--- In example “Loader loader = **new** Loader();” Loader is giving the type of data (‘int’, ‘float’) ‘loader’ is the new objects name, or variable name. ‘new Loader();’ creates new Loader object

Tutorial 1:

Set up class ‘DisplayManager’ using OpenGL display class:

* createDisplay – | Don’t understand attribs | Gave display width height | created display passing it ‘attribs’ (?object?) | Gave tittle | Using glu created the render viewport as 100% of the display |
* updateDisplay – | Synced | Refreshed |
* closeDisplay -- | Destroy display |

Created ‘MainGameLoop’ class:

* Set up while loop -- | Simpy updates the display every time it runs
* Closes display

Tutorial 2:

Vertex Array Objects (VAO):

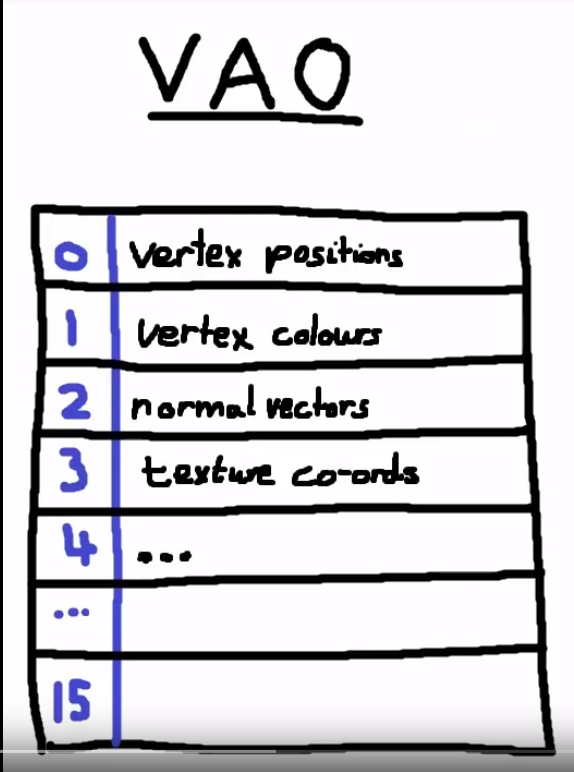
* Slots of data (attribute lists)
* Stores different types of data in separate list positions
* Each list position is called a VBO – Vertex buffer objects
* Each VAO has unique ID
* Must bind to use and unbind at end

VBO:

* Array of data representing colours, positions, normal (anything)
* Stored in separate attrib lists

Example – Storing vertex positions as x,y,z coordinates in a VBO while another attrib list represents the face colours. Together they are put into one VAO representing a character model.

Remember models are made of triangles (3 vertex’s)



Tutorial 3:

Index buffer:

* Stores positions of vertex’s to reduce data
* Because it is split into triangles and it is read counter clockwise this is needed to increase efficiency

Example: ((1,2,3), (,3,2,4), (1,3,2), (1,2,3), (1,7,3), (1,5,3)) -> (0,1,3,3,1,2)

Tutorial 4-5:

Shaders:

https://www.youtube.com/watch?v=AyNZG\_mqGVE&list=PLRIWtICgwaX0u7Rf9zkZhLoLuZVfUksDP&index=4

* Shader = self made rendering functions – Using languahe glsl
* Vertex shader: One time per vertex. Uses VAO as input. Decides where to put vertexs using coordinates, uses input – excecutes program – outputs something to be used as inputs to Fragment shader.
* Fragment shader runs one time per pixel. Uses output of vertex shader to figure out the final colour of that pixel. Output is always RGB.
* VAO -> Vertex shader -> Fragment shader -> pixels
* Complicated

Tutorial 6:

Textures:

* Texures are aligned using (u,v) coordinates. Basiccaly x,y
* 0,0 = top right 1,1 = bottom right
* Stored in VAO as VBO (pos 1)

Tutorial 7:

Matrices and uniform variables:

* Uniform variables can be edited anywhere in the java and sent straight to the vertex or fragment shader avoiding the model data and VAO. Used for lighting instructions and other effecting variables.
* Matricies are used to move around and change things to the models without making new VAO. This also allows for duplication