Rasterization

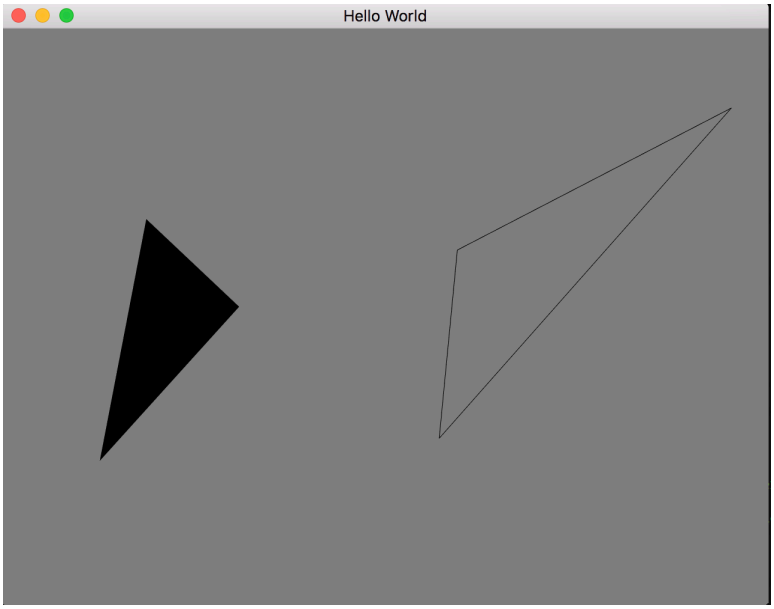
Radhika Mattoo, rm3485@nyu.edu

Overview

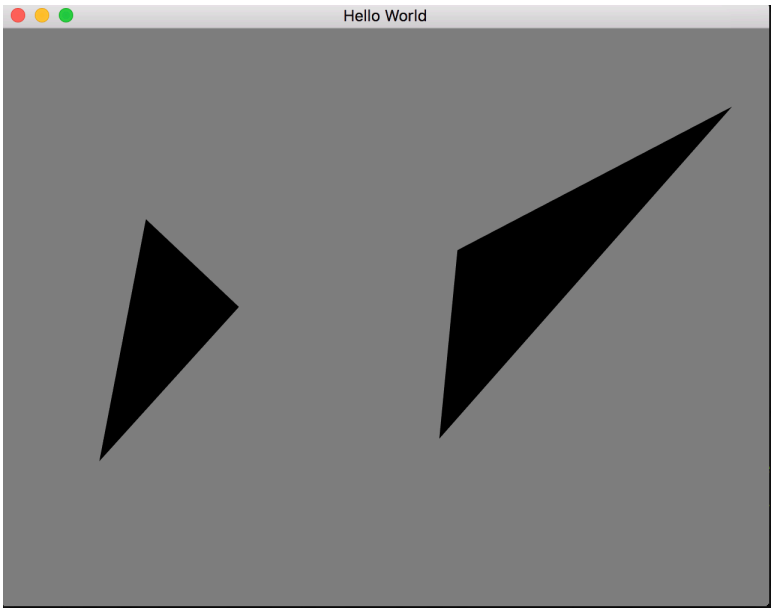
- I have implemented the homework using the letters described for each respective section
- Setup & compilation:
    1. git clone --recursive https://github.com/NYUCG2017/assignment-2-radhikamattoo.git
    2. cd assignment-2-radhikamattoo
    3. mkdir build && cd build
    4. cmake ../
- Running:
    - make
    - ./Assignment2_bin
- If you are in a mode and would like to exit to the default setting, you can press the **Escape** key, but make sure not to press it while in the middle of an action, like inserting a triangle or key framing.

1.1 Triangle Soup Editor

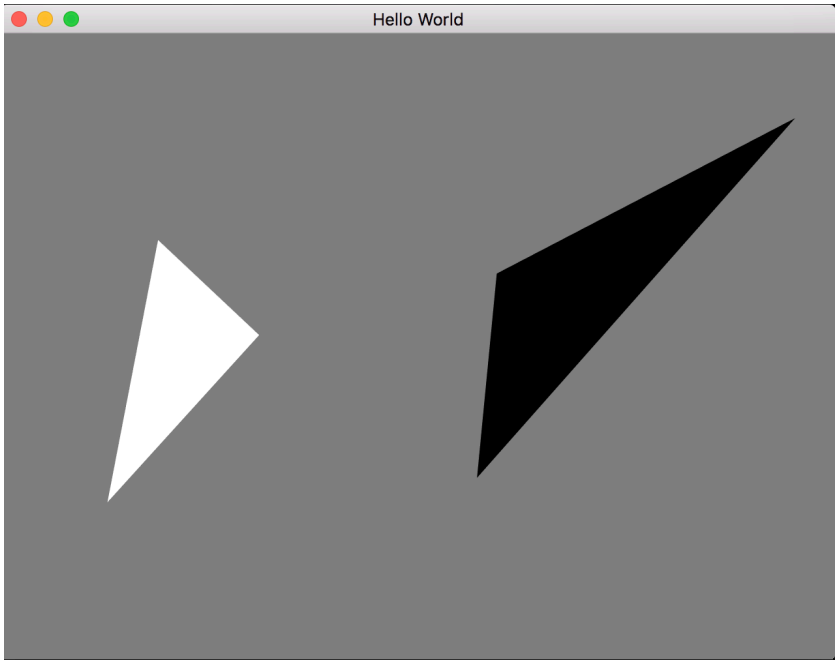
- Press **I** to enter insertion mode, and any triplet of mouse clicks will create a black triangle.
- The screenshot below shows the preview of a second triangle after 2 clicks.



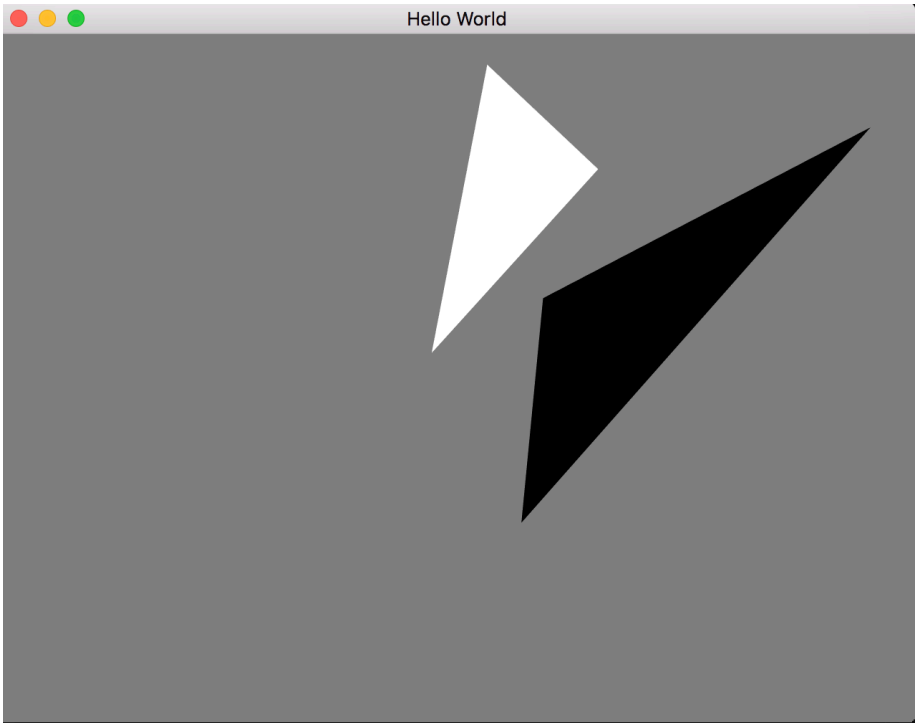
- The screenshot below shows the result from clicking a third time, where the preview from above becomes an actual triangle:



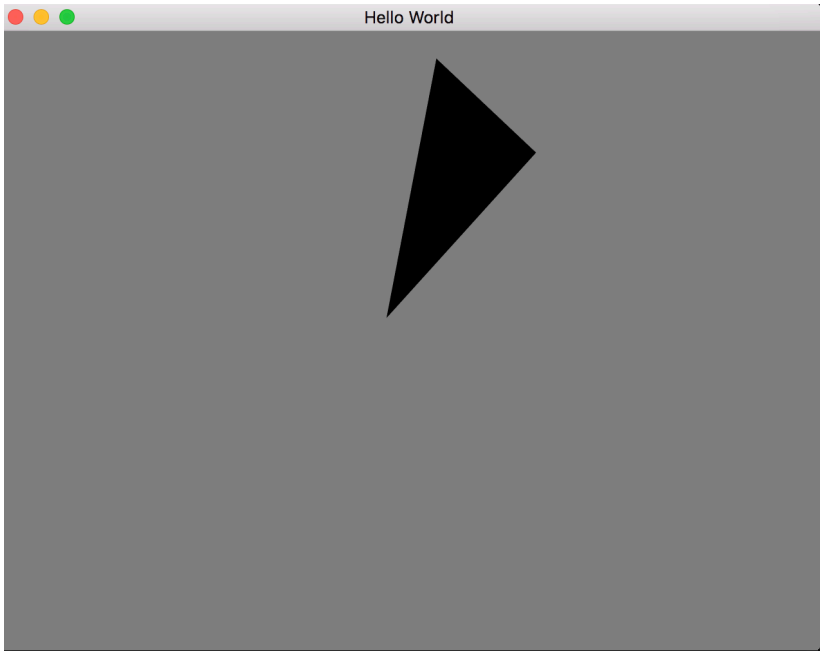
- Press **O** to enter Translation mode. Clicking on a triangle will 'highlight' the triangle by making it white, and clicking & dragging a triangle will translate it according to the movement of your mouse.
- **To account for 1.2's requirements, the triangle stays white after the mouse is released to indicate it is the selected triangle**.
- The screenshot below shows the left triangle was clicked on, thus highlighting it as the selected triangle:



- The screenshot below shows the result of clicking & dragging the selected triangle to a new position:

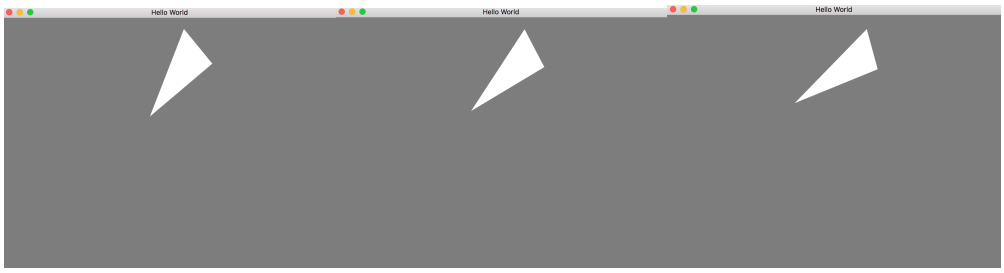


- Press **P** to enter Delete mode. Here, any triangle that is clicked will be deleted. Also, note how pressing Escape unselects any highlighted triangle, and converts it back to its original color.
- The screenshot below shows the result of deleting the right triangle, leaving us with the first triangle we translated above.

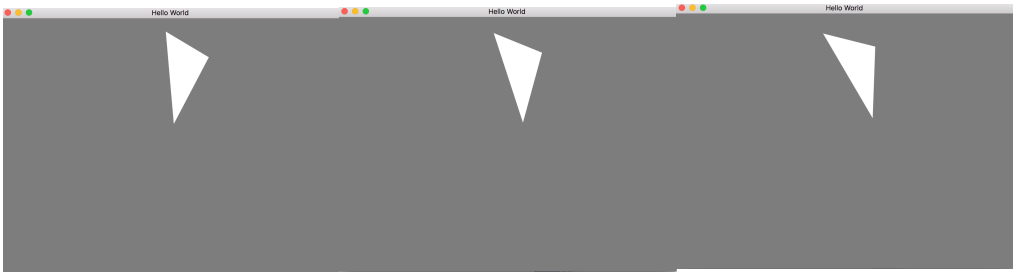


1.2 Rotation/Scale

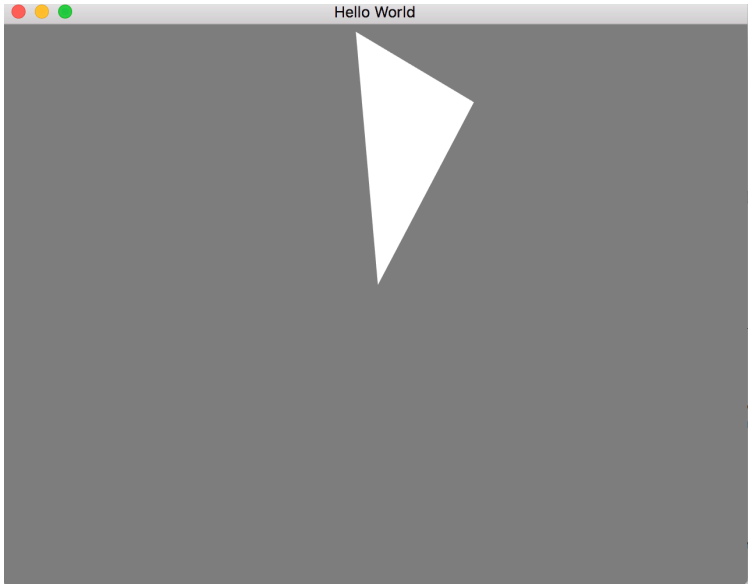
- Press **O** to enter Translation mode. Any selected/white triangle can be rotated/scaled according to the requirements using the **H**, **J**, **K**, and **L** keys.
- The below screenshots are all relative to the original position of the triangle.
- Pressing **H** 3 times, aka rotating clockwise:



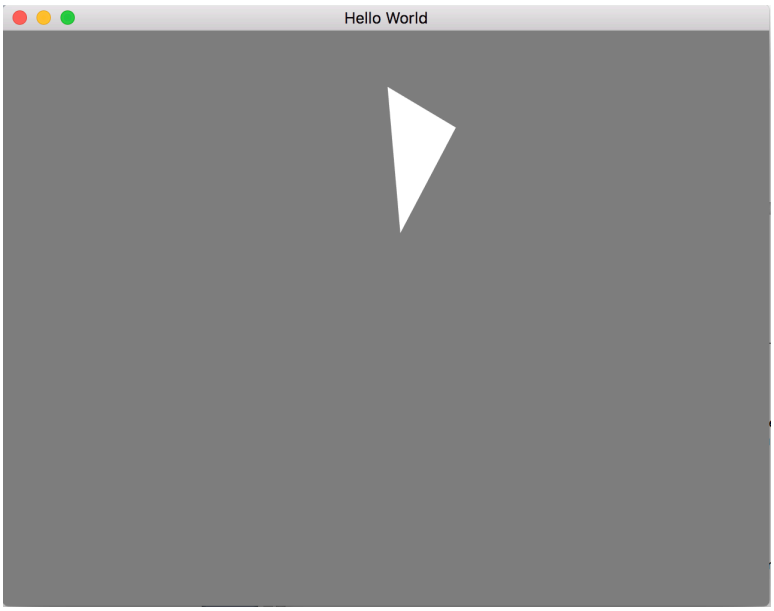
- Pressing **J** 3 times, aka rotating counterclockwise:



- Pressing **K** once, aka scaling up by 25%:

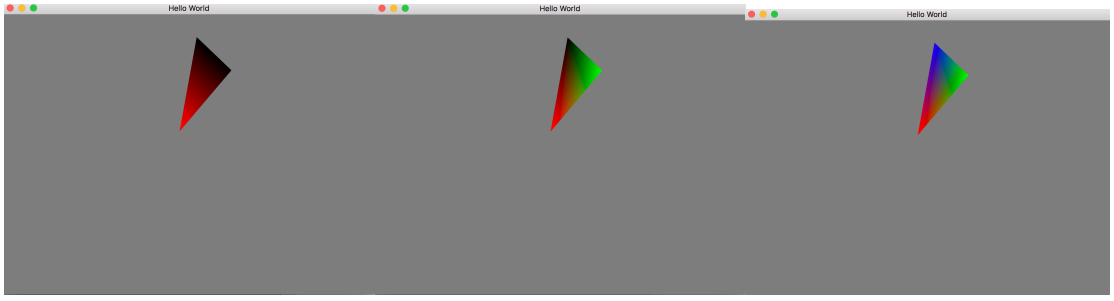


- Pressing **L** once, aka scaling down by 25%:



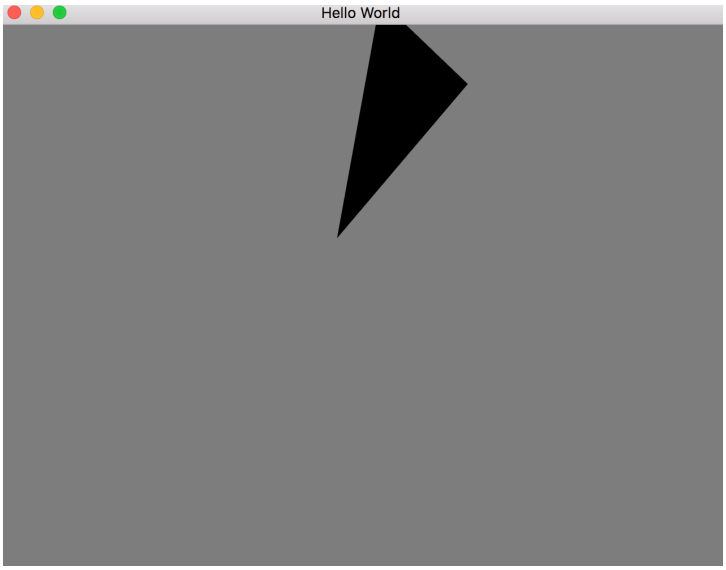
1.3 Colors

- Press **C** to enter Color mode. A click will find the closest triangle vertex, and pressing a 1 – 9 key will color it according to the number using interpolation.
- My color setup:
  - 1 – 3 is increasingly **red** (0.33, 0.66, 0.99)
  - 4 – 6 is increasingly **green** (0.33, 0.66, 0.99)
  - 6 – 9 is increasingly **blue** (0.33, 0.66, 0.99)
- The below screenshots show the results of clicking near each vertex, and pressing 3, 6, and 9.

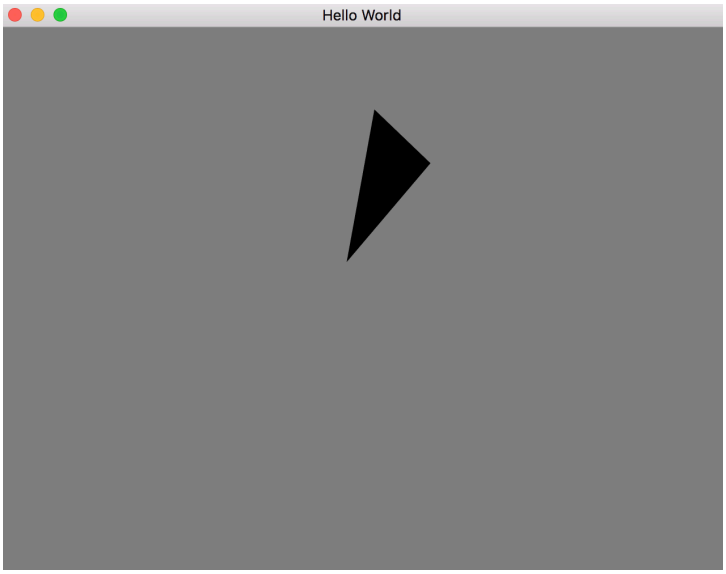


1.4 View Control

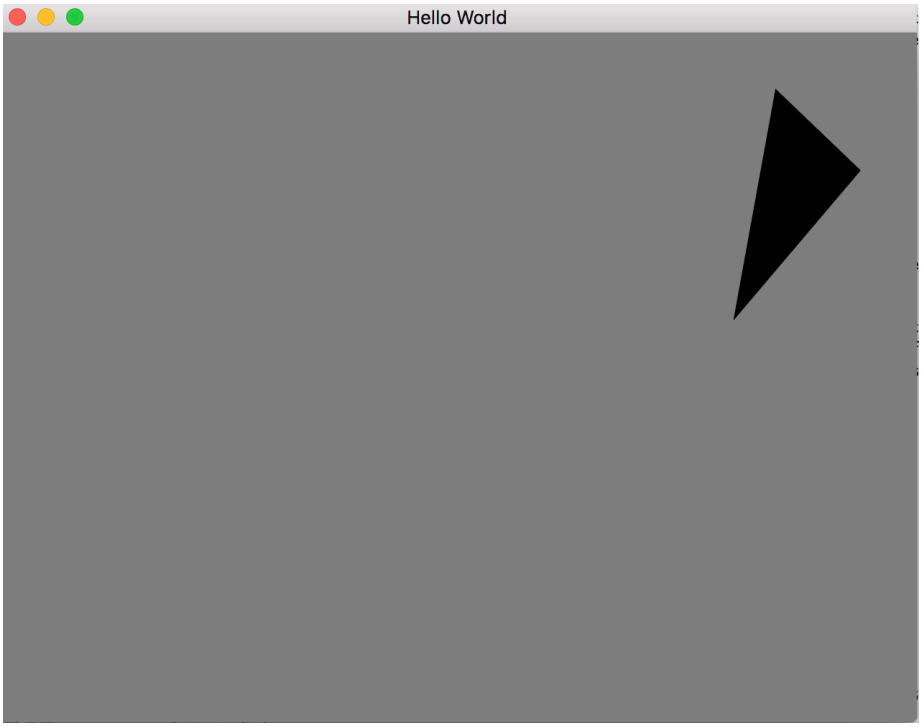
- Pressing + or – will zoom out and in on the screen using a view matrix.
- Pressing **W**, **A**, **S** and **D** pans the view down, right, up, and left, respectively, by 20% of the window size.
- The below screenshots are all relative to the original position of the triangle.
- Pressing + once:



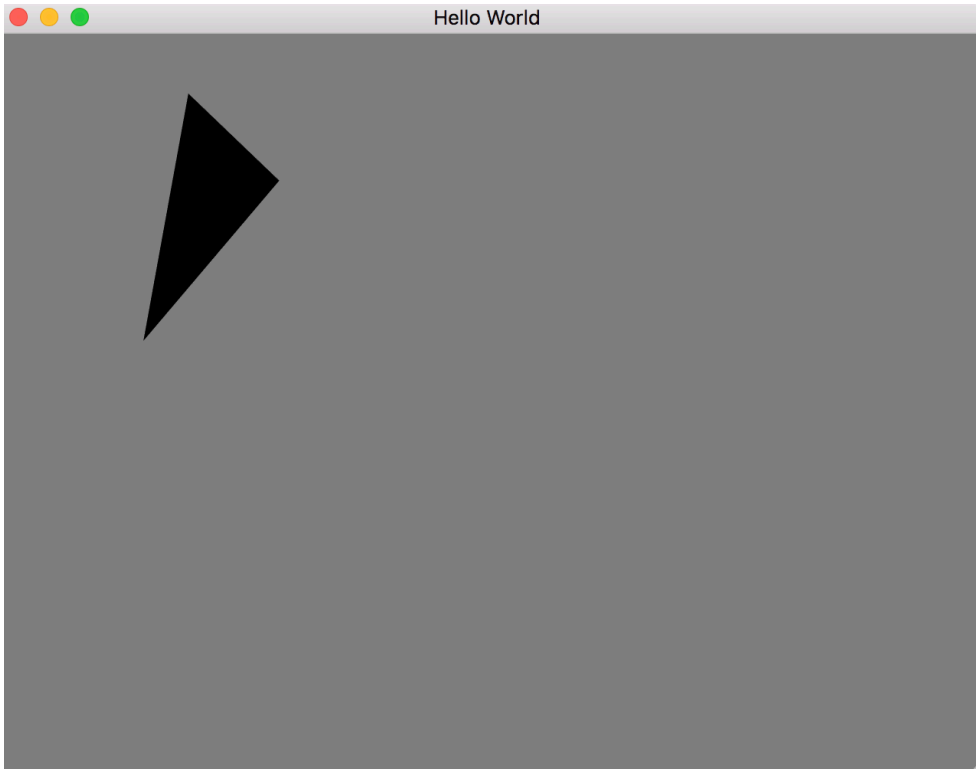
- Pressing – once:



- Pressing **A** once, aka panning right:

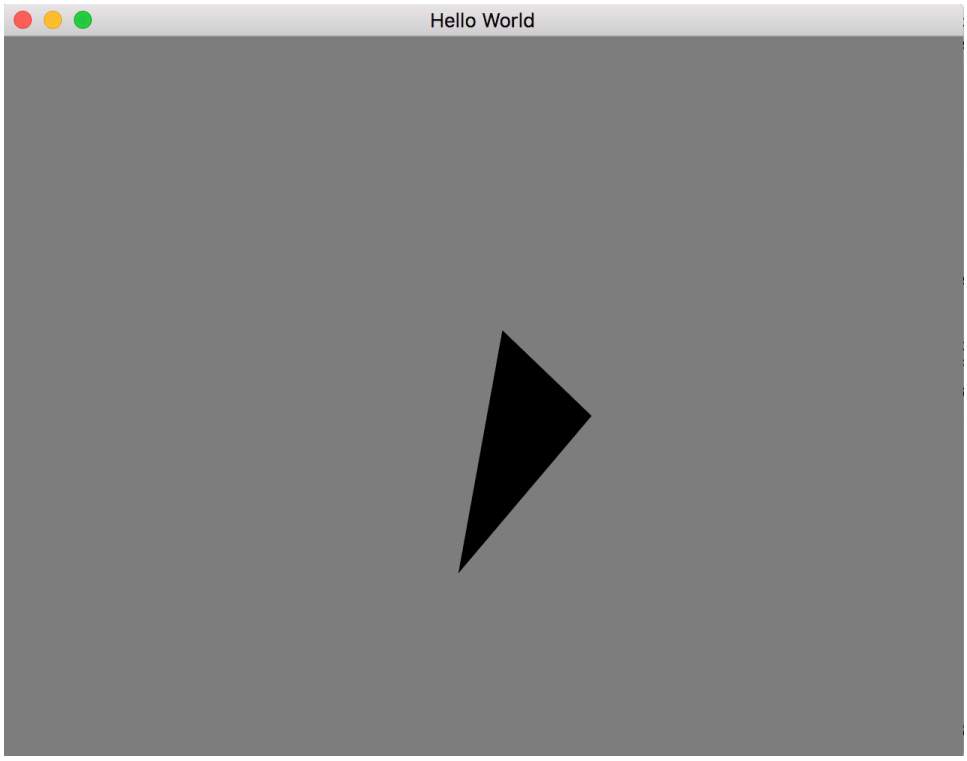


- Pressing **D** once, aka panning left:



- Pressing **S** once, aka panning up:

- Pressing **W** once, aka panning down:



1.5 Keyframing

- My animation is the movement of a triangle using linear interpolation and a timer to make it automatic.
- Press **R** to enter 'Record' mode. Here, clicking and dragging a triangle will both select it and record the starting & ending position of the action. Press **G** to animate the triangle moving (linearly) from the start to end position over 1 second.
- When the animation terminates, animation mode is automatically exited and the triangle stays in the final position of the animation. To perform another animation, you must repeat the previous step.
- The gif for the animation couldn't be embedded in the Word document, so it is located in the base directory for this homework as **animation.gif**
- Note that the gif shows 2 separate animations, where the 2nd animation recording is dragged all over the screen, but the animation is still linear according to the start position (when you click) and the end position (where you release)

1.8 Shader Translation/Scaling/Rotation

- I've implemented extra credit section 1.8, where you never touch the original triangle positions in V, and perform all transformations in the vertex shader using matrix multiplication.
- I have global, dynamically re-sized **model**, **transformation**, **scaling**, and **rotation** variables in my program that keep track of the model matrix for each primitive, and pass each one as a uniform into the vertex shader when drawing said primitive.
- If a triangle is scaled, rotated, or translated, matrix multiplication of the necessary matrices are constructed and placed into the primitive's model matrix.