ECS 175 Project 2

Sean Malloy, 998853013

spmalloy@ucdavis.edu

10/29/2018

How To Compile and Run Program

1) On the command line, run “make”.
2) My Makefile should create an executable called “main”
3) Run “main” to start program
4) Run “make clean” to clean-up object files

How to Run Program

1) You can start my program one of two ways:
   1. Run “main *inputFile*”
   2. Run “main”
2) If you provided an input file, then it will grab matrix data from that input file, otherwise it will ask you for an input file.
   1. The executable expects only 1 input file, if you put in multiple input files then it will warn you that there were too many entries and exit
3) After an input file has been specified, the program will ask you to specify which polyhedral you would like to manipulate (id’s 1 to id_number)
4) After your selection, the program will show a menu to choose which operation to do:
   1. Translate
   2. Scale
   3. Rotation
   4. Display
5) If you choose Translate, it will ask you for the translation xyz values (tx, ty, tz), Once you choose your translation, it will translate the vertices and write back to the text file you specified earlier.
6) If you choose scaling, the program will ask you for the scale factor (S). Afterwards it will do the scaling and write it back to the file
7) If you choose Rotation, it will ask you for:

    1. Radians you want to rotate by
    2. Point 1 XYZ
    3. Point 2 XYZ
8) If you choose display, it will just display the current values onto the window.
9) Once you have made your decision on the menu, and once you have correctly input the values you want, it will ask you whether you want to draw all the polyhedras at once, or whether you want to draw one at a time
    1. If you choose y, you draw all
    2. If you choose x, you draw one
10) Once it has finished drawing, it will ask the user whether or not they would want to quit, choosing y quits and n sends you back to the beginning of the menu phase. .

<u>Where did I implement my Algorithms and notes about them:</u>

Note: If no note included, then I suspect that the algorithm is working as intended.

Bresenham: Bresenham.cpp from lines 13 to 128

Translation: main.cpp from lines 150 to 156

Scaling: main.cpp lines 159 to 179

Rotation: main.cpp lines 182 to 243

Note: For the most part it seems to be working correctly, except maybe in certain edge cases, but I am a bit unsure how to test it. It seems fine to me.

Writeback: main.cpp lines 246 to 255 and lines 410 to 425

Set Screen: main.cpp lines 504 to 528

Set Boundary Box: main.cpp lines 531 to 564

World to NDC to Pixel: main.cpp lines 567 to 594

Note: I left my old code at the bottom of the file in case if you are interested on how I implemented them individually, but I choose to combine them together, to cut down function calls.

Also, I added an extra 10 pixels 9moved a bit up left), but also scaled it less by 20 pixels (moved it down 10 pixels to ensure that it is even) from my NDC to Pixel calculations, so that I could have some room from the edge of the screen to the polyhedrons! (was able to figure that out!)

Extra Credit:

- I believe the extra credit was to adjust the bounding box according to when it is needed, and I believe I have done so correctly, where it would increase in size when needed.