**Quad Copter Controller – OpenGL performance analysis**

Due to high PGU usage being observed in the task manager on (24.3.18) some concerns were raised about the optimization and performance of the ground controller implementation.

As it is the ground controller is developed in C++ using an object oriented methodology. Thus each object on screen is a closed object being able to draw itself, relatively independently from any external inputs. Meaning, each object or "sprite" has its own VAO, VBO and shader. This approach to openGL implementation is highly inefficient, but provides the developer to produce a cleaner and more hierarchical code.

The following is a very rough estimate for expected run time spent on executing just the graphical commands of the ground controller.

The ground controller includes the following graphical components on screen:

1. A status bar - 19 sprites
2. A tool bar – 17 sprites
3. A menu screen which includes one of the following options:
   1. Setup control screen – 90 sprites
   2. Manual control screen – 129 sprites
   3. Auto control screen – TBD sprites

Hence, roughly the maximal number of sprites to be rendered is:

The time to perform a buffer change (VBA or VAO) is 1[ms]/1000[sprite]

The time to perform a shader change is 2.9[ms]/1000[sprite]

The above specified numbers are based on numbers provided by: <https://www.gamedev.net/articles/programming/graphics/opengl-api-overhead-r4614/>

The numbers are to be taken with a grain of salt, as they have not been measured for my version of openGL and my specific graphics card.

Hence the total overhead for the object oriented way of doing things is:

For our purposes this provides decent performance.

**Note:** One performance based issue that should change is reloading the font texture per label. Creates a very large overhead for the texture in the GPUs memory. For an estimated number of 60 labels and a font image of 512x512 it will take up roughly 60Mbytes of memory.

**Note2**: Should go through the code and see that I don’t have any memory leaks. Meaning whether I discard memory after I finish using it.