

3-D Visualization Tools for Phylogenetic Trees

My research involves inference of phylogenetic trees from gene sequence data. Such data are drawn from many different biological studies, especially the gene sequencing projects which have drawn so much press attention lately. Through alignment and tree reconstruction methods, researchers in this area finally arrive at phylogenetic trees that describe the process of evolution that relate the species under analysis. In one sense, these trees are a visualization tool (although they have many other uses). However, the task of visualizing these trees appropriately, intuitively, and meaningfully is a serious computer graphics matter.

I propose the following project for Project 4: I will produce two versions of a 3-D visualization tool that expressly aims to allow visualization of the gigantic tree/graph datasets that are produced in cutting edge bioinformatics studies. While visualizing small trees is straightforward, it is still an open question how we can visualize enormous tree/graph datasets, especially as the scope of such evolutionary histories approach the scale of the “tree of life” - the single graph describing the evolutionary history of all organisms (a major “Holy Grail”, if you will, of phylogenetic studies).

The scope of the project will follow the Ph.D. thesis work of Dr. Tamara Munzner at the Stanford Graphics Lab, now a professor at the University of British Columbia; the name of her project is H3: Hyperbolic Quasi-Hierarchical Graphs. (Related work is discussed by University of Bergen student Tim Hughes' overview, located at <http://www.iu.uib.no/~tim/treesPage.html>.)

1. Produce a 3-D Euclidean space visualization tool. Note that Hughes indicates current tools cannot handle large datasets. Thus, this tool will be largely unuseable, and will mainly be used to demonstrate the inability of Euclidean space visualization schemes to effectively show large trees - in this vein, it will be used solely for comparison against the main project goal of producing a hyperbolic space visualization tool for such large trees. This will be the most basic "base project". Really this will be the most basic first step - read in the tree file format, and then use OpenGL calls to display GL_POINTS and GL_LINES to show the tree in standard Cartesian space. No navigation ability will initially be provided.
 1. As a bell/whistle if I have time, I will try to add mouse button 1 rotation ability to at least look around the tree.
2. Produce a 3-D hyperbolic space visualization tool in the vein of H3 as outlined in Tamara Munzner's Ph.D. thesis located at: http://graphics.stanford.edu/papers/munzner_thesis/. A similar tool that is more recent and publicly available is Walrus: <http://www.caida.org/tools/visualization/walrus/gallery1/>. I am unsure of how far I along I can get in this portion of the project since I think there is a lot to be done here – several people have written their doctoral dissertations on the topic of large tree 3-D visualization schemes. Several key components are:
 1. Hyperbolic space visualization. According to Dr. Munzner, her thesis has two main

algorithmic contributions, the first of which are covered by the following two features:

1. Two pass hyperbolic space tree arrangement layout algorithm. See Munzner's thesis p. 34-40. This operates in two passes: the first bottom-up pass attempts to estimate the geometric space required to draw each node so that it doesn't impinge on the space of other nodes in the tree, and the second top-down pass calculates the placement of each node using a sphere packing algorithm. An important calculation needed afterwards is the projection from infinite hyperbolic space coordinates to finite Euclidean projection coordinates for display (this is a neat trick – I will talk about this during my presentation). Note that this will be the main focus of this entire project - I will try to understand and implement the calculations required for just displaying a tree in hyperbolic space.
2. As a bell/whistle, I will try to implement the adaptive drawing algorithm for interactive navigation of the tree. See Munzner's thesis p.43-44 for details. Here, the algorithm uses priority queues to process nodes within the time allotted by frame rates, so that navigation of the tree is interactive. This will be a bell/whistle - I will only add this if I have time.
2. “Crystal ball” method of mouse control for changing camera in the program – just like the “debug window” controls of project 2’s ray tracer.
 1. As a required feature, I will implement the rotate feature with the first mouse button.
 2. A bell/whistle will be adding translation/zooming, picking, etc.

Bells and whistles include:

3. Picking nodes to examine attributes.
4. Manual subtree expansion/collapse.
5. Efficient realtime OpenGL rendering with no frame rate issues.
6. Node labeling.
7. Adaptive drawing algorithm to guarantee interactive navigation of trees with no framerate issues: see part 1.2 above.

To recap, the specific project requirements (required for an A???) are:

- First step: read in the tree file format, and then use OpenGL to display `GL_POINTS` and `GL_LINES` to show the tree in standard Cartesian coordinates. No navigation will be provided. This becomes a "reference" executable to show that displaying huge trees in Euclidean space doesn't work.
- Then, take that executable, and use Munzner's two-pass algorithm to instead place each node object into hyperbolic space with coordinates (r, ϕ, θ) . Use a projection calculation to convert from the hyperbolic coordinates (r, ϕ, θ) to Euclidean coordinates (x, y, z) and display in OpenGL using these Euclidean coordinates.

- Implement the rotate feature with mouse button 1 in Euclidean space.

Thus, the main goal will be to understand Munzner's two pass hyperbolic space tree layout algorithm, and the calculation to project from hyperbolic space coordinates to Euclidean space coordinates. (I understand Munzner's two pass hyperbolic space tree layout algorithm ok now, but I need to do more research on the projection from hyperbolic space to Euclidean space - this is pretty standard though.)

Anything else will be a bell/whistle.