Project 3 : Voronoi Diagram Mosaics

1 Overview

OpenGL can be used not only for drawing things, but also for solving various geometric problems. Here, we will use it to compute *Voronoi diagram* of a collection of points in the plane.

Assume you are given a finite number of points, $p_1, p_2, \ldots, p_n$. Let's call them *sites*. For each site, there is an area around it (called Voronoi region) which consists of all points for which that site is the closest site. If you are somewhere on the plane and want to get to some site as quickly as possible, you would want to walk to the site whose Voronoi region you are in, since it's the closest one.

Approximate Voronoi diagrams can be drawn using OpenGL as follows. For each site, plant a cone at that site. Assuming the plane is horizontal (think of it as the ground) the cones grow perpendicularly up. The angles of the cones are all the same. Assign different colors to the cones and draw them, using a parallel projection in the direction perpendicular to the ground. When drawing, use no illumination, just the cones' colors. Thus, the color of a pixel will be the same as the color of the cone that can be seen from that pixel when looking (vertically) up.

In this project, you will program Voronoi diagram computation in OpenGL, that is, set up the projection as described above, model the cones (using triangles) and draw them in different colors. To make it more fun, the colors for the cones could be either random or selected from a ppm file. It will probably be best to use a filtered (smoothed) image for best results – use your favourite image processing program to do that.

2 Details

Your code should start by showing a Voronoi diagram of 32 randomly chosen (all within the viewport) sites. Choose the colors for the sites randomly (say, from a uniform distribution on rgb values).

Program the following menu items.

Spray more points. Increase the number of sites by a factor of two (e.g. if we choose this menu item seven times, we would like to see a Voronoi diagram of 4096 sites).

Show/hide sites. This one should toggle between two modes: sites visible (drawn as black dots) and sites invisible (not drawn at all, just their Voronoi regions visible). Initially, show the sites.

Reset. Go back to the first setting: 32 sites, random coloring of Voronoi regions, no movement, sites shown as black dots.

Move points. This should cause the sites to move along circular trajectories around the center of the window. Choose the angular velocity of each point

randomly (make the velocity of each site different). Try to experiment with velocity so that things look nicely.

Toggle coloring method. There are two coloring methods we'll use. The first is just random colors for each Voronoi region. The second is colors looked up from a square ppm image. You can assume this will be an ascii PPM file (such as you produced in project 1), with the same x- and y- resolutions. Scale the image so that it tightly fits into the window before using it for looking up colors.

When the sites are moving, we would like the colors to behave in the following way. If the current coloring mode is 'random', keep the cone's color constant for each of the moving sites. If the color mode is based on image lookup, look up color from image for each frame. I wonder how it will look like, possibly very annoying especially for images with lots of sharp edges. Try running it on heavily filtered (smoothed) images.

3 Grading

Submit your code to the usual email address. The deadline is 3/15/2005, late penalty as usual. Send a tar file with a makefile and all your program files. We will untar (things should extract to the current directory), compile (type 'make') and run it like this: 'proj3 test_image.ppm'.

1. Voronoi diagrams look OK, 40

2. Animations OK 30

3. UI stable and robust 30.

4 New relevant OpenGL commands

`glOrtho`, `glColor*`, glVertex4*... that should be it! Perhaps `glViewport`.