
Scan Conversion

Due date: Wednesday, October 6, 12:00 pm

1 What your program should do

Write a program that performs two tasks: line drawing and scan conversion for polygons with texture mapping. The program should be interactive, that is, the endpoints of the lines and the vertices of the polygon should be defined interactively by the user. Switching between the two modes can be done using the GLUT pull-down menus or the space bar.

In this assignment only, you should not use OpenGL functions for drawing lines and polygons. Create an internal array of RGB values, and do all drawing there; display the image using OpenGL raster functions `glRasterPos`, `glDrawPixels`, as it was explained at the lecture.

Your program should do the following:

- Draw line segments using Bresenham algorithm. The user picks two endpoints with the mouse; when the second endpoint is picked, the line is drawn; old line segments should not be erased as the new ones are added. An important thing to remember: in the `draw` callback in your code you should not assume that the results from the previous calls are visible; you should start from scratch by calling `glClear`. For this purpose, you need to maintain a list of line segments specified by the user.
- Read an image in ppm format (see the description below) and display it. The user can pick four points in the image; calculate the transformation that would map the original image into the quadrangle picked by the user, and compute the warped image using scan conversion.

For the purposes of calculating the transformation, assume that the coordinates in the image extend from -1 to 1 in x and y directions.

Assume that vertices of the quadrangle are picked in order, i.e. the point of the image with coordinates (-1,-1) should map to the first point, (1,-1) to the second picked point, (1,1) to the third and (-1,1) to the last. Check the input for validity, i.e. detect and refuse to accept concave and self-intersecting quadrangles, and quadrangles with three vertices on one line.

2 What to turn in

A working version of your program; please follow the instructions on the Web page when submitting your homework!

3 PPM format description

This description is from the SGI's IFL man page. The portable pixmap format is a lowest common denominator color image file format. The definition is as follows:

- A “magic number” for identifying the file type. A PPM file’s magic number is the two characters P3.
- Whitespace (blanks, TABs, CRs, LFs).
- A width, formatted as ASCII characters in decimal.
- Whitespace.
- A height, again in ASCII decimal.
- Whitespace.
- The maximum color-component value, again in ASCII decimal.
- Whitespace.
- Width×height pixels, each three ASCII decimal values between 0 and the specified maximum value, starting at the top-left corner of the pixmap, proceeding in normal English reading order. The three values for each pixel represent red, green, and blue, respectively; a value of 0 means that color is off, and the maximum value means that color is maxed out.
- Characters from a # to the next end-of-line are ignored (comments).
- No line should be longer than 70 characters.

Here is an example of a small pixmap in this format:

```
P3
# feep.ppm
4 4
15
0 0 0 0 0 0 0 0 0 15 0 15
0 0 0 0 15 7 0 0 0 0 0 0
0 0 0 0 0 0 0 15 7 0 0 0
15 0 15 0 0 0 0 0 0 0 0 0
```

Programs that read this format should be as lenient as possible, accepting anything that looks remotely like a pixmap.