Vertex Coordinates**

**EC = Eye Vertex Coordinates**

**Per-vertex in variables**

**Per-vertex out variables**

**Fragment Shader**

- **gl_FragColor**
- **Fragment Processing, Texturing, Per-fragment Lighting**

**Framebuffer**

**Rasterization**

**Uniform Variables**

**Per-fragment in variables**

MC = Model Vertex Coordinates
WC = World Vertex Coordinates
EC = Eye Vertex Coordinates
The Basic Computer Graphics Pipeline, Shader-style

Per-vertex \textit{in} variables

\begin{itemize}
\item gl\_Vertex, gl\_Normal, gl\_Color
\end{itemize}

Uniform Variables

\begin{itemize}
\item gl\_ModelViewMatrix, gl\_ProjectionMatrix, gl\_ModelViewProjectionMatrix
\end{itemize}

Per-vertex \textit{out} variables

\begin{itemize}
\item gl\_Position
\end{itemize}

Vertex Shader

Fragment Shader

Per-fragment \textit{in} variables

\begin{itemize}
\item gl\_FragColor
\end{itemize}

Framebuffer

Rasterization

Uniform Variables
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.