

**Fall 2009**  
**22C:151 Introduction to Computer Graphics**  
**Assignment 2**

**Due: Wednesday September 9<sup>th</sup> at 11:59pm**

**Goal:** Write a program that rasterizes lines (using the Bresenham/midpoint algorithm) and outputs them to PPM files. Write a second program that draws the same lines using OpenGL and GLUT.

**Problem 1 (10 points):** (8 points) Write a program that rasterizes the following lines (using the Bresenham algorithm) into a  $512 \times 512$  buffer (in memory), and output the result to a PPM file. The background should be black (i.e.,  $RGB = [0,0,0]$ ). Use a utility program to convert the PPM file to GIF, PNG or BMP to display on your web page (e.g., use *ppmtogif*, *ppmtobmp*, or *GIMP* on Linux systems or *GIMP* on Windows systems). To reduce data-entry time, these points are available online in an electronic format. (Note: This program should not use OpenGL.)

- **Red** lines: (1,1)→(1, 510), (1,510)→(510,510), (510,510)→(510,1), (510,1)→(1,1)
- **Green** lines: (32,420)→(82,395), (30,316)→(90,335), (82,395)→(30,316)
- **Purple** (red + blue) lines: (100,500)→(147,507), (104,446)→(149,448), (102,404)→(150,380), (102,404)→(104,446), (147,507)→(149,448)
- **Blue** lines: (176,340)→(223,341), (176,340)→(185,228), (237,240)→(185,228)
- **White** lines: (250,300)→(250,280), (248,240)→(253,220)
- **Yellow** (red + green) lines: (293,249)→(307,286), (307,286)→(307,144), (264,145)→(347,148)
- **Cyan** (green + blue) lines: (405,231)→(361,213), (361,213)→(366,181), (366,181)→(421,131), (421,131)→(358,82)
- **Orange** (red + 0.5\*green) lines: (444,174)→(474,179), (474,179)→(442,38), (391,38)→(493,45)

(2 points) In a second  $512 \times 512$  image, draw the same lines, but change colors gradually from one end to the other. For example, draw the line from (1,1) to (1, 510) starting at red and gradually blending to green at (1, 510). I want you to think about and come up with a blending method on your own. There is no right or wrong way to blend here, as long as the color varies gradually over the line. Make sure to use contrasting colors at the endpoints (white & black, red & green, yellow & blue, etc.)

**Problem 2 (10 points):** (5 points) Write an OpenGL/GLUT program that draws the same lines as in Problem 1 (using the same colors). Using a *keyboard callback*, allow the user to switch back and forth between the solid lines (`glShadeModel(GL_FLAT);`) and those with gradually varying colors (`glShadeModel(GL_SMOOTH);`) by hitting the key 'c'. Capture the result in an image and post on your web page. Make sure to label your images, so I know which is from Problem 1 and which is from Problem 2.

(5 points) Allow the user to draw an additional line by clicking the *right* mouse button at the two endpoints.

**Extra Credit (2 points):** Using code from Problem 2, write an OpenGL-based program that displays and updates the current time (i.e., a clock). Don't make this too hard: use your line drawing routines from above.

**NOTE:** Please remember to include a "README" file with your submission detailing how to compile, how to run, and any problems that might exist with your submission. Also remember to post result images (but not code!) on your webpage.