An Introduction to GPUs and WebGL

Patrick Cozzi
@pjcozzi
pjcozzi@siggraph.org

Analytical Graphics, Inc.
University of Pennsylvania
About Me

See http://www.seas.upenn.edu/~pcozzi/
Analytical Graphics, Inc.

- A few videos
University of Pennsylvania

- A few CIS 565 projects

Hao Wu

Ishaan Singh
Yingting Xiao
Xiaoyan Zhu

Mikey Chen
Vimanyu Jain
Today

- Graphics Pipeline
- GPU design
- WebGL
Part I: Graphics Pipeline
Graphics Review: Rendering

Model of a scene:
- 3D surface geometry (e.g., triangle mesh)
- surface materials
  - lights
  - camera

How does each triangle contribute to each pixel in the image?

Image credit: Henrik Wann Jensen

Graphics Review: Rendering

- Rendering
  - Goal: Assign color to pixels

- Two Parts
  - Visible surfaces
    - What is in front of what for a given view
  - Shading
    - Simulate the interaction of material and light to produce a pixel color
Graphics Pipeline Walkthrough

1. Vertex Assembly
2. Vertex Shader
3. Primitive Assembly
4. Rasterizer
5. Fragment Shader
6. Per-Fragment Tests
7. Blend
8. Framebuffer

Vertex Shader

- Transform incoming vertex position from *model* to *clip* coordinates
- Perform additional per-vertex computations; modify, add, or remove attributes passed down the pipeline
- Formerly called the *Transform and Lighting* (T&L) stage. Why?

\[
P_{\text{clip}} = (M_{\text{model-view-projection}})(P_{\text{model}})
\]
Transforms in the Pipeline
Vertex Shader

- Example: Textures can provide height maps for displacement mapping

Images from [http://developer.nvidia.com/content/vertex-texture-fetch](http://developer.nvidia.com/content/vertex-texture-fetch)
Rasterization

• Determine what pixels a primitive overlaps

• How would you implement this?

• What about aliasing?

• What happens to non-position vertex attributes during rasterization?

• What is the triangle-to-fragment ratio?
Fragment Shader

- Shades the fragment by simulating the interaction of light and material
- Loosely, the combination of a fragment shader and its uniform inputs is a *material*
- Also called a *Pixel Shader* (D3D)

What exactly is the fragment input?
What are examples of useful uniforms? Useful textures?
float diffuse = max(dot(N, L), 0.0);
Fragment Shader

- Another example: Texture Mapping
Fragment Shader

- Lighting and texture mapping
Fragment Shader

- Another example: Bump mapping
Fragment Shader

- Another example: Bump mapping
Depth Test

- Finds visible surfaces
- Once called “ridiculously expensive”
- Also called the *z-test*
  
  - Does it need to be after fragment shading?

Diagram:

- Vertex Assembly
- Vertex Shader
- Primitive Assembly
- Rasterizer
- Fragment Shader
- Per-Fragment Tests
- Blend
- Framebuffer

- Scissor Test
- Stencil Test
- Depth Test
Depth Test

Image from http://www.virtualglobebook.com/
Depth Test

Image from http://www.virtualglobebook.com/
Part II: GPU design
Early 90s – Pre GPU

Wolfenstein 3D, 1992

- Interactive software rendering (no GPUs yet)

Doom I, 1993

NOTE: SGI was building interactive rendering supercomputers, but this was beginning of interactive 3D graphics on PC
AMD Toyshop Demo

http://www.youtube.com/watch?v=LtxvpS5AYHQ
AMD Leo Demo

http://www.youtube.com/watch?v=zYweEn6DFcU
GPU Compute + Rendering

http://www.nvidia.com/object/GTX_400_games_demos.html
CPU and GPU Trends

Chart from: http://proteneer.com/blog/?p=263
Why GPUs?

- Exploit Parallelism
  - CPU and GPU executing in parallel
  - Data-parallel
  - Pipeline parallel

- Hardware: texture filtering, rasterization, MAD, sqrt, etc.
NVIDIA GeForce 256 (1999)

In hardware:
- Fixed-function vertex shading (T&L)
- Multi-texturing: bump maps, light maps, etc.
- 10 million polygons per second
- Direct3D 7
- AGP bus

NVIDIA GeForce 8 (2006)

- Ground-up GPU redesign
- Geometry Shaders
- Transform-feedback
- OpenGL 3 / Direct3D 10
- Unified shader processors
- Support for GPU Compute
Why Unify Shader Processors?
Evolution of the Programmable Graphics Pipeline
Part III: WebGL
WebGL

- The web has text, images, and video
  - What is the next media-type?
- We want to support
  - Windows, Linux, Mac
  - Desktop and mobile
Bring 3D to the Masses

- Put it in on a webpage
  - Does not require a plugin or install
- Make it run on most GPUs
WebGL

- OpenGL ES 2.0 for JavaScript
  - Seriously, JavaScript


**WebGL Performance**

<table>
<thead>
<tr>
<th></th>
<th>32x32</th>
<th>64x64</th>
<th>128x128</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>C++</strong></td>
<td>1.9 ms</td>
<td>6.25 ms</td>
<td>58.82 ms</td>
</tr>
<tr>
<td><strong>Chrome 18</strong></td>
<td>27.77 ms</td>
<td>111.11 ms</td>
<td>454.54 ms</td>
</tr>
<tr>
<td><strong>x slowdown</strong></td>
<td>14.62</td>
<td>17.78</td>
<td>7.73</td>
</tr>
</tbody>
</table>

**CPU-intensive**

<table>
<thead>
<tr>
<th></th>
<th>32x32</th>
<th>64x64</th>
<th>128x128</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>C++</strong></td>
<td>3.33 ms</td>
<td>9.43 ms</td>
<td>37.03 ms</td>
</tr>
<tr>
<td><strong>Chrome 18</strong></td>
<td>12.82 ms</td>
<td>22.72 ms</td>
<td>41.66 ms</td>
</tr>
<tr>
<td><strong>x slowdown</strong></td>
<td>3.85</td>
<td>2.41</td>
<td>1.13</td>
</tr>
</tbody>
</table>

**GPU-intensive (256 draws per frame)**
High-Profile WebGL Uses
More Cesium WebGL Apps
WebGL in Teaching

INTERACTIVE COMPUTER GRAPHICS
A Top-Down Approach with WebGL
7TH EDITION

Edward Angel
Dave Shreiner

Solving Transparent Z-Buffering, Part 2

SIGGRAPH 2014
Penn Student Work
WebGL Globe

Rohith Chandran
Ray Marching Distance Fields

Nop Jiarathanakul
Procedural Infinite City

// building off paulo falcao's raymarch framework -alice (http://y-alice.blogspot.com/)
// added random building size -h3r3 ;)
float CL = 5
precision highp float;
def

uniform vec2 resolution;
uniform float time;
uniform vec2 mouse;
// Simple raymarching sandbox with camera

// Raymarching Distance Fields
// About http://www.iquilezles.org/www/articles/raymarchingdf/raymarchingdf.htm
// Also known as Sphere Tracing
// Original seen here: http://twitter.com/#!/paulofalcao/statuses/134807547860353024

// Declare functions
vec2 ObjUnion(in vec2 d1, in vec2 d2);
vec2 floorPlane(in vec3 p);
vec2 color_checkers(in vec3 p);
vec2 roundBox(in vec3 p);
vec2 sBox(vec3 p, vec3 b);
vec3 color_white(in vec3 p);
vec2 distanceField(in vec3 p);
vec2 simpleBuilding (vec3 p, vec3 b);
vec4 applyRay (in vec4 currColor, in vec3 ray);

Alice Yang
WebGL Hot-Air Balloons

Yuanhui Chen
WebGL Deferred Shading

Hao Wu and Guanyu He
WebGL Deferred Shading

Sijie Tian and Yuqin Shao
WebGL Tools – Web Apps
WebGL Tools – Web Apps
WebGL Tools - Firefox
WebGL Tools - Chrome