Robin Weiland

ECG pipeline

Layout

→ *I*nput *A*ssember Stage

- reads user-filled buffers with primitive data (points, lines and/or triangles)
- assemble this data into several different primitive types that will be used later
    - vertices into primitive types (triangle strip, line list, ...)
    - drop incomplete primitives
- `DirectX` attach system generated values (helps with shading)

↓ *Vertex Shader Stage*

- **input** single vertex
- **output** single vertex
- must be active for pipeline execution (at least simple pass-thru)
- geometry-processing:
  - coordinate/normal transformations (`Modelview Matrix`)
  - perspective projection of vertices (`Projection Matrix`)
- per-vertex operations:
  - transformations
  - per-vertex lighting
  - skinning
  - morphing

↓ Geometry Shader Stage

- **input** all vertices for a full primitive (1: `point`, 2: `line`, 3: `triangle`)
- **output**
- *when active* invoked for every primitive passed down to it
- algorithms:
  - point sprite expansion
  - dynamic particle system
  - Fin/Fur Generation
  - shadow volume Generation
  - single pass Render-to-Cubemap
  - per-primitive material swapping
  - per-primitive material setup (including generation of Barycentric coordinates for *PS*)

↘ Stream Output Stage

- **outputs** data into buffers in memory
- after *GS* or *VS*
- data can be fed back into the pipeline (e.g. multiple passes)
  - into *IA*
  - into programmable shader (e.g. with the `Load()` method)

↓ *Rasterizer Stage*

- **input**
- **output**
- converts vector information (shapes or primitives in $\mathbb{R}^3$) into a raster image (pixels)
- each primitive gets converted into pixels (pixel center) with interpolation across the primitive surface
- *RS* includes:
  - clipping vertices into view frustum
  - performing a division by z for perspective
  - mapping primitives into a 2D viewport
  - determining the invocation of the *PS*

↓ *P*ixel Shader Stage

- **input**
- **output** per-pixel *color values* from:
  - constant variables
  - texture data
  - interpolated per-pixel values
  - other data
- enables:
  - per-pixel lighting
  - post processing

→ Output Merger Stage

- **input**
- **output** final rendered pixel color form:
    - pipeline state
    - pixel data from *PS*
    - contents of render targets
    - contents of depth/stencil buffers
- determines which pixels are visible
- performs blending