CS352 Computer Graphics – Program 3
Raster Algorithms

Assigned on Wednesday, April 2, 2003
Due Date and Time: Monday, April 15, 2003, BEFORE class

Goal
In this program, you will write a set of routines for scan conversion of lines and circles, as well as the flood-fill algorithm.

Getting Started

I have provided a skeleton application that does everything in terms of taking care of creating a window and handling the user interface. This skeleton application is written using OpenGL and GLUT library. However, other than making sure that the required .dll and .lib files are located properly on the machine you are working on, you do not need to know OpenGL to attempt this homework. All you need to do is to look for the /* your code here */ comments in the skeleton program to figure out where to start.

At the very start of your work, it is advisable to make sure that the skeleton application complies and runs properly on your machine. If it does not, that is an indication that OpenGL and GLUT files may be missing. GLUT installation files are located in the Install folder on \Badar\Common\CS352. If you have difficulty in installation, seek help from Sadaf Shams.

When you run the skeleton application, you should see a black window. If you click the right mouse button in the window, you will see a small dot painted at the location of the mouse. If you press ‘c’, the dots will clear.

In the skeleton program (skeleton.cpp, downloadable here), you will simply manipulate the image data structure. This is a 3D array, with the first index being the x-coordinate (column), the second being the y-coordinate (row) and the third being the r, g, or b layer. For example, image[10][20][0] refers to the red color component of the 11th column from the left and the 21st row from the right (note that indices start from zero). Whatever changes you make to this image will be reflected in the window on the screen. The size of the image is fixed at 250×250. It is your responsibility to ensure that out of range array accesses do not happen.

The skeleton program also provides callback functions for mouse and keyboard input. Example functions are included in the skeleton application. You can add more callbacks in a similar fashion, as needed. Some simple functionality is already added, like a function to clear the whole window to a single color.
Tasks

1. Write a function to draw line segments using the midpoint algorithm. Look for the function `void DrawLine(int x0, int y0, int x1, int y1, unsigned char r, unsigned char g, unsigned char b)` and write your code there. This function should draw a line in image from `(x0, y0)` to `(x1, y1)` in color `(r, g, b)`. The function must not use any floating point operation at all. You do not have to implement clipping: i.e if any of the end-points specified are outside the window, you may choose to not draw the line at all.

   This function will get called when you press ‘1’ and after that, press the left mouse button and drag the mouse. The location of pressing the button down specifies first end point, and the location of releasing the button specifies the 2nd end point.

2. Write a function to draw a circle using the midpoint algorithm. You should use the optimized version of the algorithm as discussed in class. Look for the function `void DrawCircle(int xc, int yc, int radius, unsigned char r, unsigned char g, unsigned char b)` and write your code there. This function should draw a circle in image centered at `(xc, yc)` in color `(r, g, b)`.

3. Write a function to flood-fill a shape. Flood filling is a simple recursive algorithm to color the inside of a closed shape. It works as follows: The input is a pixel location and a paint color. Whatever is the color at the input point is taken to be the background color. This color is replaced by the new paint color. Then the four neighbors (top, bottom, right, left) are recursively visited. If they are of the background color, they are made of the paint color. Recursion keeps happening till the boundary pixels of the shape are reached, which, by definition should have a different color than the background color.

   Look for the function `floodFill(int x, int y, unsigned char r, unsigned char g, unsigned char b)` and write your flood-fill code there. This function is automatically called when the right mouse button is clicked.

4. Write another function to draw ‘fat’ circles. This can be done by drawing a circle of radius $r_i$ and $r_o$ and floodfilling the interior. Or you could check the candidate pixels on whether their distance from the center is between $r_i$ and $r_o$. There could be several ways to improve efficiency there. Document your procedure in your report.

5. Modify your ‘fat’ circles program so that it draws anti-aliased circles. Do this by ‘super sampling”; subdividing each pixel into several subpixels and then deciding how many of these are actually inside the circle.
6. Another function provided in the code is `executeScript()`. This function is called when ‘s’ is pressed. You can write a script of your commands in this function, and they will be called on key-press. Write a script to make a complicated, interesting drawing, with multiple colors using your simple program. Give a title to your art work and submit it along with your report. (Leave the script within code, so that your code can be tested using the same script). The best drawings will be displayed in class ☺

These tasks are required. In addition, you may add additional functionality, like making patterned lines, pattern filling, rectangle drawing, paint brushes and polygonal shapes. Your must try at least one of the extra ‘bells and whistles’ to obtain full credit.

Happy Painting!