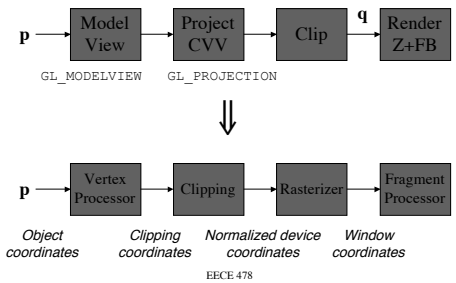
EECE 478

GPU Programming

Shader Languages

- NVIDIA Cg
- OpenGL Shader Language (GLSL)

Pipeline (Revisited)



Vertex Processor

Input: Vertices
- Position, color, normal, texture coords, material properties

Output: Vertices
- Position, color, normal, texture coords, material properties

Traditional pipeline:
- Modelview and projection transforms
- Lighting calculations

Vertex Processor (2)

Programmable Vertex Processor:
- Lighting: Color, material, lights ⇒ Color
- Programmed lighting:
  - Color can be calculated using different program
- Programmed attributes:
  - Any of vertex-bound attributes can be calculated or modified

Vertex Programs: GLSL

```
/* Pass-through shader */

void main(void)
{
    gl_Position =
      gl_ProjectionMatrix *
      gl_ModelViewMatrix *
      gl_Vertex;
}
```

```
/* Simple shader (red) */

const vec4 red =
    vec4(1.0, 0.0, 0.0, 1.0);

void main(void)
{
    gl_Position =
      gl_ProjectionMatrix *
      gl_ModelViewMatrix *
      gl_Vertex;
    gl_FrontColor = red;
}
```

Vertex Shaders: Basics

- Standard variables for input/output
  - gl_Position
  - gl_Color, gl_FrontColor
  - gl_Normal
  - gl_MultiTexCoord*X,* gl_TexCoord
- Program operates on *every* vertex in pipeline
- Values assigned are bound to vertex

Vertex Shaders: Variables

Data types:
- vec2, vec3, vec4: Indexed by [] or .x, .y, .z
- mat2, mat3, mat4

Variable kinds:
- *const* = cannot be modified
- *attribute* = once per-vertex (includes standard bound values: gl_Color, gl_Position, etc.)
- *uniform* = bound in application program, but constant within primitive
- *varying* = per-vertex and set in vertex program (used to pass info to fragment shader)

Vertex Shaders: Operators

Standard arithmetic:
- Assignment
- Add/subtract (vector, matrix or const)
- Multiplication (m*v, m*m, c*v, c*m, c*c)

Functions:
- dot(), length(), distance()
- sin(), cos(), asin(), …
- pow(), log2(), abs(), sqrt(), max(), min(), …

Swizzling:

a.xyz = b.yxz *or* a.xyz = b.xyx

Fragment Shaders

- Operate per-*pixel*
  - Pixel-by-pixel representation of triangle
  - Vertex values interpolated from corners
- Output is assignment of buffer values:
  - gl_FragColor
  - gl_FragDepth
  - Other values can be passed from Vertex Shader via 'var' variables
  - Textures accessed via 'sampler' variables

Other Resources

- Opntag:
  - http://opntag.net/user/leei/memos/tag/478,glsl
- NeHe Tutorial #21
  - http://nehe.gamedev.net/data/articles/article.asp?article=21
- Books on reserve (ICICS Reading Room)
  - Game Programming Gems
  - GPU Programming Gems