CPSC 314
Computer Graphics

Dinesh K. Pai

Nuts and bolts of
OpenGL programming
Announcements

- Assignment 1 now out. Due Jan 24.
  - See <coursepage>/resources.html
  - README.txt has details on where to get required libraries, and instructions for Windows and Linux.
  - TA is working on instructions for the Mac. Input welcome.
- “Prerequisite letter”. Contact me if you received email about this.
- Today: Some practical aspects of programming with OpenGL and vertex shaders.
Introduction to Assignment 1

- Switch to demo
What you need to get started..

- GLUT and freeGLUT
- GLEW
- GLM
- GLSL
The good news

- Even though there are lots of details and options, a few useful things go a long way.
- After initial setup, most of your effort will be on translating graphics concepts into code.
- For Assignment 1, this is already setup for you. You mainly have to ensure you have the right libraries... and focus on the vertex shader.
GLalphabet soup

- GLUT and freeGLUT
  - Window system interface, animation timers
  - Simple, cross-platform. Alternative: GLFW
  - Event-driven interaction, register “callback”
  - Example: glutDisplayFunc(display)

- GLEW
- GLM
- GLSL
GLalphabet soup

- GLUT and freeGLUT
- GLEW
  - Magic to access OpenGL extensions. Just do it.
- GLM
- GLSL
GLalphabet soup

- GLUT and freeGLUT
- GLEW
- GLM
  - A math library that matches GLSL math functions (more about that later)
  - NOTE: we won’t use Cvec3, Matrix4, etc. from Text
  - GLM is more general
- GLSL
GLalphabet soup

- GLUT and freeGLUT
- GLEW
- GLM
- GLSL
  - OpenGL shading language
  - C-like, w. data types and functions useful for graphics
    - vec3, vec4, dvec4, mat4, sampler2D …
      (OpenGL data are floats unless qualified)
    - <matrix-vector multiplication>, smoothstep, reflect,…
  - Used for both vertex shaders and fragment shaders, with small differences
Pattern of an OpenGL program

```c
int main(int argc, char **argv) {
    initGlutState(argc, argv);
    glewInit(); // load the OpenGL extensions

    initGLState();
    initShaders();
    initBuffers();

    ...
    glutMainLoop();
    return 0;
}
```
Call back function “display"

- Registered with GLUT using glutDisplayFunc(display)

```c
static void display(void) {
    glUseProgram(h_program)
    glClear(GL_COLOR_BUFFER_BIT |
            GL_DEPTH_BUFFER_BIT);
    drawObj();
    glutSwapBuffers();
}
```
A closer look at GLSL shaders
Creating a Shader Program

Figure 2.1  Shader-compilation command sequence
Source: OpenGL programming guide, 8th edition
#version 130

uniform float uVertexScale;

in vec2 aPosition;
in vec3 aColor;
in vec2 aTexCoord0, aTexCoord1;

out vec3 vColor;
out vec2 vTexCoord0, vTexCoord1;

void main() {
    gl_Position = vec4(aPosition.x * uVertexScale, aPosition.y, 0,1);
    vColor = aColor;
    vTexCoord0 = aTexCoord0;
    vTexCoord1 = aTexCoord1;
}
#version 330
layout (location = 0) in vec3 Position;
layout (location = 1) in vec3 Normal;
struct MaterialInfo {
    vec3 Ka;
    vec3 Kd;
    vec3 Ks;
    float Shininess;
};
uniform MaterialInfo Material;
uniform mat4 MVP; // ModelViewProjection Matrix

void main()
{
    vec3 tnorm = normalize( NormalMatrix * Normal );
    ... 
    gl_Position = MVP * vec4(Position, 1.0);
}
Next class

- Wrap up OpenGL nuts and bolts
- Back to 3D Math for Graphics
  - Read rest of Chapter 2, Chapter 3 up to 3.5.