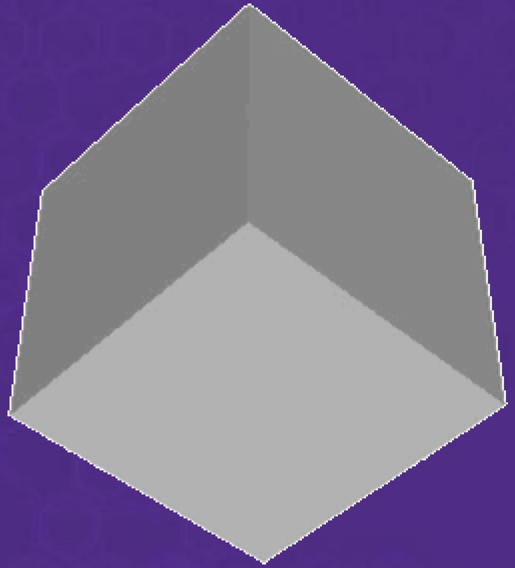
Developing Graphics Frameworks with Java and OpenGL

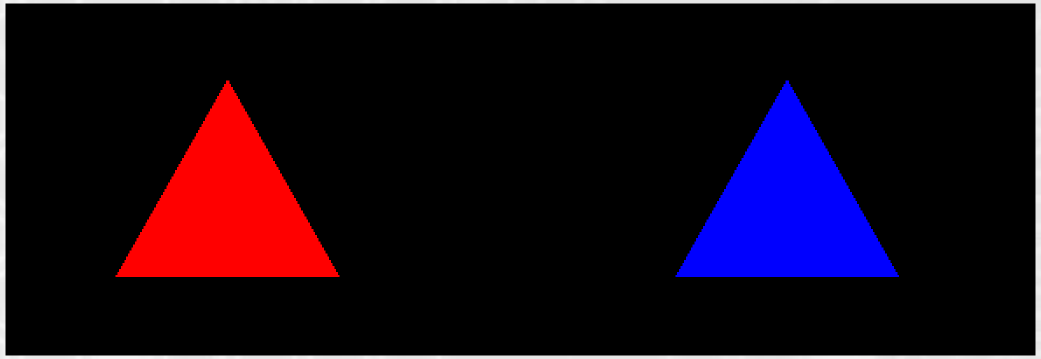


Part 09: Uniform Variables

Uniform Variables

- Recall: type qualifiers indicate source/destination of data
  - **`in`**: data received from previous stage in graphics pipeline
  - **`uniform`**: data received directly from CPU
- Used to send constant values to all vertices/fragments
  - apply same transformation to all vertices
  - apply same color to all fragments
- Bonus: Uniform values can be changed efficiently between (but not during) draw calls to create animated effects and interactive programs

Sample Application



- Could accomplish (inefficiently) with multiple vertex buffers (4)
- Instead: use one vertex buffer for vertex positions; write shader programs with uniform variables to adjust positions and apply different colors
  - Note: same uniform values apply to every vertex during draw call

Shader Code

```
----------------------------------------
in vec3 position;
uniform vec3 translation;
void main() {
    vec3 pos = position + translation;
    gl_Position = vec4(pos.x, pos.y, pos.z, 1.0);
}
----------------------------------------
uniform vec3 baseColor;
out vec4 fragColor;
void main() {
    fragColor = vec4(baseColor.r, baseColor.g, baseColor.b, 1);
}
----------------------------------------
```

OpenGL functions for Uniforms

- **`glGetUniformLocation`**(*programRef, variableName*)
  Returns reference to uniform variable named *variableName* in program referenced by *programRef*

- **`glUniform`**{ **`1|2|3|4`** }{ **`f|i`** }(*variableRef, value1, …*)
  Specify value of uniform variable *variableRef* in the currently bound program.
  - number in function name = number of values sent
  - letter (**`f`** or **`i`**) refers to data type (float or integer)
  - ex: **`glUniform1i`** , **`glUniform3f`** , etc.

Uniform class

- Similar to Attribute class, stores name of data type, the data, and reference to corresponding variable in a program
- Unlike Attribute class, data is re-uploaded before each render, and the data is not a float[]: stored as various Java objects with public access to change values later

- `public class Uniform<T>`
- `public Uniform(dataType, data)`
- `public void locateVariable(programRef, variableName)`
- `public void uploadData()`