Computational Geometry Project

Don Sheehy

February 22, 2010

1 The Goal

Implement and Visualize a geometric

- algorithm,
- construction,
- problem, or
- object.

2 Why?

In doing this project I hope you will

1. learn about a particular subarea of computational geometry in more depth,
2. teach each other something,
3. make something worthwhile, and
4. advance computational geometry, by providing clear learning aids for teachers and researchers around the world.

3 Output

- **The final output of the project should be suitable for viewing on the web.**
- Java applets are the preferred visualization, though videos, or even a gallery of stills may suffice for some projects.
- The code will be open source.
- You will use git for version control and will share the repository on **github**.
- You may work in groups of **up to 2 people**.
- Style Counts

4 Major concerns

You will have to figure out how to deal with the following 5 concerns.

1. Handling input
2. Handling output
3. Internal Data Stuctures
4. Internal and External interfaces
5. Numerical Precision

These 5 concerns represent the bulk of the uncertainty in the project. I recommend laying out a strategy for each as soon as possible. You should also identify which are fundamental to your particular project so you don't waste your time working on the wrong things.

5 Some possible Projects

1. **The Geometry of Failure** - Visualize the fractals produced by linear predicates on floating point numbers for different algorithms. This has been done before, but I think you can do better, producing higher resolution images.

2. **Visualize 4-dimensional configuration spaces** - Given some object with 4 degrees of freedom, the space of possible configurations is 4-dimensional. This can be visualized with line segments or with polytope projections.

3. **Delaunay/Voronoi Refinement** - A common method making Voronoi diagrams that have nice "fat" Voronoi cells is the add extra points called Steiner points. These refined Voronoi diagrams are very useful. Visualize the refinement process.

4. **Linear-time Centerpoints** - Visually explain the clever algorithm to compute centerpoints in linear time.

5. **Linear-time ham sandwich cuts** - Visually explain how to find ham sandwich cuts in linear time.

6. **Splitting a Delaunay Triangulation/Polytope** - Given a polytope or just a Delaunay triangulation with vertices colored blue and red, split polytope/triangulation into two by color in linear time. Show how this algorithm works.

7. **Graph Drawing** - Do something cool with graph drawing.

8. **Simplicial Depth in the plane** - The simplicial depth of a point x with respect to a point set P is the number of triangles with vertices in P that contain x. Visualize the depth contours for a given points set.

9. **Skip Quadtrees** - By combining skip lists and quadtrees, one can do a kind of dynamic point location. Visualize how this data structure behaves.

10. **Output sensitive convex hull** - If the convex hull of points in 2D has only h vertices, then it is possible to achieve running times of $O(n \log h)$. This is called an output-sensitive bound and it is achieved by Chan's algorithm. Show how this clever divide and conquer algorithm works.

11. **Visualizing flips and flip graphs** - Represent the flip graph of triangulations of a point set.

6 First Steps

By Friday, February 26, you need to do the following.

- Decide whether you will work with a partner and if so, who.
- Get `git` installed.
- Get an account at `github.com`.
- Create a repository for your project (one per group).
- Check in and push the repository to `github`.
- Send your github username and the name of your project repository to Don.
- Have some idea of what you *might* want to work on.