

Evolution of Graphics API

Sujal Bista
CMSC 828V



Tomb Raider I



Tomb Raider II



Tomb Raider III



Tomb Raider TLR



Tomb Raider C



Tomb Raider AOD



Tomb Raider L



Tomb Raider A



TR Underworld

Outline

- Introduction to OpenGL
- Evolution of OpenGL
- Recent advances in OpenGL API ver 3.2
- New features of DirectX 11
- Conclusion

OpenGL

- Open Graphics Library
- Industry standard high performance graphics API
- Developed by Silicon Graphics Inc. in 1992
- Widely used in CAD, scientific visualization, games
- Cross-platform



OpenGL 1.x

Features

- Fixed function (no shaders)
- OpenGL state machine
- Draw basic primitives (lines, triangles ...)

OpenGL 2.x

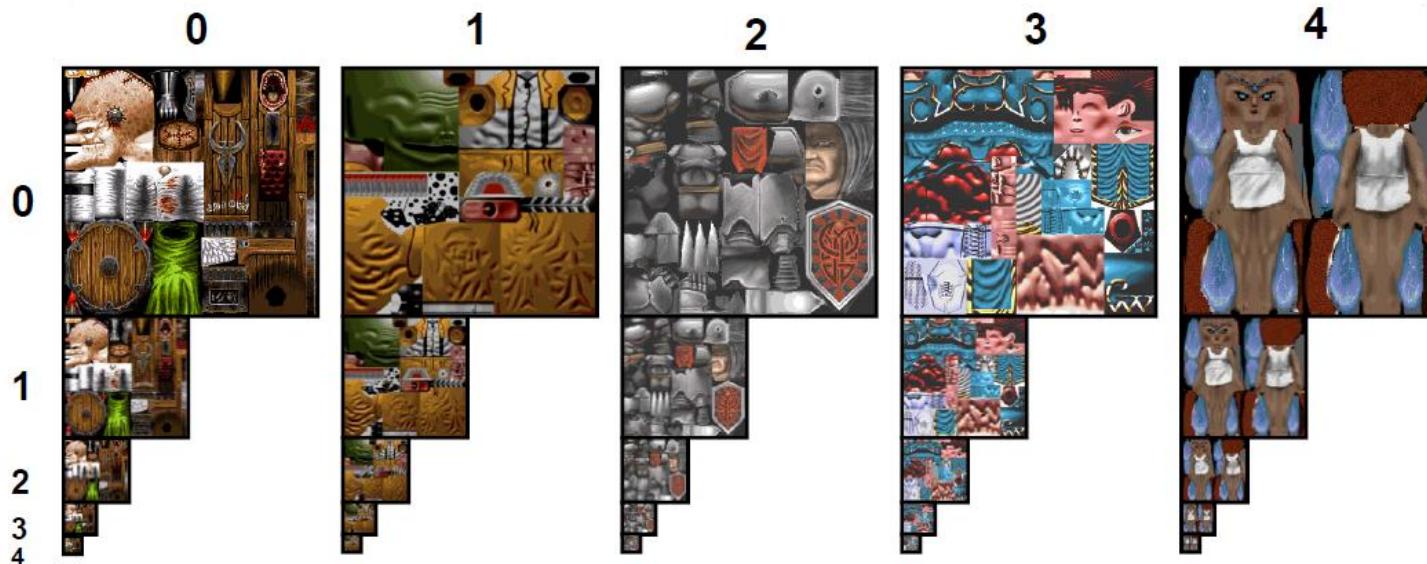
Features

- Shading language (GLSL)
- Multiple rendering targets
- Non-power of two textures
- Automatic mipmap generation
- Texture access in vertex shaders

OpenGL 3.x

Features

- Geometry shader
- Vertex Array Objects(collection of state)
- Texture Arrays



OpenGL 3.x

Features continued..

- OpenCL helper functions (CopyBuffers)
- Costume texture filter
- Increased multi-sample rendering quality
 - fragment shader per sample instead of per pixel
- Depth clamp
 - Near/Far plane Clipping

OpenGL 3.x

Need for Change

- 15+ old API difficult to maintain
- Some functions do not scale well
 - Bindings required for every little modifications
- Increase similarity of Direct3D
 - Developers need cross API and platforms

OpenGL 3.x

Deprecation Model

- Allows API to mature without breaking backward compatibility
- Deprecation \approx feature marked for removal
- Old, slow and redundant functions are removed
- Streamline the API

OpenGL 3.x

Deprecation Model in Action

- OpenGL 3.0 marked some features as deprecated
- OpenGL 3.1 removed these deprecated features
- To support market need ARB_compatibility extension is created to provide support for deprecated feature. (Optional for vendors to implement)

OpenGL 3.x

OpenGL 3.2 ARB created two profiles

- Core profile : Supports streamlined features. No deprecated features.
- Compatibility profile : Supports all the features.
- Nvidia has no interest in removing any features
- Suggests using compatibility profile for backward compatibility

OpenGL 3.x

Life of OpenGL function/features

- Extension path



- Deprecation path



OpenGL 3.x

Deprecated Functions/Features

- Fixed Function vertex and fragment processing (eg. `glMultMatrix`, `glRotate` etc.)
- Color index mode
- GLSL 1.10 and 1.20
- Alpha test
- Accumulation buffers

OpenGL 3.x

Deprecated Functions/Features Continued...

- Intermediate mode (tortures the video card)
 - glVertex, glTexCoord, glNormal, glColor
 - Begin/End primitive specifications
 - “Send more data once instead of less data a lot of time”
 - Use vertex arrays and array drawing commands
- Wide lines (>1 pixel) and line stipple
- Polygon mode (Front/Back)
- Texture borders

OpenGL 3.x

Deprecated Functions/Features Continued...

- Automatic mipmap generation
- Selection and feedback mode (PushName, LoadName)
- Display lists
- Attribute stacks

OpenGL 3.2 Example I

//SCENE INITIALIZATION

// Two VAOs allocation

```
glGenVertexArrays(2, &m_vaoID[0]);
```

// First VAO setup

```
glBindVertexArray(m_vaoID[0]);
glGenBuffers(2, m_vboID);
glBindBuffer(GL_ARRAY_BUFFER, m_vboID[0]);
glBufferData(GL_ARRAY_BUFFER, 9*sizeof(GLfloat), vert,
GL_STATIC_DRAW);
glVertexAttribPointer((GLuint)0, 3, GL_FLOAT, GL_FALSE,
0, 0);
glEnableVertexAttribArray(0);
glBindBuffer(GL_ARRAY_BUFFER, m_vboID[1]);
glBufferData(GL_ARRAY_BUFFER, 9*sizeof(GLfloat), col,
GL_STATIC_DRAW);
glVertexAttribPointer((GLuint)1, 3, GL_FLOAT, GL_FALSE,
0, 0);
glEnableVertexAttribArray(1);
```

// Second VAO setup

```
glBindVertexArray(m_vaoID[1]);
glGenBuffers(1, &m_vboID[2]);
glBindBuffer(GL_ARRAY_BUFFER, m_vboID[2]);
glBufferData(GL_ARRAY_BUFFER, 9*sizeof(GLfloat), vert2,
GL_STATIC_DRAW);
glVertexAttribPointer((GLuint)0, 3, GL_FLOAT, GL_FALSE,
0, 0);
```

//SHADER SETUP

```
glBindAttribLocation(shaderID,0,"in_Position");
glBindAttribLocation(shaderID,1,"in_Color");
```

//SIMPLE VERTEX SHADER

```
in vec3 in_Position;
in vec3 in_Color;
out vec3 ex_Color;
void main(void)
{
    gl_Position = vec4(in_Position, 1.0);
    ex_Color = in_Color;
}
```

//RENDER

```
// select first VAO
glBindVertexArray(m_vaoID[0]);
// draw first object
glDrawArrays(GL_TRIANGLES, 0, 3);
// select second VAO
glBindVertexArray(m_vaoID[1]);
// set constant color attribute
glVertexAttrib3f((GLuint)1, 1.0, 0.0, 0.0);
// draw second object
glDrawArrays(GL_TRIANGLES, 0, 3);
```

OpenGL 3.2 Example II

//SIMPLE DRAW CALL

//bind attrib position and color

```
glBindAttribLocation(shaderprogram, 0, "in_Position");  
glBindAttribLocation(shaderprogram, 1, "in_Color");
```

//pass modelview projection matrix

```
glUniformMatrix4fv(glGetUniformLocation(shaderprogram,  
"mvpmatrix"), 1, GL_FALSE, modelViewProjectionmatrix);
```

//draw call

```
glDrawArrays(GL_TRIANGLES, 0, 12);
```

//VERTEX SHADER

```
#version 150
```

```
in vec3 in_Position;
```

```
in vec3 in_Color;
```

```
//modelview projection
```

```
uniform mat4 mvpmatrix;
```

```
out vec3 ex_Color;
```

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

```
// Multiply the mvp matrix by the vertex to obtain our final  
vertex position
```

```
    gl_Position = mvpmatrix * vec4(in_Position, 1.0);
```

```
    ex_Color = in_Color;
```

```
}
```

//FRAGMENT SHADER

```
#version 150
```

```
precision highp float;
```

```
in vec3 ex_Color;
```

```
out vec4 gl_FragColor;
```

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

```
    gl_FragColor = vec4(ex_Color,1.0);
```

```
}
```

Interesting New Extensions

Direct state access

- Edit object by name
- Similar to Direct3D API
- Still Vendor specific Nvidia with collaboration with ID, S3, Blizzard etc.

Interesting New Extensions

Direct state access

Command	Old	New
Uniformly scaling modelview matrix by 2	<pre>GLenum savedMatrixMode; glGetIntegerv(GL_MATRIX_MODE, &savedMatrixMode); glMatrixMode(GL_MODELVIEW); glScaleMatrixf(2,2,2); glMatrixMode(savedMatrixMode);</pre>	<pre>glMatrixScalefEXT(GL_MODELVIEW, 2,2,2);</pre>
Binding textures to texture units	<pre>glActiveTexture(GL_TEXTURE0); glBindTexture(GL_TEXTURE_2D, texobj);</pre>	<pre>glBindMultiTexture(GL_TEXTURE5, GL_TEXTURE_2D, texobj);</pre>
Updating a uniform or program parameter	<pre>glBindProgramARB(GL_VERTEX_PROGRAM, vp); glProgramLocalParameter4fARB(index, x,y,z,w); glUseProgram(glslprog); glUniform4f(location, x,y,z,w);</pre>	<pre>glNamedProgramLocalParameter4fEXT(vp, index, x,y,z,w); glProgramUniform4fEXT(glslprog, location, x,y,z,w);</pre>

Interesting New Extensions

Bindless Graphics

- Binding different buffers are very costly
- Use GPU address rather than by name
- Driver no longer has to fetch GPU address from system memory
- Measurements have shown that bindless graphics can result in more than 7x speedup (NVIDIA)

DirectX11

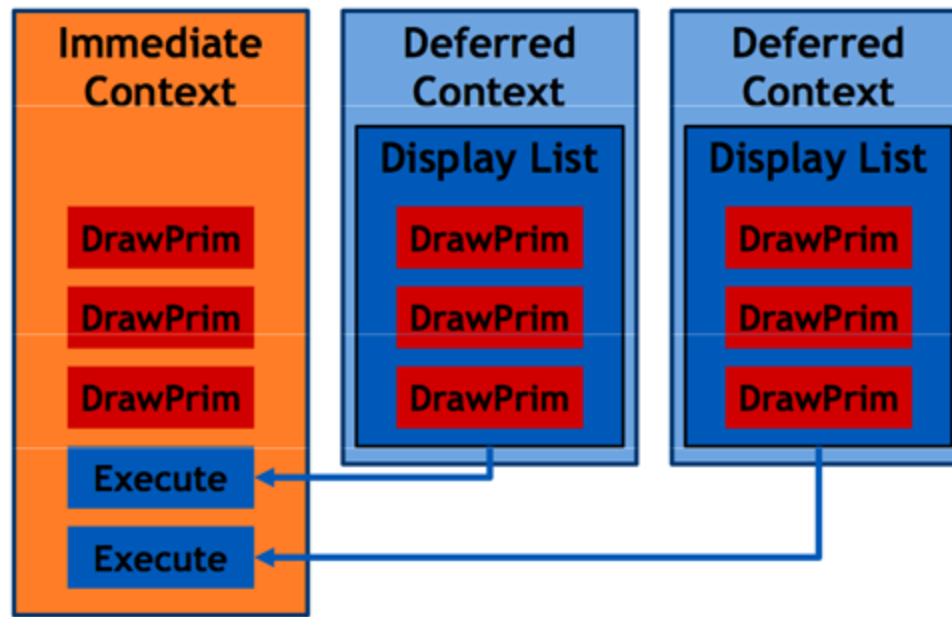
New Features

- Hull shader – Takes patches and control point and outputs data on how to configure tessellator
- Tessellator – Takes coarse shapes and breaks them into small parts based on the input from hull shader
 - Fixed function but high speed
 - Geometry shader vs Tessellation steps(flexibility vs speed)
- Domain shader – Takes generated points from tessellator and manipulates them to form appropriate geometry. (Shifts, displacements, etc.)

DirectX11

New Features Continued...

- Compute shader
- Multi-thread API using deferred contexts



Conclusion

- Fundamental changes
- Optimized and stronger API
- Intel's Larrabee might add features/changes
- Higher initial barrier for the starters

Sources

- http://http.download.nvidia.com/developer/Papers/2005/OpenGL_2.0/NVIDIA_OpenGL_2.0_Support.pdf
- <http://ubuntuforums.org/archive/index.php/t-816081.html>
- http://developer.download.nvidia.com/presentations/2008/NVISION/NVISION08_opengl_mjk_web.pdf
- http://www.khronos.org/developers/library/2009_siggraph_bof_opengl/OpenGL-BOF-OpenGL-3.2-Overview-Siggraph-Aug09.pdf
- <http://www.gaming-zone.net/wp-content/uploads/2008/08/larahist.gif>
- <http://www.opengl.org/registry/doc/glspec32.compatibility.20090803.pdf>
- <http://www.anandtech.com/printarticle.aspx?i=3507>
- <http://www.opengl.org/registry/doc/glspec32.core.20090803.pdf>
- http://sites.google.com/site/opengl_tutorials_by_aks/introduction-to-opengl-3-1---tutorial-01
- http://www.slideshare.net/Mark_Kilgard/opengl-32-and-more
- [http://www.opengl.org/wiki/Tutorial3:_Rendering_3D_Objects_\(C_/SDL\)](http://www.opengl.org/wiki/Tutorial3:_Rendering_3D_Objects_(C_/SDL))
- http://developer.download.nvidia.com/opengl/tutorials/bindless_graphics.pdf

Questions??



Metal Gear Solid 4



Final Fantasy XIII