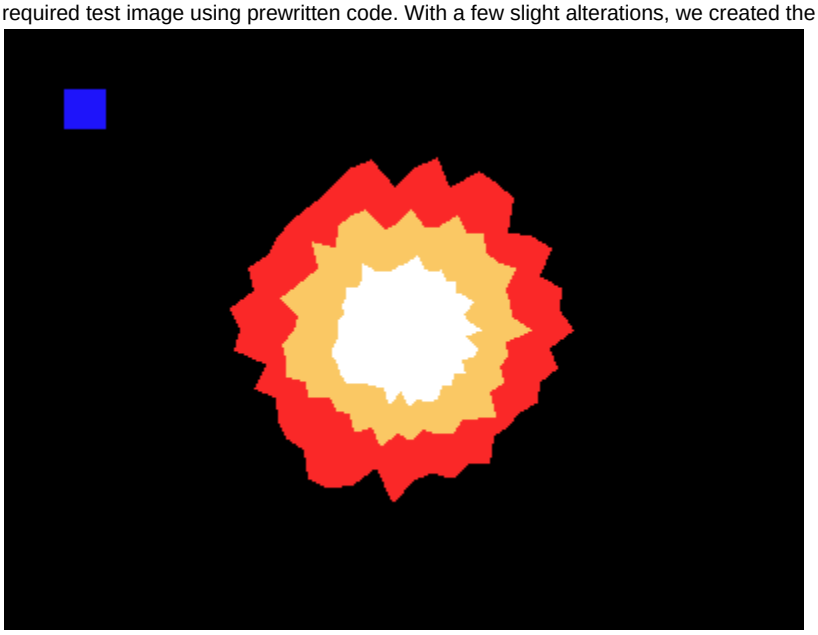
Russell_Lowenstein Assignment 4

CS 351 Assignment 4: Scanline Fill

Justin Russell & Adam Lowenstein

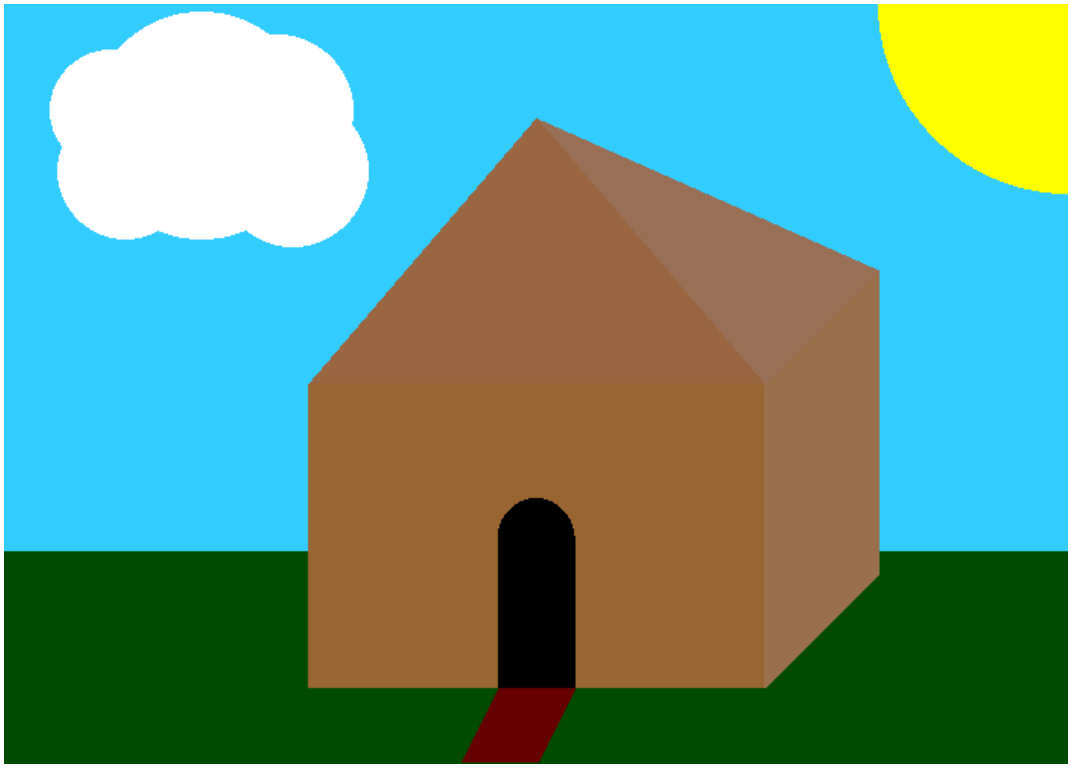
Abstract
The goal of this assignment was to create images of polygons and polylines and implement the scanline fill algorithm on polygons. We first created definitions for polygons, polylines, and vectors. Then, we created polygons and polylines defined by an arbitrary number of vertices which were not restricted to the image boundaries. Our scanfill code was based on the skeleton provided by Professor Maxwell. The scanfill algorithm needed to account for polygon edges and vertices outside the plane of the image. After creating the algorithms and fill functions, we generated a required test image using prewritten code. With a few slight alterations, we created the following image:



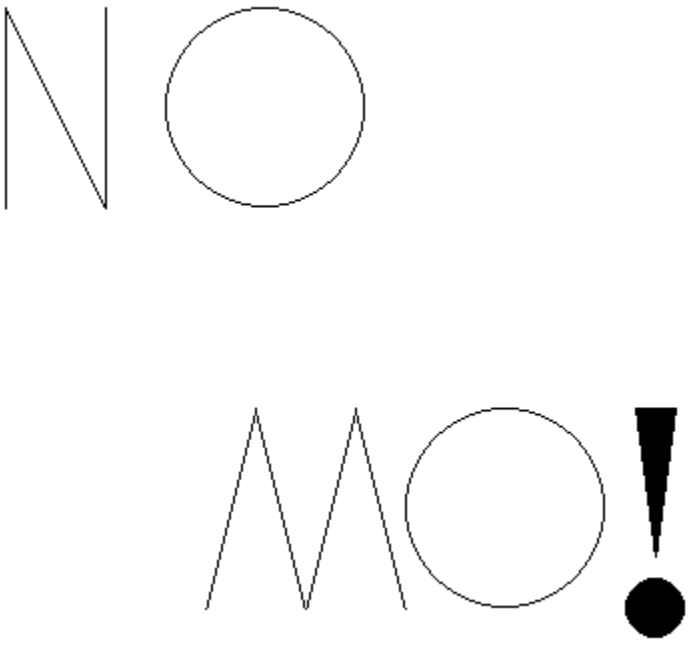
We then created two custom images as shown in the sections below.

Description
We began this project by defining a vector type structure that was identical to the point structure in everything but the name. We also created polygon and polyline structures; polylines were simply polygons that did not close (and in turn could not be filled). They both had similar functions, took an arbitrary number of vertices (stored as an array), and could be mapped (but not drawn) outside the image boundaries. Then, to demonstrate the capabilities of these functions, we generated two images. One highlights the scanfill algorithm, and the other showcases the polyline functions.

Algorithms & Pictures
The first image below was created to represent a 3D scene. It features a number of differently-shaped polygons of a varying number of vertices that are all filled in with different colors using the shade fill algorithm. It also shows how we were able to combine the circle functions from Assignment 3 and include them in this assignment:



The second image utilizes polylines to convey a secret message. It is accented by a scanfilled exclamation point in the bottom righthand corner:



Conclusion

This assignment continued to reinforce the importance of always writing strong code that is applicable in any situation. If an algorithm only works in special cases, like polygons that are always contained with in the canvas boundaries, it's only valid in those specific cases. It took some extra

work, but our code can be applied to any shape with any coordinates (as evidenced by the sun in our pastoral scene, which is anchored in the top righthand corner). It was a rewarding assignment because we were finally able to see our work start to come together in a more realistic way. Lines are great, but filled shapes of all sizes are better and much more widely applicable.
