OpenGL 3 & 4

CSCI 4239/5239
Advanced Computer Graphics
Spring 2020

What is new in OpenGL 3&4

- Additional shaders
  - Geometry (OpenGL 3.2)
  - Tesselation (OpenGL 4.0)
  - Compute (OpenGL 4.3)
- New syntax for passing variables
  - “in” from previous stage
  - “out” to next stage
  - Deprecating most predefined variables
- Building objects from vertex arrays
- Deprecating OpenGL transformations

Deprecated Features

- glBegin() glEnd()
  - Use vertex buffer objects instead
- glTranslate() glRotate() glScale()
  - Use vmath or glm or roll your own
  - Qt povides QVector* and QMatrix*
- Display lists
- *Deprecated features remain available through the compatibility profile, but are not available in the core profile which is common with OpenGL ES*

Vertex Arrays

- Pass all the vertex values to OpenGL as a single array of values rather than numerous calls to glVertex, glColor, etc.
- Draw objects using glDrawArrays() or glDrawElements()

Vertex Buffer Objects (VBO)

- Stored on the GPU
- Addressed analogous to textures
  - glGenBuffers() - generate unique names
  - glBindBuffer() - select buffer
  - glBufferData() - copy data to buffer
  - glBufferSubData() - copy partial data
  - glEnableVertexAttribArray() - enable array
  - glVertexAttribPointer() - map attribute

glVertexAttribPointer(index,size,type,normalized,stride,pointer)

- index: 0,1,.. must match layout
- size: dimension of variable (1,2,3,4)
- type: variable type (e.g. GL_FLOAT)
- normalize: if true map integers to 0-1
- stride: bytes between data values
- pointer: offset of data values (in bytes)
- *The data comes from the current vertex buffer selected using* glBindBuffer()
- *Activate* glEnableVertexAttribArray(index)

Qt Observations

- QMatrix4x4 is great for GL4 matrices
  - Projections using glu-alikes
  - Transformations using gl-alikes
- QOpenGLbuffer encapsulates VBO
  - Set using Qt methods
- Attach using QOpenGLShaderProgram methods instead of OpenGL calls
  - QOpenGLbuffer specific

OpenGL 3&4 Adoption

- On the desktop, you can do gradual adoption
  - OpenGL 3&4 style shaders with glBegin()
  - VBOs with OpenGL 2 shaders
  - The compatibility profile supports both
- With OpenGL ES it is all or nothing
  - Smaller footprint
  - Fewer legacy implementations