The Graphics Pipeline Revisited

(thanks to David Luebke, University of Virginia)

Classic Rendering Pipeline

(as taught in Intro Graphics courses)

Classic Rendering Pipeline

What's wrong with this model (for an OpenGL system)?

• Model/view transforms combined
• Really “vertices” not “primitives”
  —Making this the vertex pipeline
• There's a lot going on in the “scan conversion” stage!
  —Primitive assembly
  —Rasterization
  —Texture mapping
  —Per-pixel lighting
  —Visibility (Z-buffer)
• We refer to these collectively as the pixel or fragment pipeline

High-Level Pipeline

Back up & think about the larger picture:

Application Processing

Geometry Processing ➔ Rasterization

• What sort of tasks does each stage perform?

High-Level Pipeline

<table>
<thead>
<tr>
<th>Application</th>
<th>Geometry</th>
<th>Rasterization</th>
</tr>
</thead>
<tbody>
<tr>
<td>(a.k.a. “vertex pipeline”)</td>
<td>(a.k.a. “pixel pipeline” or “fragment pipeline”)</td>
<td></td>
</tr>
<tr>
<td>Handle input</td>
<td>Transform</td>
<td>Rasterize (fill pixels)</td>
</tr>
<tr>
<td>Simulation &amp; AI</td>
<td>Lighting</td>
<td>Interpolate vertex parameters</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Look up/filter textures</td>
</tr>
<tr>
<td>Culling</td>
<td>Skinning</td>
<td>Z- and stencil tests</td>
</tr>
<tr>
<td>LOD selection</td>
<td>Calculate texture coords</td>
<td>Blending</td>
</tr>
<tr>
<td>Prefetching</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Graphics Pipeline: CPU

SIM

CULL

DRAW

DISPLAY
Graphics Pipeline: GPU

Note:
- Vertex processor does all transform and lighting
- Pipe widths vary
  - Intra-GPU pipes wider than CPU-GPU pipe
  - Thin GPU-CPU pipe
- Many caches and FIFOs not shown
- Soon: render-to-vertex-array

Here's what's cool:
- Can now program vertex processor!
- Can now program pixel processor!

Graphics Pipeline: GPU

Don’t forget! Not your father’s (or professor’s) GPU

Possible Bottlenecks