0

* Preprocessor
  + #include
    - Will open the given header file, and paste its contents into where the “#include” is
  + #define
    - Will search for one word, and replace it with another
    - “#define INTEGER int”
      * Finds “INTEGER”, replaces it with “int”
* Compiling
  + ctrl +f7 – compile only
  + Errors begin with “C” for “compiler”
  + Takes text files and convert them into an intermediate format called an object file. These files are then passed onto the linker, which takes it from there.
  + Every .cpp file sent to the compiler will result in an object file
  + Read generated assebly code
    - properties->c/c++->output files->Assembler Output->Assembly-Only Listing (/FA)
    - checkout the .asm file
* Linker
  + Builds -> compile + link
  + Errors begin with “LNK” for “linker”
  + Finds the functions that you have declared in above code or headers
  + Static function means it can only be used within that translation unit, essentially making it “private” to that file
    - Can still be “included”, but then is now private to that included file
  + Inline
    - Copies function code to where it is called instead of actually calling that function

1

* Open GL is a specifcation/interface, NOT an implementation
* Graphics card manufacturers actually create implenations of open gl

2

* GLFW
  + Lightweight library to easily create cross platform OS windows
  + Linker matches up header files to libraries containing functions
* Setup VS project
  + New empty project
  + Properties
  + C/C++->General
    - Additonal Include Directories
      * $(SolutionDir)Dependencies\GLFW\include
  + Linker -> General
    - Additonal Library Directories
      * $(SolutionDir)Dependencies\GLFW\lib-vc2015
  + Linker -> Input
    - Additonal Dependencies
      * glfw3.lib
      * opengl32.lib
      * User32.lib
      * Gdi32.lib

3

* GLEW
  + OpenGL Extension Wrangler
  + Figures out where the OpenGL function implementations are for your platform and gives you access to them.
  + Add to include directories
  + Add to library directories
  + Add to additional dependencies (.lib)

4

* Open GL is a state machine. You must set the state of GL in order to do things. No object creation.
* Vertex buffer
  + a block of memory (bytes) that is stored on the GPU
* Shader
  + Program executed on the GPU to read that data and display it on the screen
* glDrawArrays
  + used without index buffer
* glDrawElements
  + used with index buffer
* glBindBuffer
  + sets a particular buffer to “active” that will be drawn when GL is called to draw

5

* glVertexAttribPointer
  + define an array of generic vertex attribute data
  + must call “glEnableVertexAttribArray(0);”
  + stride
    - offset b/w each vertex
  + pointer
    - the offset b/w the generic vertex attributes.
* a vertex can contain more than just the “vertex position”, can include texture coords, normals, etc

6

* shader
  + a “block of code” (program) that runs on your GPU
  + vertex shader (called 1st)
    - called for each vertex being rendered
    - for a triangle, gets called 3 times
    - primary purpose is to tell open gl where you want that vertex to be in your window
  + fragment(pixel) shader (called 2nd)
    - runs once for each pixel being drawn(rasterized) on our screen
    - primary purpose is to decide which color our pixel should be
    - don’t do expensive operations here, b/c it gets called for every pixel displayed
    - sometimes calculations are necessary
      * ex: lighting

7

* "layout(location = 0) in vec4 position"
  + correlates to the vertex attribute pointer
  + e.g: “glVertexAttribPointer(0, 2, GL\_FLOAT, GL\_FALSE, sizeof(float) \* 2, 0);”
  + need to use vec4, even if using vec2 for 2d position data
  + b/c we’ve set the vertex attrib pointer, it will cast to a vec4 based upon or settings