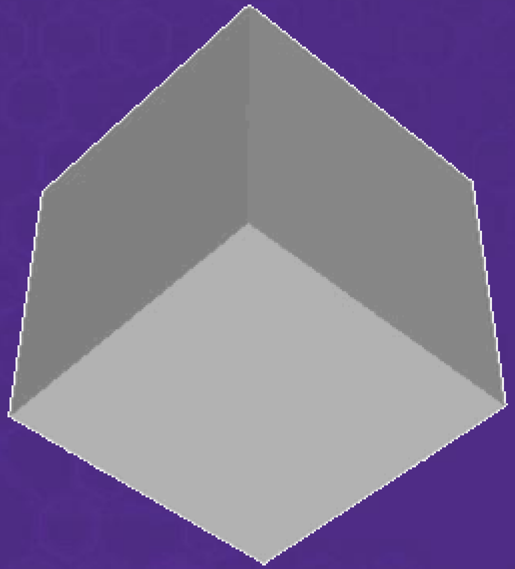
Developing Graphics Frameworks with Java and OpenGL



Part 04: Drawing a Point

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, linked
- Use GPU program during application main loop

Vertex Array Objects (VAOs)

- Vertex array objects used to manage vertex related data
  - which buffers associated to which shader in variables

- **`glGenVertexArrays(`** *vaoCount* **)**
  Returns a set of available VAO references (a total of *vaoCount* references)

- **`glBindVertexArray(`** *vaoRef* **)**
  Binds the VAO referenced by parameter *vaoRef;*
  any commands relating to VAOs use the currently bound VAO.
  Unbinds any VAO that was previously bound.

Using the GPU program

- **`glUseProgram(`** *programRef* **`)`**
  Specifies the GPU program to use during rendering
  (the program referenced by *programRef*)

- **`glDrawArrays(`** *drawMode* , *firstIndex* , *indexCount* **`)`**
  Draws geometric primitives (points, lines, or triangles) using the GPU program specified by **`glUseProgram`**.
  - Vertex shader uses data from arrays stored in vertex buffers, beginning at index *firstIndex*; total number of array elements specified by *indexCount.*
  - Type of geometric primitive specified by *drawMode*; value is an OpenGL constant such as GL_POINTS, GL_LINES, GL_LINE_LOOP, GL_TRIANGLES, etc.

Render Settings

- **`glPointSize(`** *size* **`)`**
  Specifies that points should be rendered with diameter (in pixels) equal to the integer parameter *size* (default value 1)