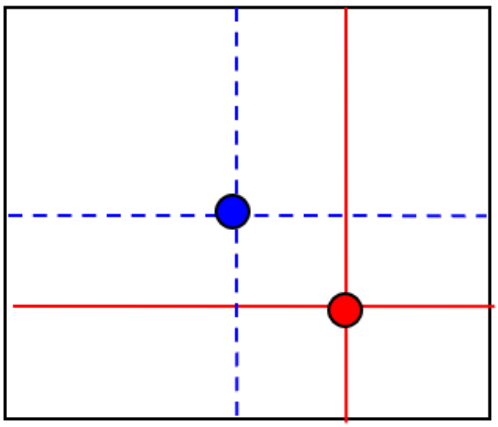
Assignment 4 Writeup Ryan Shendler

For this assignment, I implemented three different programs. The first program loads the flower.bmp image and then allows the user to warp the image by dragging the center point. The second program loads the flower.bmp image and then rotates it around the z-axis. The third program loads the file teapot.obj as a 3D model and allows the user to view it from different angles.

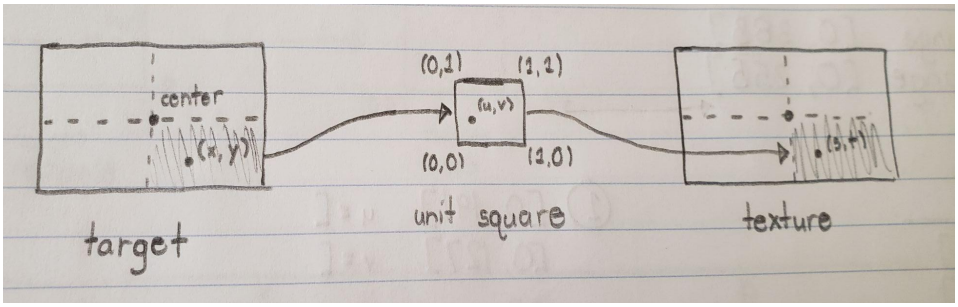
The first program is implemented in warp.cpp. In order to implement the texture warping, I treated the texture image as being made up of four rectangles, all of which shared a vertex at the center point.



My implementation for the texture warping

In order to warp the image, users can click and drag the center point, which will change the value of the global array center. This array stores the coordinates of the current location of the center point. The actual texture warping code is implemented in the function renderPoint(). This function takes a point in the target rectangle and calculates the corresponding point in the texture space. The function then sets the color of the point to match the color of its corresponding texel. I calculated the corresponding texel coordinates by first determining which of the four

sub-rectangles the point lies in. I then mapped this sub rectangle to the unit square and calculated the new coordinates of the point. Finally, I used bilinear interpolation to calculate the coordinates of the corresponding texel.



My transformation pipeline

Because the quadrilateral in the texture space that I am mapping to is a rectangle, the bilinear interpolation calculation simplified to the window-to-viewport mapping calculation that we implemented in Assignment 2.

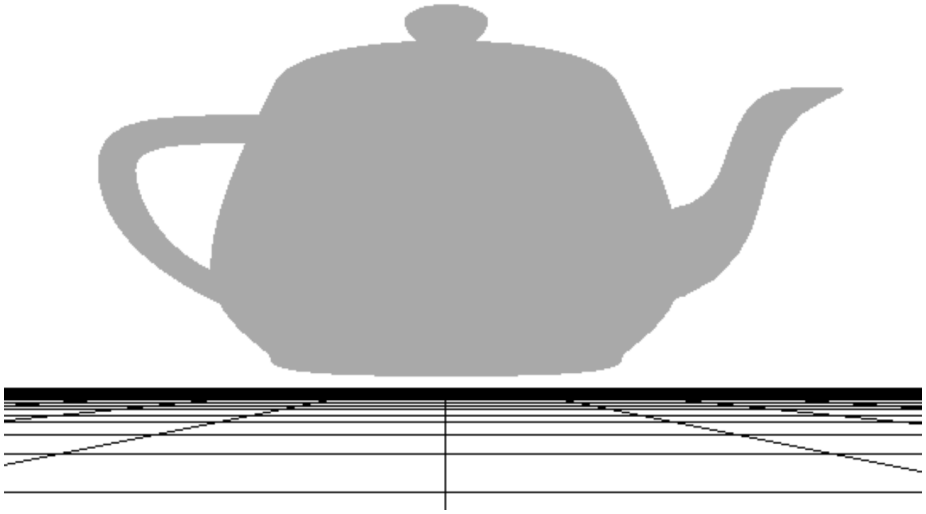
Image before and after warping

The second program is implemented in rotate.cpp. This implementation uses the loadImage() file to load the image and then uses the glRotatef() function to rotate the image around the z-axis. Although the assignment said to display the image every 30 degrees, I found that this made the animation very choppy and made it difficult to get a good look at the image

before it jumped to its next location. Therefore, I set the animation to display the image every 0.05 degrees, which makes the rotation of the image appear much smoother.

Image before and after a rotation of approximately 30 degrees

My third program is implemented in obj.cpp. This implementation uses the function loadObj() to load the vertices and faces from the file teapot.obj. It then uses the function drawObj() to draw a 3D model of a teapot using these vertices and faces. I also copied over my dynamic camera implementation from Assignment 4, which allows the user to view the teapot model from different angles. My camera implementation has the following controls: “W” and “S” keys for pitch, “A” and “D” keys for yaw, left and right arrow keys for roll, and up and down arrow keys for sliding along the viewing direction.



The teapot 3D model loaded from teapot.obj

In order to run my code, all you have to do is open the Assignment4.sln file in Visual Studio and build the project that you want to run.