CISC 829 Computational Geometry

HW3 due May 14

WRITTEN EXERCISE:

The following is problem H fron the recently completed finals of the ACM Programming Context. As it turns out, no team was able to complete this problem. Of course, they have not had the benefit of this course on computational geometry …

You probably never heard of the artist Peer. He is not well known, much to his regret. Peer was one of the inventors of *monochromy*, which means that each of his paintings has a single color, but in different shades. He also believed in the use of simple geometric forms. During his triangle period, Peer drew triangles on a rectangular canvas, making sure their borders did not intersect. He would then choose a color, and fill the regions. Peer would paint the outermost region (the canvas itself) with the lightest shade of the color chosen. Then step by step, he would fill more inner regions with a darker shade of the same color.

In a way the process was quite mechanical. The only thing Peer considered difficult was to decide, after drawing the triangles, how many different shades he would need. You must design an algorithm to do that calculation for him.

Your algorithm will have a collection of triangles as its input. It should calculate the number of different shades needed to paint the regions according to the given rule. Your algorithm must also detect the rare times that Peer makes a mistake and draws triangles that intersect. Two triangles are considered intersecting if the edges of one triangle have at least one point in common with the edges of the other. In that case, the collection of triangles is invalid.

Be sure to carefully describe your algorithm and to analyze the running time, which should be as small as possible …

PROGRAMMING EXERCISES:

The programming exercise this time has these objectives:

1. Familiarize with Delaunay Triangulation
2. Really understand how a heap works by implementing it
3. Get some hand-on experience with Loop subdivision
4. Treat an image as a surface

A. Find a grayscale image of a relatively high resolution (say 512x512). Randomly sample 10% of the pixels and treat them as points. The height of each point corresponds to its intensity.

B. Use any publicly available libraries or implementations (e.g, OpenCV) to construct the Delaunay Triangulation of these points.

C. Rasterize all triangles (say using OpenGL, and yes, your graphics group member can help) to obtain an image with the resolution of the source image and compare its quality with the source image.

D. Now implement Loop subdivision to generate a finer mesh from the base mesh.

E. Repeat step C and re-compare the image quality.

F. You can view this approach as an image super-resolution algorithm. Experiment your program on different images and discuss why it works better for certain type of images than the other.