Introduction to CIS 536/636

Step 1: Declare shader programs and shader objects

Step 2: Load, add and compile/link programs in

Loading Programs without

Fragment Shader

```
void main() {
    gl_FragColor = vec4(0.0, 1.0, 0.0, 1.0);
}
```

```
void main(void)
```

```
vec4 a = gl_Vertex;
```

```
{ void main(void) {
```

```
Scaled vertex is transformed with concatenated modelview and projection matrix.

Incoming x and y components are scaled with a factor 0.5
```

```
A: Makes everything green!
```

```
Q: What does this do?
```

```
{ void main(void)
```

```
{ void main(void)
```

```
void main(void)
```

```
myFragmentShader->compile();
```

```
myVertexShader->load("simple.vert");
```

```
myShader->end();
```

```
myShader->begin();
```

```
myShader->addShader(myFragmentShader);
```

Review [1]:
Simple OGLSL Vertex & Pixel Shaders

Vertex Shader

```
int main(void)
```

```
myVertexShader->compile();
```

```
myFragmentShader->load("simple.frag");
```

Review [2]:
OGLSL Loading, Compiling, Linking

Loading Programs without

Review [3]:
OGLSL Shader Application

Loading Programs without

Instructor home page: http://www.cis.ksu.edu/~bhsu

List of videos (trailers, shorts, etc.)


http://youtu.be/b_UqzLBFz4Y

http://bit.ly/gDWqUb

etc.

etc.

etc.


etc.


Public mirror web site: http://www.kddresearch.org/Courses/CIS636

Wikipedia, Scene Graph

https://en.wikipedia.org/wiki/Scene_graph

http://www.newtek.com/lightwave/

Adobe Maya: http://students.autodesk.com


Defining TVertex data structure: position/normal/texture tuple:

```
struct TVertex {
    D3DXVECTOR3 position;
    D3DXVECTOR3 Normal;
    D3DXVECTOR3 Tex;
};
```

Vertex declaration to describe this structure:

```
const D3DVERTEXELEMENT9 dec[4] = {
    {0, 0,  D3DDECLTYPE_FLOAT3, D3DDECLMETHOD_DEFAULT, D3DDECLUSAGE_POSITION,0},
    {0, 12, D3DDECLTYPE_FLOAT3, D3DDECLMETHOD_DEFAULT, D3DDECLUSAGE_NORMAL, 0},
    {0, 24, D3DDECLTYPE_FLOAT2, D3DDECLMETHOD_DEFAULT, D3DDECLUSAGE_TEXCOORD,0},
    D3DDECL_END()
};
```

Each line corresponds to one of the elements in TVertex. The data in each line is:

- WORD Stream;
- WORD Offset;
- BYTE Type;
- BYTE Method;
- BYTE Usage;
- BYTE UsageIndex

We need to tell Direct3D about our vertex declaration using the following call:

```
IDirect3DVertexDeclaration9 m_vertexDeclaration;
gDevice->CreateVertexDeclaration(dec,&m_vertexDeclaration);
```

To render:

```
gDevice->SetStreamSource( 0, m_vb,0, sizeof( TVertex));
gDevice->SetVertexDeclaration(m_vertexDeclaration);
```

Scene Graph: General Data Structure used in CG
- Used to: compute visibility, set up rendering pipeline
- Nodes
  - Leaves: primitive components
  - Interior: assembly operations, modelview transformations
- Root(s): scene or major objects

Scene Graph Traversal: Initial Step – Drives Rendering

Aesthetics: Non-Photorealistic Shading, Aliasing
- Non-Photorealistic Rendering: Aimed at Achieving Natural Aesthetic
  - Cartoon shaders: use sharp gradient (thresholded)
  - Pencil shaders: blurring, stippling
- CGA and Realism
  - Term from signal processing
  - Two sampled signals indistinguishable from (aliases of) one another
  - Examples: jaggies, Moiré vibration
- Anti-aliasing: operations to prevent such effects
- Temporal Aliasing
  - Similar effect in animation
  - Small artifact can be much more jarring!
  - Example: think of flecks in traditional film reels

Next Time: Lab 3 OpenGL Shading & Transparency
- Set Up Point Light Sources
- Set Up Materials, Turn Lights On
- Start Drawing (glBegin … glEnd)

Preview: Painter’s Algorithm

Preview: Z-buffering
Computing & Information Sciences  
Kansas State University  

CIS 536/636  
Introduction to Computer Graphics  

Lecture 15 of 41  

Trailers: Video Games  

- *Crysis 2* © 2011 Electronic Arts  
  [http://youtu.be/j4mOGhWSXYQ](http://youtu.be/j4mOGhWSXYQ)  
- *Starcraft II: Wings of Liberty* © 2010 Blizzard  
- *Rage & id Tech 5* © 2011 id Software  
- *Unreal Engine 3* © 2004-2011 Epic/Valve  
  [http://youtu.be/MGf0oGGGQqQ](http://youtu.be/MGf0oGGGQqQ)  

Videos: CG Feature Films & Shorts  

- *Wall-E* © 2008 Disney/Pixar  
- *Kung-Fu Panda* © 2008 DreamWorks Animation SKG  
- *Shrek Forever After* © 2010 DreamWorks Animation SKG  
  [http://youtu.be/u7__TG7swg0](http://youtu.be/u7__TG7swg0)  
- *Happy Feet* © 2006 Warner Brothers  
- *Luxo Jr.* © 1986 Pixar Animation Studios  
- *Monsters Inc.* © 2001 Disney/Pixar  
  [http://youtu.be/cvOQeozL4S0](http://youtu.be/cvOQeozL4S0)  

Summary  

- Last Time: Shader Languages – OGLSL & Direct3D  
  - OpenGL Shading Language (OGLSL or GLSL) – main topic  
  - High-Level Shading Language (HLSL) & Direct3D  
- Today: Scene Graphs; Computer-Generated Animation Demos, Videos  
  - Scene graphs and state – main topic  
  - State of CGA: videos  
- Issues  
  - Photorealism and non-photorealistic rendering (NPR)  
  - Making most of hardware  
  - Role of animators (see CNBC’s Pixar Story, [http://bit.ly/gShkXL](http://bit.ly/gShkXL))  
  - Techniques showcased  
  - Multipass texturing  
  - Alpha compositing/blending  
  - Portals and binary space partitioning  
- Demos to download: Maya, LightWave  

Terminology  

- **Scene Graph**: General Data Structure used in CG  
  - Used to: compute visibility, set up rendering pipeline  
  - Actual graph: general graph, forest, or rooted tree  
- **Scene Graph Traversal**: Initial Step – Drives Rendering  
- Features of Scene Graphs  
  - **Spatial partitioning**: e.g. using bounding volume hierarchies  
  - **Leaves**: primitive components  
  - **Interior nodes**: assembly operations, modelview transformations  
  - **Roots**: scene or major objects  
- **Non-Photorealistic Rendering**: Aimed at Achieving Natural Aesthetic  
  - Cartoon shaders; use sharp gradient (thresholded)  
  - **Pencil shaders**: blurring, stippling  
- **CGA and Realism**  
  - Aliasing & anti-aliasing  
  - Temporal aliasing & temporal anti-aliasing