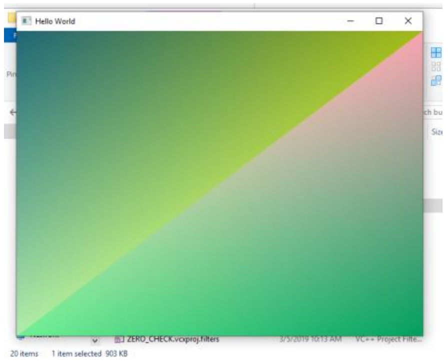
# Assignment_3 – pyrokuna (Ryan Gutierrez)

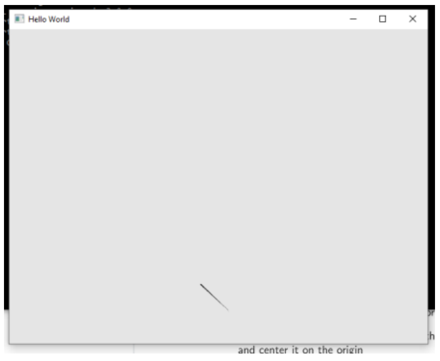
When producing the source code for Assignment 3, I was not able to properly implement any of the parts entirely. Although, I was able to get bits and pieces working. I made attempts to implement everything with the exception of Phong shading. To compile this assignment, open the source code, and replace the file paths in the key_callback function to wherever your bumpy_cube.off and bunny.off files are within your computer. Then use CMake Gui to create a build of the assignment (using visual studios) the way we were instructed for the first and second assignments in this course. Then build in Visual Studios (ctrl+shift+B) and execute the executable it produces.

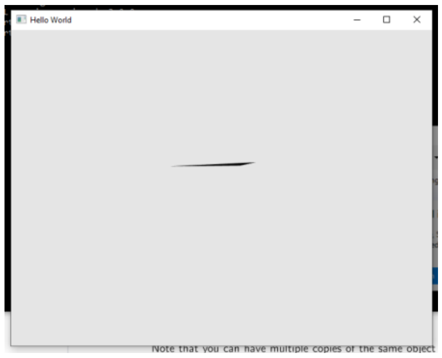
Key Bindings:

| | |
|---|---|
| 1 | Creates unit cube. |
| 2 | Creates bumpy cube. |
| 3 | Creates bunny. |
| W | Goes into wireframe mode. |
| F | Goes into flat shading mode. |
| P | Goes into phong shading mode. |
| UP | Translates objects up. |
| DOWN | Translates objects down. |
| LEFT | Translates objects left. |
| RIGHT | Translates objects right. |
| X | Rotates a programmer designated direction on the X axis. |
| Y | Rotates a programmer designated direction on the Y axis. |
| Z | Rotates a programmer designated direction on the Z axis. |
| DELETE | Deletes a user specified object. |
| I | Translates the camera up. |
| K | Translates the camera down. |
| J | Translates the camera right. |
| L | Translates the camera right. |
| B | Scales up ((B)igger) a user specified object. |
| S | Scales down ((S)maller) a user specified object. |

## Part 1.1





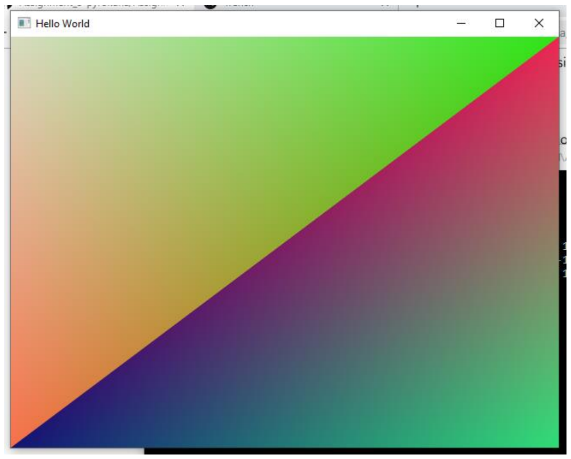


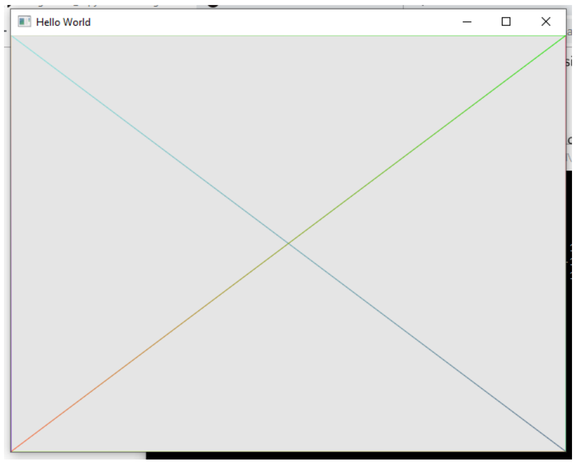
Part 1.1 was in some ways implemented properly. The left most picture shows a unit cube, although not with a proper camera implementation. The center picture displays my attempt at a bumpy cube which, for the most part seemed to be implemented properly, only it was not added to the vertex matrix properly; the same goes for the bunny (the right most picture). In implementing mesh input, I first opened the files, then parsed the data grabbing each vertex and storing them into a two-dimensional array. I did the same with the face data and used the face data to access the designated vertices and attempted (unsuccessfully) to update the vertex matrix.

I did not really attempt to create a light source, although had I done it, I would have gone about it like this: Create a vector which designated its position and update its attributes in the fragment shader.

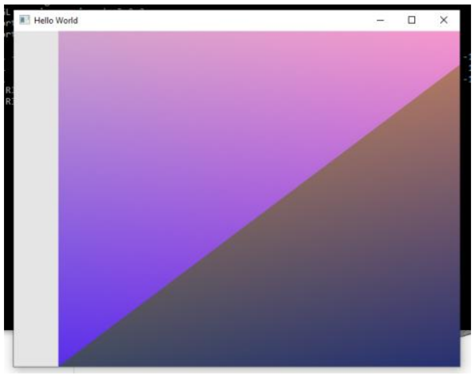
Note that while based on the values of the vertex matrix it seems as though multiple objects can be created, there is no visual proof of this and my program sometimes crashes after multiple attempts at creating new objects (more so when creating bumpy cubes or bunny's, likely because of their size).

## Part 1.2





For part 1.2, I was able to implement wireframe as shown above (left is without right is with) but not able to implement phong shading as I did not have a light source. Although I do understand that in phong shading, the normals are specified on the vertices of the mesh and interpolated in the interior and that the lighting equation should be evaluated for each fragment. I was not able to compute the normals either, but had I done so I would have first computed the per-face normals and average those to compute the neighboring vertex normals.



The two pictures displayed above portray translation to some extent (left with a cube and right with a bumpy cube). While objects do not have their own translation, rotation, and scale, they are able to translate as a whole with the keys specified earlier in the README.

## Part 1.3

Unfortunately, I was never able to properly implement a camera, despite relentless attempts. As far as camera movement goes, I only set up key bindings for that task. In my attempts to implement the camera properly (commented out in my source code), I created model, view, and projection matrices, calculated the necessary values, and updated the vertex shader with all of the necessary information. Despite a seemingly correct implementation, nothing ever appeared in the window when I used this method.