

Fall 2006
22C:151 Introduction to Computer Graphics
Assignment 3

Due: Wednesday September 13th at 11:59pm

Goal: Write a program that scan converts triangles and outputs them to a PPM file. Write a second program that draws the same triangles using OpenGL and GLUT.

Problem 1 (10 points): Write a program that draws the following triangles into a 512×512 buffer (in memory). The background should be black (i.e., $RGB = [0,0,0]$). Output the result to a PPM file, convert the results to JPG, GIF, PNG, or BMP and display them on your web page. Note coordinates are specified relative to an origin in the lower-left corner of the image. (*This program should not use OpenGL*)

- A **red** triangle with vertices (1, 1), (31, 1), and (31, 31)
- A **white** triangle with vertices (45, 450), (70, 420), and (95, 450)
- A **blue** triangle with vertices (155, 250), (225, 220), and (195, 290)
- A **green** triangle with vertices (80, 220), (150, 250), and (110, 290)
- A triangle with a **green** vertex at (430, 500), a **red** vertex at (500, 430), and a **blue** vertex at (510, 510)
- A triangle with a **yellow** vertex at (300, 40), a **blue** vertex at (500, 70), and a **white** vertex at (450, 80)
- A triangle with a **purple** vertex at (300, 300), a **black** vertex at (400, 300), and a **black** vertex at (300, 400)
- A triangle with a **green** vertex at (300, 400), a **green** vertex at (400, 300), and a **black** vertex at (400, 400)
- Also draw two white lines (use code from Homework 2) from (300, 399) to (400, 399) and from (299, 300) to (299, 400). (*If these overlap with any of the above triangles, please draw the lines over the triangles.*)

Problem 2 (10 points): Write an OpenGL/GLUT program that draws the same triangles (and lines) as in Problem 1, using the same colors. Capture the result in an image and post it on your web page. Make sure to label your images, so I know which is from Problem 1 and which is from Problem 2.

Extra Credit (2 points): Combine Problems 1 and 2 into a single OpenGL program, with an GLUT keyboard callback to switch from OpenGL rendering to your scan conversion, making sure to label which approach is currently being displayed. When using your scan conversion routine, you should use `glBegin(GL_POINTS);` to draw individual pixels.