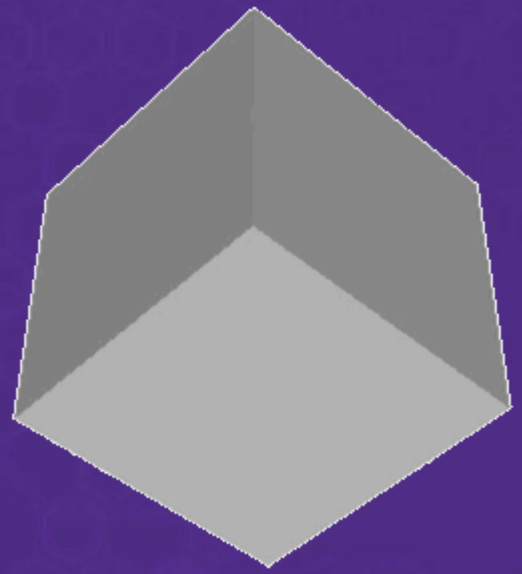
Developing Graphics Frameworks with Java and OpenGL



Part 03: Compiling GPU Programs

To Render a Point…

- Create a window that displays image data from GPU
- Write shader programs to draw a single yellow point
  - *Vertex shader*: computes vertex data (position)
  - *Fragment shader*: computes pixel appearance (color)
- Shader code must be sent to GPU, compiled, and linked into a program, before it can be used during the application main loop

OpenGL Shading Language

- Basic data types: **`bool`**, **`int`**, **`float`**, **`vecN`**, **`matN`** (N = 2,3,4)
- **`vec4 v`**: access elements as array **`v[0]`**, **`v[1]`**, **`v[2]`**, **`v[3]`**, or dot notation (**`v.x`**, **`v.y`**, **`v.z`**, **`v.w`** or **`v.r`**, **`v.g`**, **`v.b`**, **`v.a`**)
- Shaders must contain main functions in this format:

```
void main()
{
    // code statements here
}
```

Passing Data to Shader Programs

- data passed in and out of shaders by variables declared with *type qualifiers*: keywords that modify variable properties
- **`in`** indicates variable value supplied by previous pipeline stage
- **`out`** indicates variable value will be passed to next pipeline stage
- vertex shader:
  - **`in`** values supplied from a buffer
  - **`out`** values passed to fragment shader
- fragment shader:
  - **`in`** values supplied from vertex shader (interpolated by rasterizer)
  - **`out`** values stored in a buffer (color, depth, stencil)

The Simplest Vertex Shader

- Vertex Shader must set value: **`vec4 gl_Position`**

Shader code to render points in center of screen:

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

The Simplest Fragment Shader

- ~~Fragment Shader must set value: `vec4 gl_FragColor`~~
- Fragment Shader must declare and set value of pixel color

Shader code to render pixels in yellow:

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

OpenGL documentation

- need to use OpenGL functions to load and compile shader code, and link shaders into GPU programs

- official documentation: http://www.khronos.org/registry/OpenGL-Refpages/

`functionName``(` *`parameter1, parameter2, …`* `)`
Description of function and parameters.

GPU: Shaders

- **`glCreateShader`**(*shaderType*)
  Create a shader object to store the source code; returns a reference value.
  Type of shader (vertex or fragment) specified by *shaderType* parameter; value is a constant (GL_VERTEX_SHADER or GL_FRAGMENT_SHADER)

- **`glShaderSource`**(*shaderRef*, *shaderCode*)
  Stores source code in the string *shaderCode* in the shader referenced by *shaderRef*.

- **`glCompileShader`**(*shaderRef*)
  Compiles source code stored in the shader referenced by *shaderRef*.

GPU: Error Checking

- **`glGetShaderiv`**(*shaderRef* , *shaderInfo*)
  Returns information about shader referenced by *shaderRef.*
  Type of information specified by *shaderInfo* parameter;
  value is a constant (such as GL_SHADER_TYPE or GL_COMPILE_STATUS).

- **`glGetShaderInfoLog`**(*shaderRef*)
  Returns information about compilation (errors) of the shader *shaderRef*.

- **`glDeleteShader`**(*shaderRef*)
  Frees the memory and reference used by the shader *shaderRef.*

GPU: Programs

- **`glCreateProgram`**()
  Create a program object to store shaders; returns a reference value.

- **`glAttachShader`**(*programRef*, *shaderRef*)
  Attaches a shader referenced by *shaderRef* to the program referenced by *programRef*.

- **`glLinkProgram`**(*programRef*)
  Links the vertex and fragment shaders previously attached to *programRef*.
  Verifies in/out variables are declared consistently.

GPU: Error Checking

- **`glGetProgramiv`**(*programRef* , *programInfo*)
  Returns information about program referenced by *programRef.*
  Type of information specified by *programInfo* parameter;
  value is a constant (such as GL_LINK_STATUS)

- **`glGetProgramInfoLog`**(*programRef*)
  Returns information about linking (errors) of the program *programRef*.

- **`glDeleteProgram`**(*programRef*)
  Frees the memory and reference used by the program *programRef.*

Java: OpenGLUtils class

- `public class OpenGLUtils()`
- `public static String readFile(textFileName)`
- `public static int initShader(shaderCode, shaderType)`
- `public static int initProgram(vShaderCode, fShaderCode)`
- `public static int initFromFiles(vFileName, fFileName)`