

Modern OpenGL

Lorenz Diener

Entropia EV

19. Juni 2010

Introduction



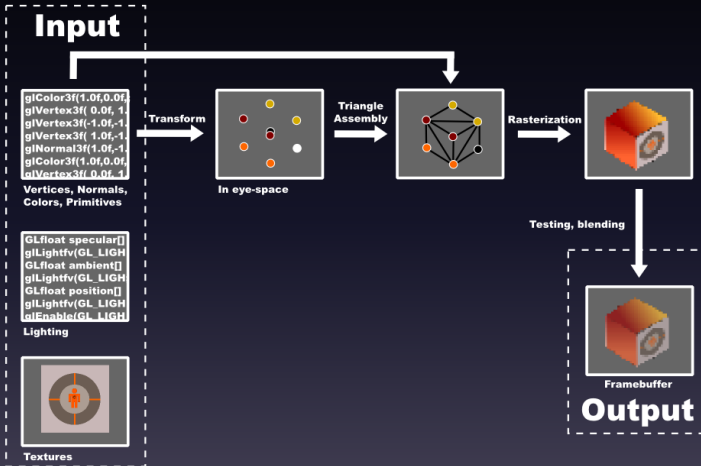
Introduction

- Current version: 4.0
- Documentation not beginner-friendly
- Hard to figure out what is state of the art

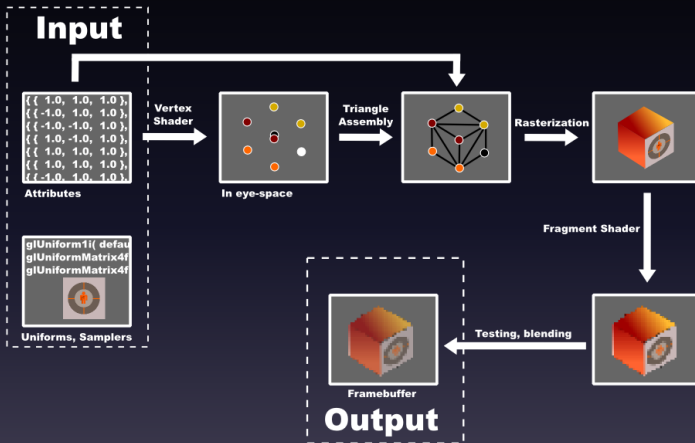
Introduction

- Today: Introduction
- OpenGL, modern
- Also: WebGL, OpenGL ES 2.0
- Demo application (GLUT, GLEW)

Fixed-Function-Pipeline



Programmable Pipeline



Opening a window

```
glutInit( &argc, argv );  
glutInitDisplayMode( GLUT_RGB | GLUT_DOUBLE | GLUT_DEPTH );  
  
char modeString[255] = "";  
sprintf( modeString, "%dx%d:24",  
        WINDOW_WIDTH, WINDOW_HEIGHT );  
glutGameModeString( modeString );  
glutEnterGameMode();  
  
glewInit();  
  
glutDisplayFunc( draw );  
glutTimerFunc( 15, update, 0 );
```

OpenGL-Setup

```
glEnable( GL_DEPTH_TEST );  
glClearDepth( 1.0f );  
glDepthFunc( GL_LEQUAL );
```

```
glClearColor( 0.0f, 0.0f, 0.0f, 1.0f );
```

```
glEnable( GL_CULL_FACE );  
glCullFace( GL_BACK );
```

Result



Buffer-Objects

```
vertex vertices[] = {  
  { { 1.0, 1.0, 1.0 }, 1.0,  
    { 0.0, 0.0, 1.0 },  
    0.0, 0.0 },
```

```
  { { -1.0, 1.0, 1.0 }, 1.0,  
    { 0.0, 0.0, 1.0 },  
    1.0, 0.0 }
```

```
[...]
```

Buffer-Objects

```
GLuint bo;  
glGenBuffers( 1, &bo );  
glBindBuffer( GL_ARRAY_BUFFER, bo );  
glBufferData(  
    GL_ARRAY_BUFFER,  
    vertices ,  
    sizeof( vertices ),  
    GL_STATIC_DRAW  
);
```

Textures

- Loading: Library, homegrown loader
- glGenTextures
- glBindTexture
- glTexParameteri - Interpolation!
- glTexImage2D

Result



Shader

- Vertex Shader, Fragment Shader, (Geometry Shader)
- Massively parallel on the GPU
- GLSL

Vertex-Shader

- Input: Attributes, uniforms
- Output: `gl_Position`, varying-variablen
- `gl_Position = viewVertex * modelview * projection;`

Modelview-Matrix

- Linear transforms: Rotation, scaling, shearing
- "Same values, different axes"
- http://en.wikipedia.org/wiki/Linear_map

Homogenous coordinates

- Translation is not linear!
- Idea: Add another row to the matrix, a 1.0 to every vector.
- (Will be useful again later)

Projection-Matrix

- Projection space
- Cube from -1 bis 1, everything outside: Clipping
- Front face = screen
- Z-axis pointing into screen
- Transform world-space into this - perspective projection.

Fragment-Shader

- Input: Varyings vom Vertex-Shader
- Output: `gl_FragColor`
- `gl_FragColor = vec4(1.0, 0.0, 1.0, 1.0);`

Loadin shaders

```
GLchar *shaderSrc = loadFile( file );
GLsizei length = strlen ( shaderSrc );
GLuint shader = glCreateShader( type );
glShaderSource(
    shader,
    1,
    (const GLchar**)&shaderSrc,
    &length
);
free( shaderSrc );
glCompileShader( shader );
```

Linking shaders

```
GLuint vertexShader =
    loadShader( GL_VERTEX_SHADER, "simple.vert" );
GLuint fragmentShader =
    loadShader( GL_FRAGMENT_SHADER, "simple.frag" );
shaderProgram =
    makeShaderProgram( vertexShader, fragmentShader );
vertexNormal =
    glGetUniformLocation( shaderProgram, "position" );
modelviewMatrix =
    glGetUniformLocation( shaderProgram, "modelview" );
projectionMatrix =
    glGetUniformLocation( shaderProgram, "projection" );
```

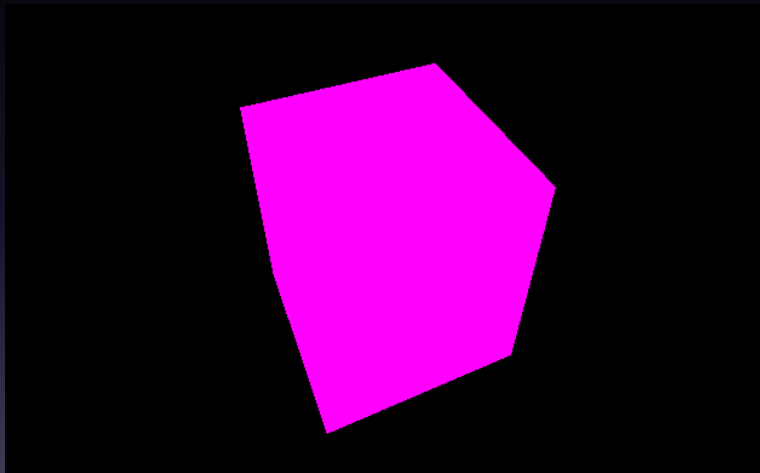
Rendering, Part 1

```
glUseProgram( shaderProgram );
glUniformMatrix4fv( modelviewMatrix, 1, GL_TRUE, mv );
glUniformMatrix4fv( projectionMatrix, 1, GL_TRUE, proj );
glBindBuffer( GL_ARRAY_BUFFER, vertexBuffer );
glVertexAttribPointer(
    vertexPosition,
    4,
    GL_FLOAT,
    GL_FALSE,
    sizeof(GLfloat) * 9,
    (void*)0
);
glEnableVertexAttribArray( vertexPosition );
```

Rendering, Part 2

```
glBindBuffer( GL_ELEMENT_ARRAY_BUFFER, elementBuffer );  
glDrawElements(  
    GL_QUADS,  
    24,  
    GL_UNSIGNED_SHORT,  
    (void*)0  
);  
  
glutSwapBuffers();
```

Result

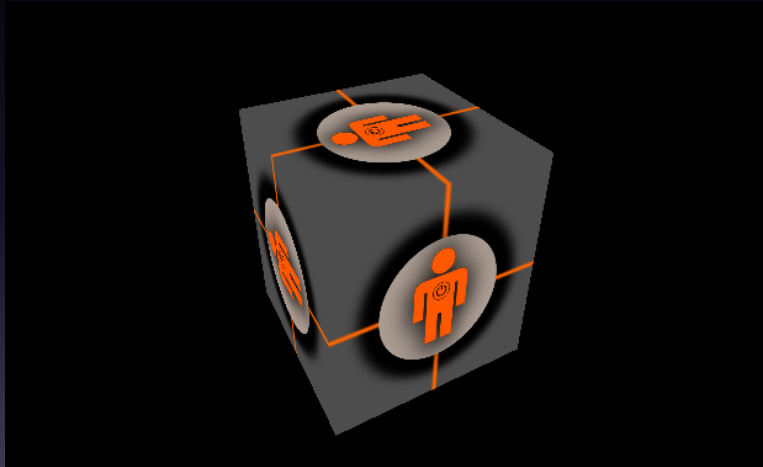


Texturing

- Bind to texture-unit
- Pass texture coordinates as attribute
- Per varying to fragment shader - interpolation!
- In fragment shader: `texture2D(texture, texcoord);`

```
glActiveTexture( GL_TEXTURE0 );  
glBindTexture( GL_TEXTURE_2D, defaultTextureData );  
glUniform1i( defaultTexture, 0 );
```

Result



Light

- Simple: Phong-Shading
- Ambient, Diffuse, Specular
- http://en.wikipedia.org/wiki/Phong_shading
- Normals (parameter) - also need transforming!

```
viewVertex = vertex * modelview;  
eye = normalize( viewVertex.xyz );  
light = normalize( lightPos - viewVertex.xyz );  
normal = normalize( (vec4(vnormal, 0.0) * normalview ).xyz );
```

Light, fragment-shader

```
float lambertTerm = dot( normal, light );
if( lambertTerm > 0.0 ) {
    color += lightDiffuse * materialColor * lambertTerm;
    vec3 specDir = normalize( reflect( light, -normal ) );
    float specular = pow(
max( 0.0, dot(specDir, eye) ),
materialShinyness
);
    color += lightSpecular *
materialSpecular *
specular;
}
```

Result



Code

Questions?