Visualizing the Tree of Life

New Mexico

Supercomputing Challenge

Final Report

April 4, 2012

Team 102

Pinon Elementary

Los Alamos, NM

Team Members:

Ryan Swart

Teacher:

Mrs. Chrien

Project Mentor:

Pieter Swart

Executive Summary

The Tree of Life is the classification of all living things on Earth. It is important to understand the *Tree of Life* because it explains much of how life came to be, and it is useful to understand to preserve vanishing species, many of which we do not know. Because of its size it is impossible to understand unless we can find new ways to visualize it.

My goal was to write a program to visualize any tree in 3D, allowing one to zoom in and out and to view details, thereby providing a better understanding of a very complex data set. I developed a series of programs in the Processing language that I later combined, and then used several libraries to interactively visualize my tree in 3D. An important component in the process of designing accurate programs is that we created test trees, so one can immediately see any mistakes in the code.

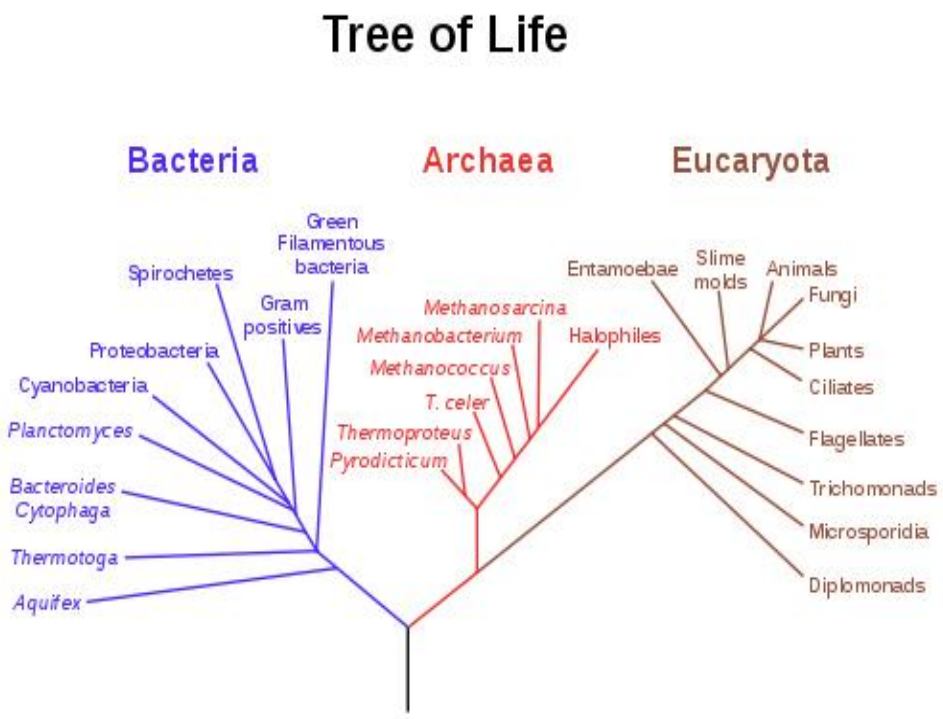
This software should also be useful to help us understand other huge datasets, such as the connection between different countries or between different stocks.

Once our program was complete, we found that rotating and being able to magnify the trees created a better understanding of the complex datasets.

Background

The tree of life is the classification of all living things on Earth. It is important to understand the *Tree of Life* because it explains much of how life came to be, and it is useful to understand to preserve vanishing species, many of which we do not know. Carolus Linnaeus (1707 – 1778) was one of the first people to start classifying animals similarly to the tree of life. There currently is 1.7 million species known to man. With bacteria and archaes, there may have been/is as many as 400,000,000 species. More than 15,000 new species are discovered per year in the field by scientists.

Why is this important? It is useful to understand huge datasets such as complex ecosystems. It can help predict what and when species are going extinct. It might help discover new medicines and antidotes. It is far too complex to publish in a book.



A crude visualization of the tree of life

Carolus Linnaeus, as depicted on the 100 Kronor note of Sweden

Problem Statement

My goal was to write a program to visualize any tree in 3D, allowing one to zoom in and out and to view details, thereby providing a better understanding of large and complex data sets. I developed a series of programs in the Processing language that I later combined, and then used several libraries to interactively visualize my tree in 3D. A problem with visualization software is that testing is more difficult. An important component in the process of designing accurate programs is adding test cases, so that one can immediately see any mistakes in our code.

Description of methods

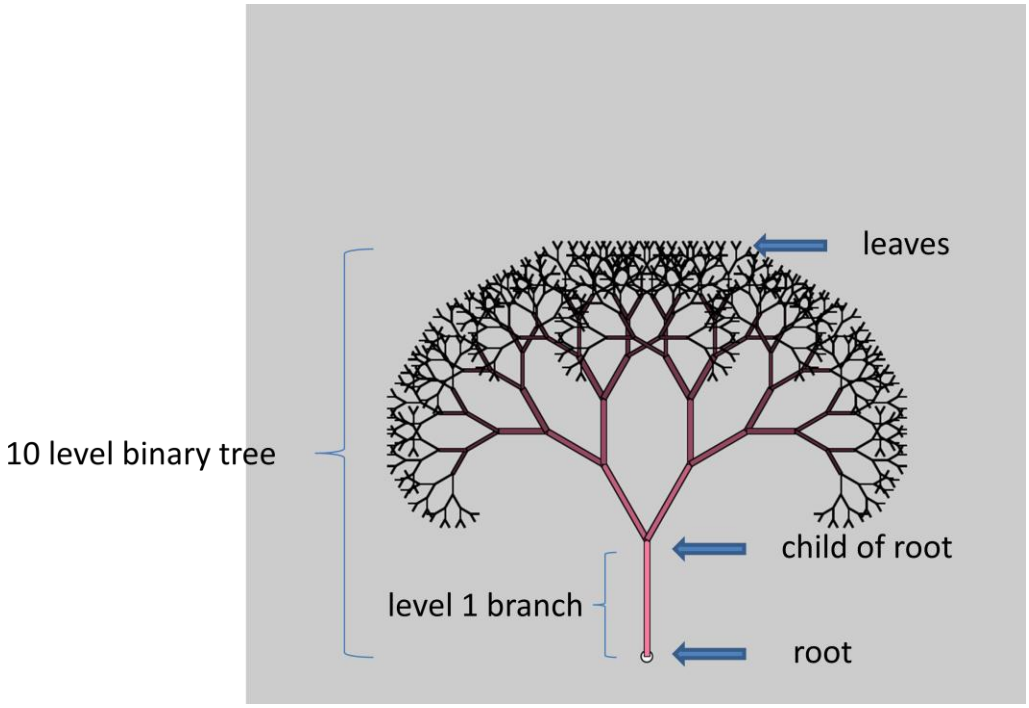
Using the Processing language, we developed a sequence of algorithms to analyze the input file describing a general tree. Using special data-structures designed for this purpose, the tree is then drawn in 3D with the correct text labels. It is important that branches do not overlap and hide information. My goal was to visualize any tree in 3D, allowing one to zoom in and out and to view details, thereby providing a better understanding and avoid messy details. I used several Processing libraries to interactively visualize my tree in 3D. An important component in the process of designing accurate programs is testing and debugging. I created several small example test trees, so one can immediately see if there are any mistakes in our code. Colors and thickness is used to better visualize the information.

Trees

Trees are important data structures in programming and in organizing complex data sets. In addition to the Tree of Life, well-known examples include;

- the Yahoo subject index,
- the Dewey Decimal System, and
- playoffs brackets (for example, chess, Mathcounts, and baseball).

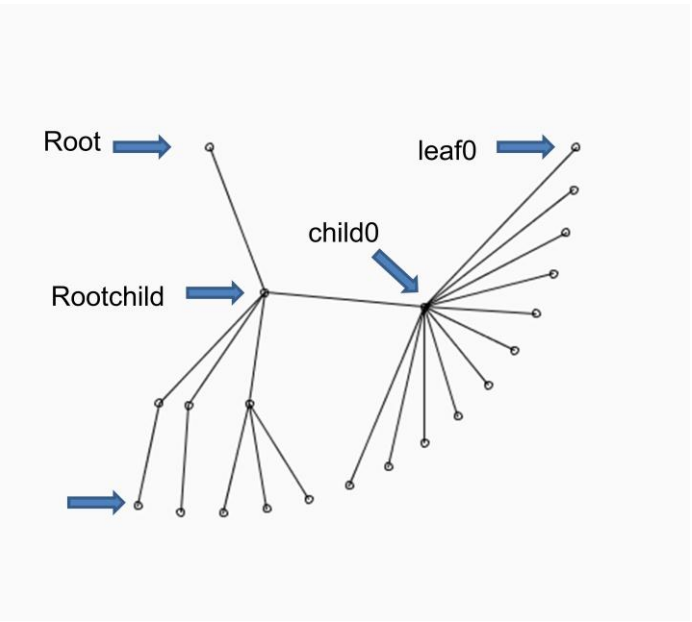
The next figure shows the terminology that we are using. Our trees have “branches” connected at “nodes”. We are working with a special subclass of trees, known as *rooted, labeled trees*. A special node is called the root of the tree, and is usually at the center of all our trees. The nodes furthest from the root are known as the leaves.



The basic terminology of a tree that used

In the trees studied here, each node has a text label. In the tree of life, for example, the label on each leaf node is the species. For each tree all my programs read as input a text file, listing each branch (as shown below.) The first line is always a comment, usually identifying and describing the tree. The first node on the second line is always the root. The input file can be of any length.

```
// testing testing tree
Root,Rootchild
Rootchild,child0
Rootchild,child1
Rootchild,child2
Rootchild,child3
child0,leaf0
child0,leaf1
child0,leaf2
child0,leaf3
child0,leaf4
child0,leaf5
child0,leaf6
child0,leaf7
child0,leaf8
child0,leaf9
child0,leaf10
child1,leaf11
child1,leaf12
child1,leaf13
child2,leaf14
child3,leaf15
```



Small example tree with its input file. The "Root" node is at level 0, the "rootchild" node is at level 1, and so on.

Tree drawing algorithm

1) Read the input file.

2) Convert text to numbers using a Java HashMap data-structure. After that we need only work with integer labels.

3) Load branch info for each branch into a branch class.

4) Determine the levels for each branch and node.

5) Calculate the children of each node.

6) First draw the leaves in a predefined circle segment..

7) Work backwards to the root. We use a function called "kickback" to find the position of each parent from its children. See Figure and description below.

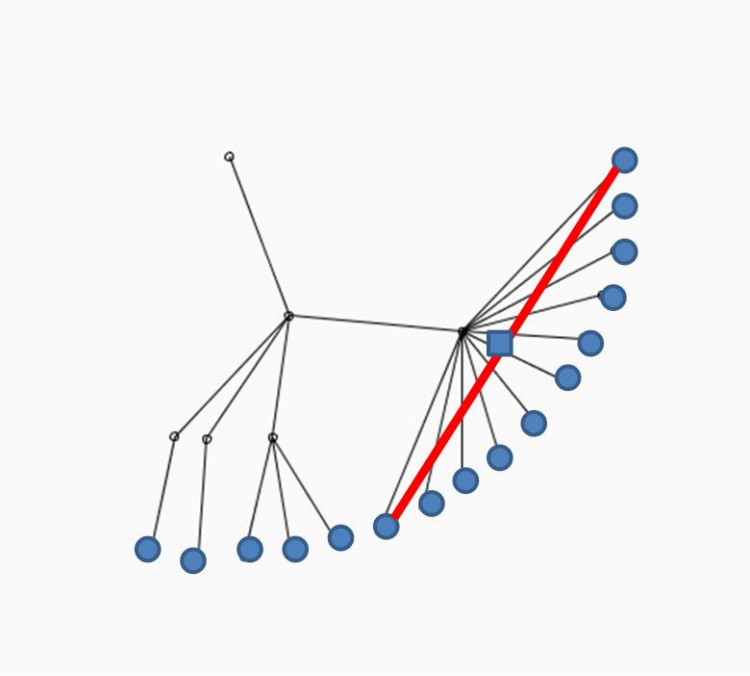


Illustration of the kickback function. Step 1: calculate the average center of the leaves. The blue square(the parent node) is placed at the average center.

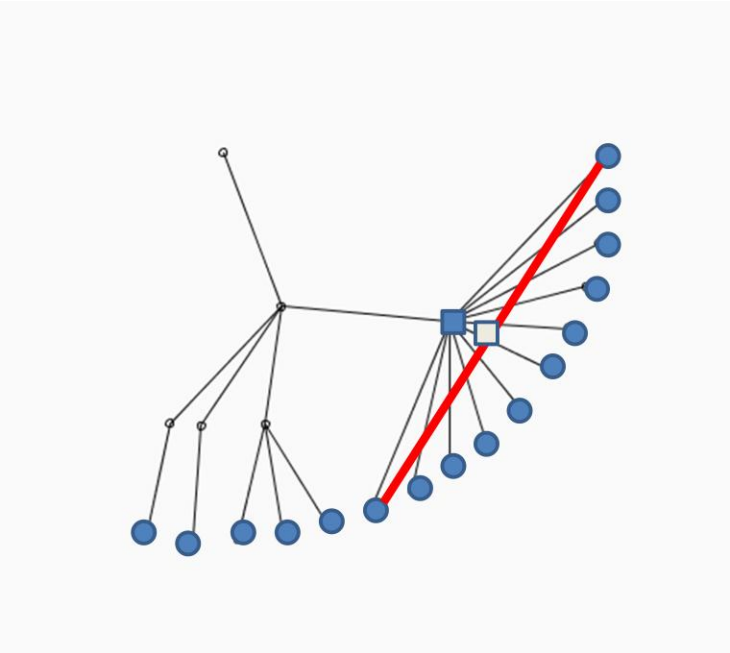
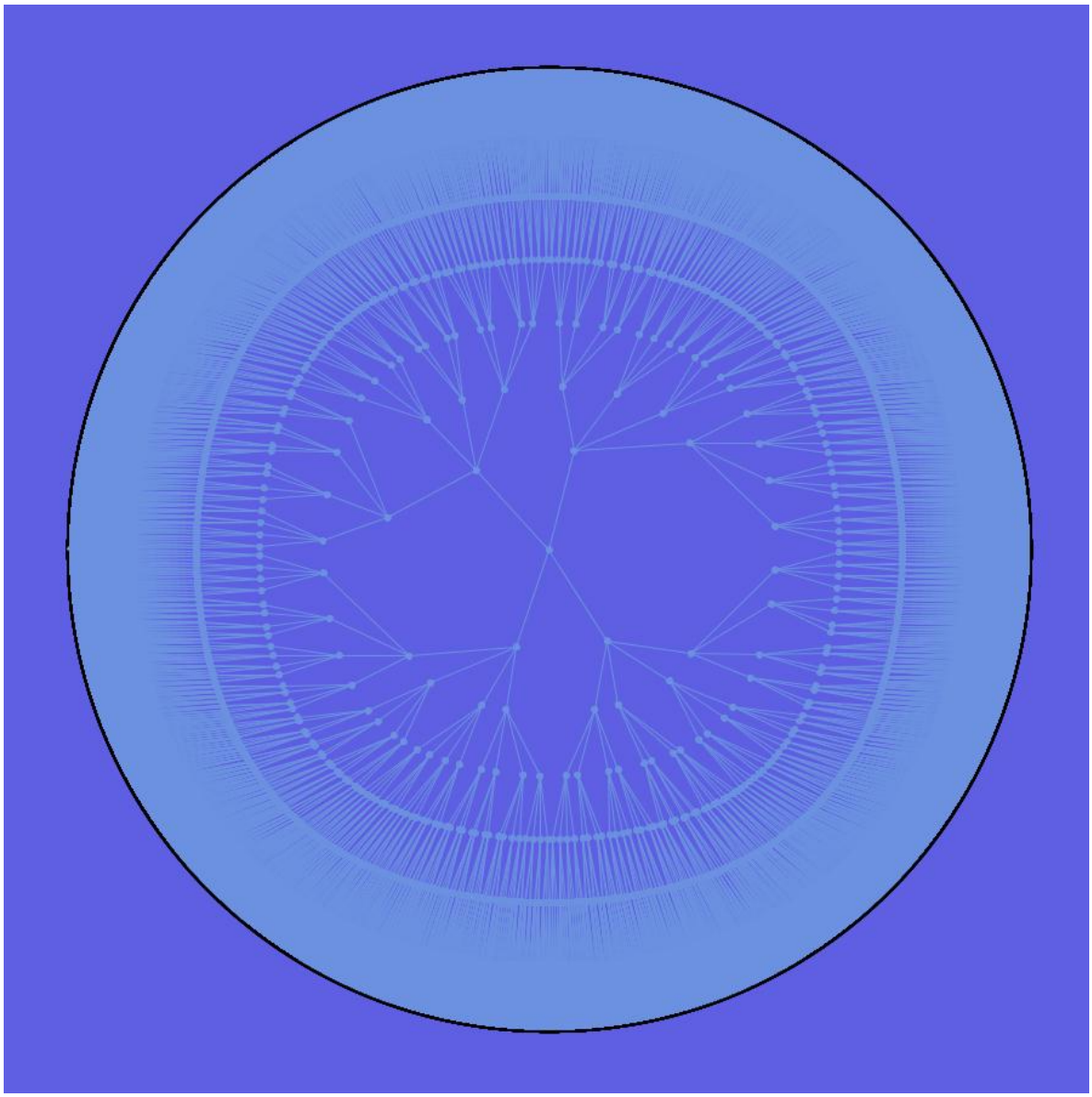


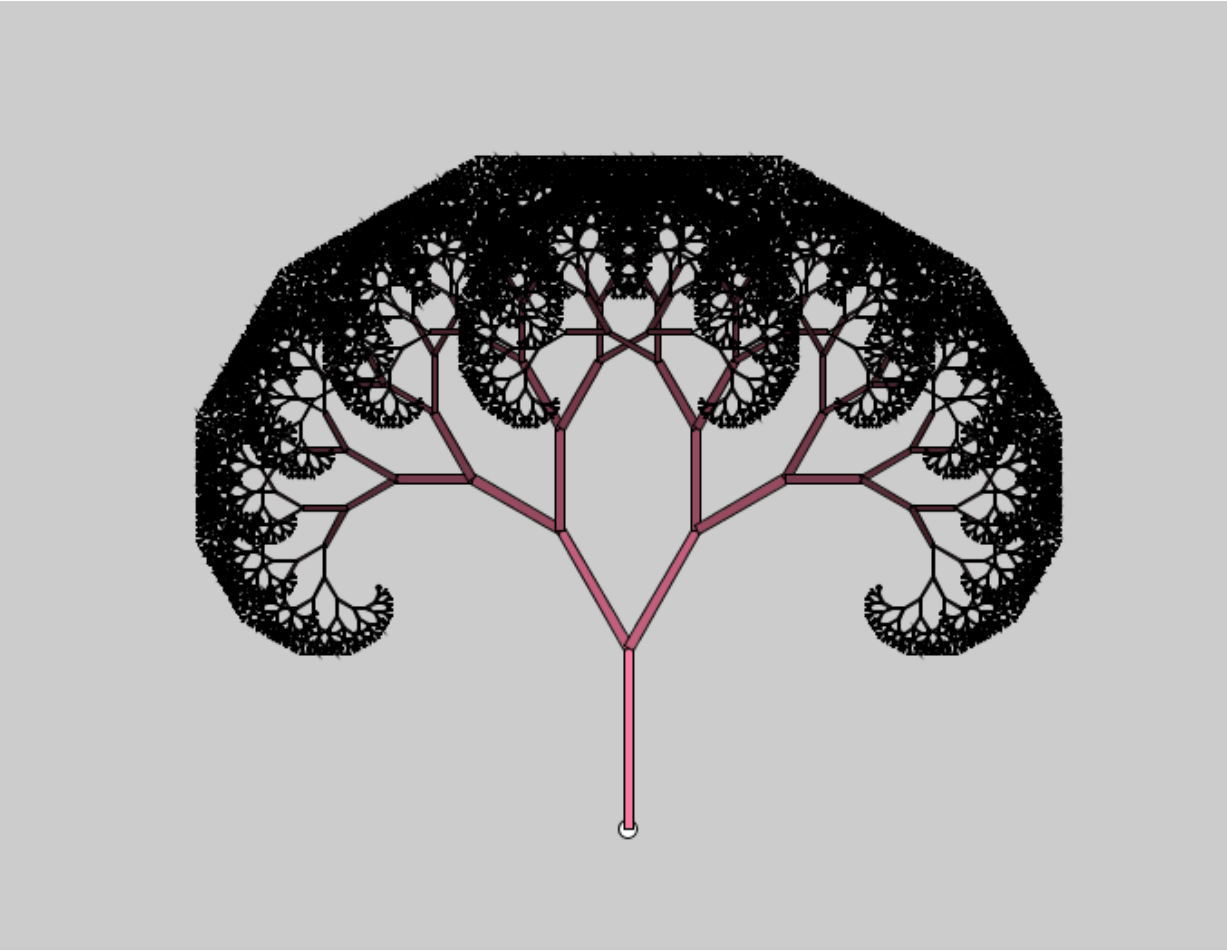
Illustration of the kickback function. Step 2: Kickback, now we take the square (average center), and move("kick") it back to a (second) invisible circle, toward the root node.

I then used several powerful visualization libraries, including: OpenGL, Shapes3D, and PeasyCam. OpenGL allows smoother lines in 3D. Shapes3D allows more 3D objects. PeasyCam allows you to zoom in and out and rotate.

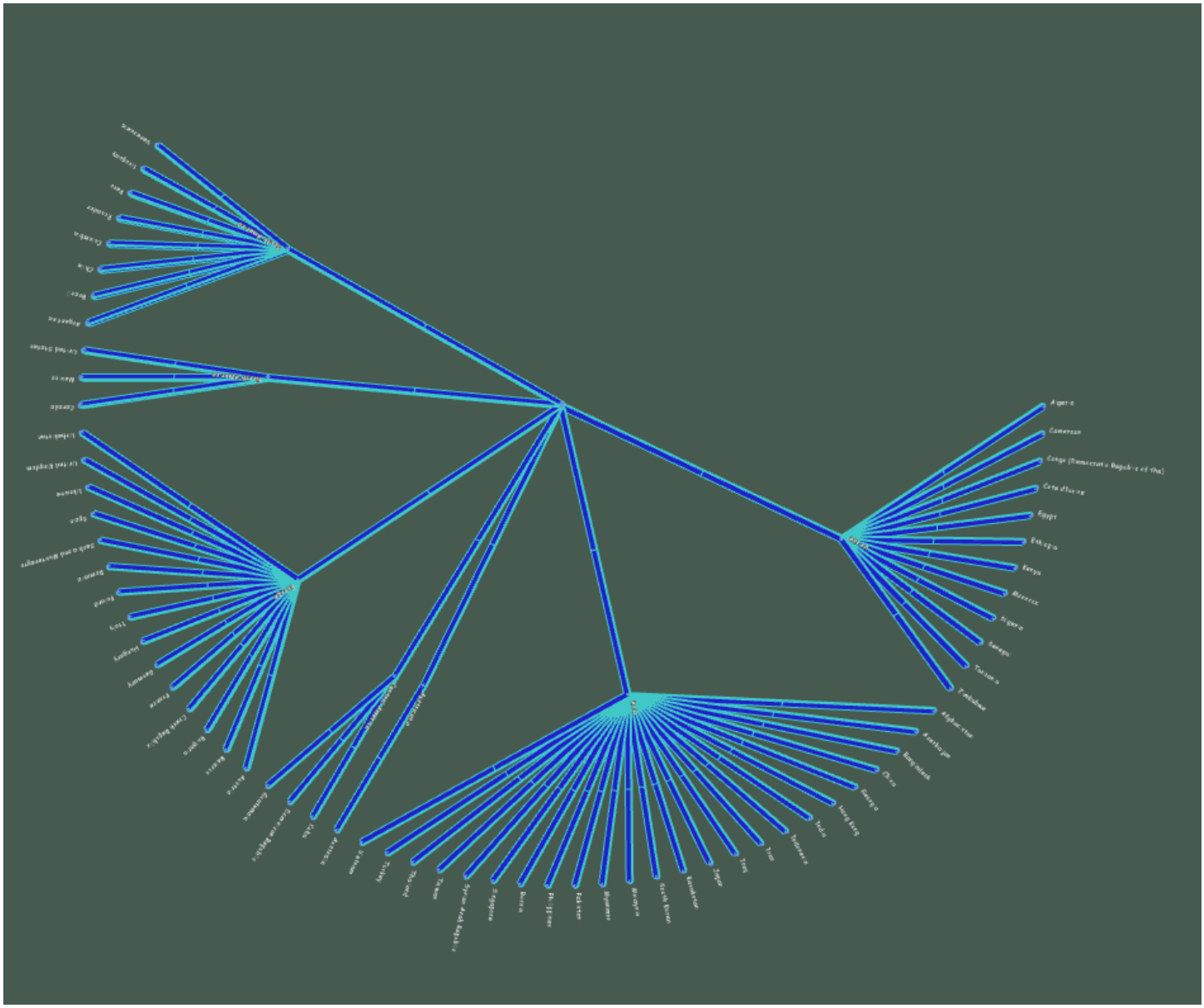
Results



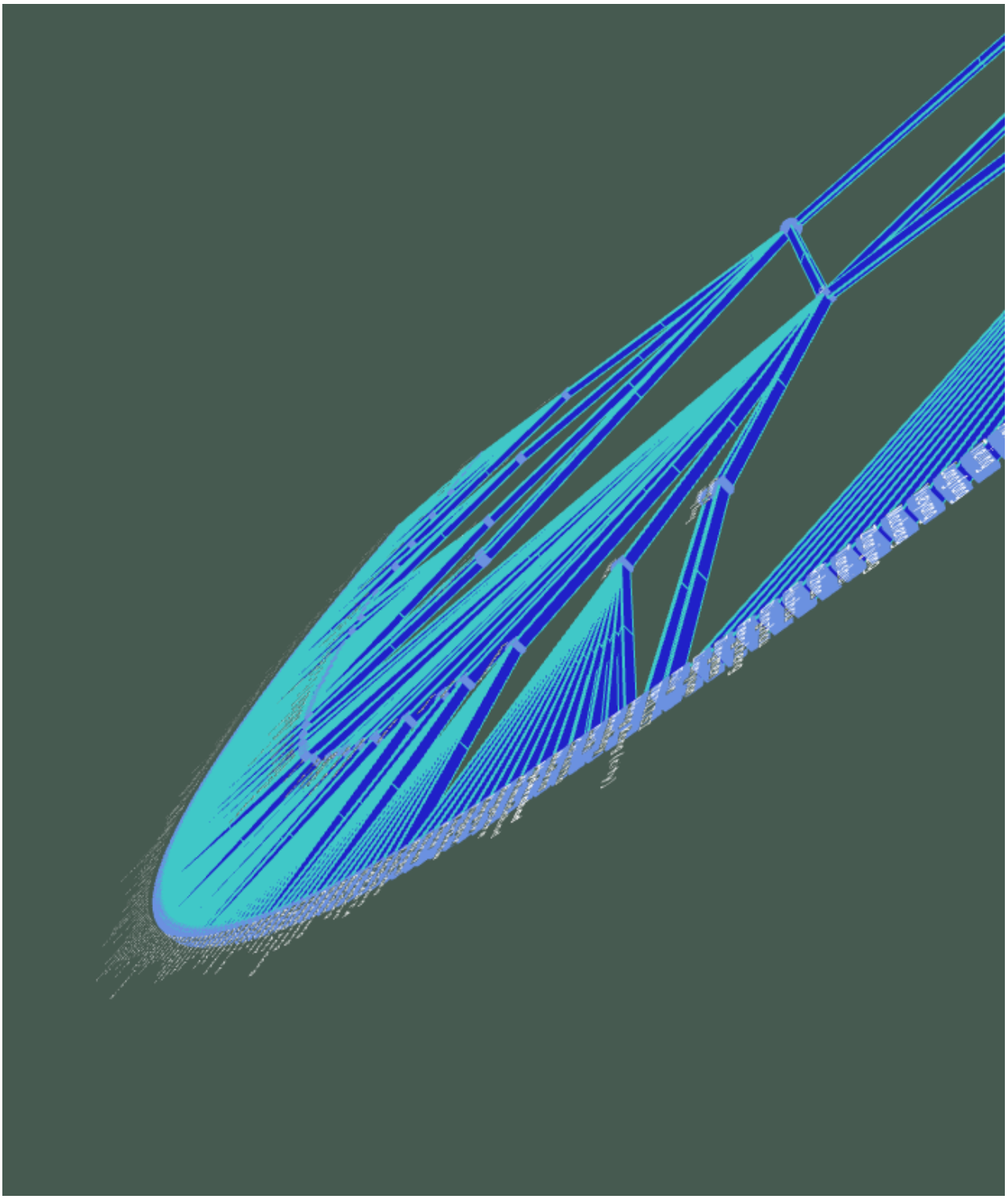
A “powers of four” tree



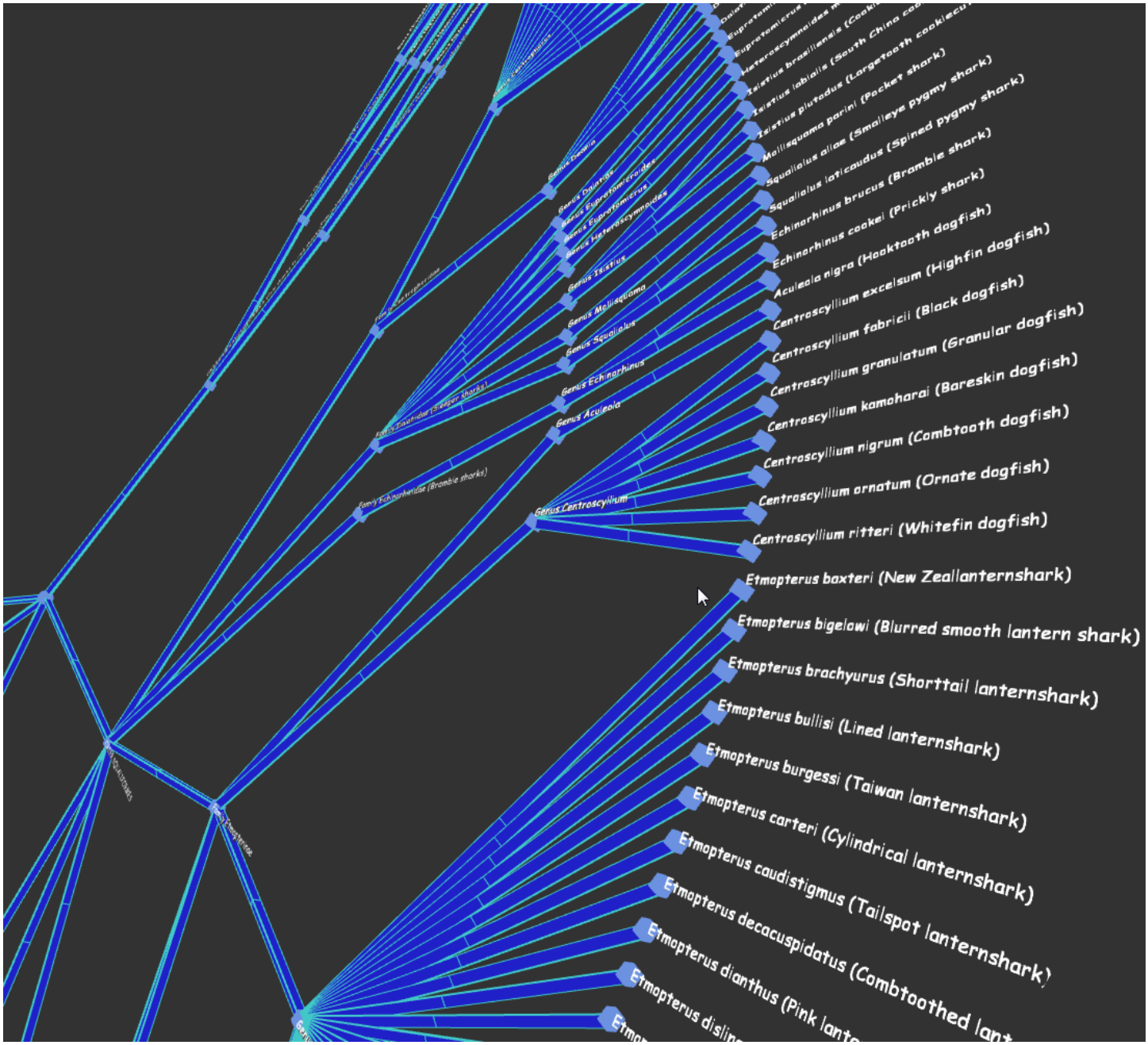
A 14 level binary tree with 8,224 branches drawn using a branching process. Here symmetry is preserved but branches overlap.



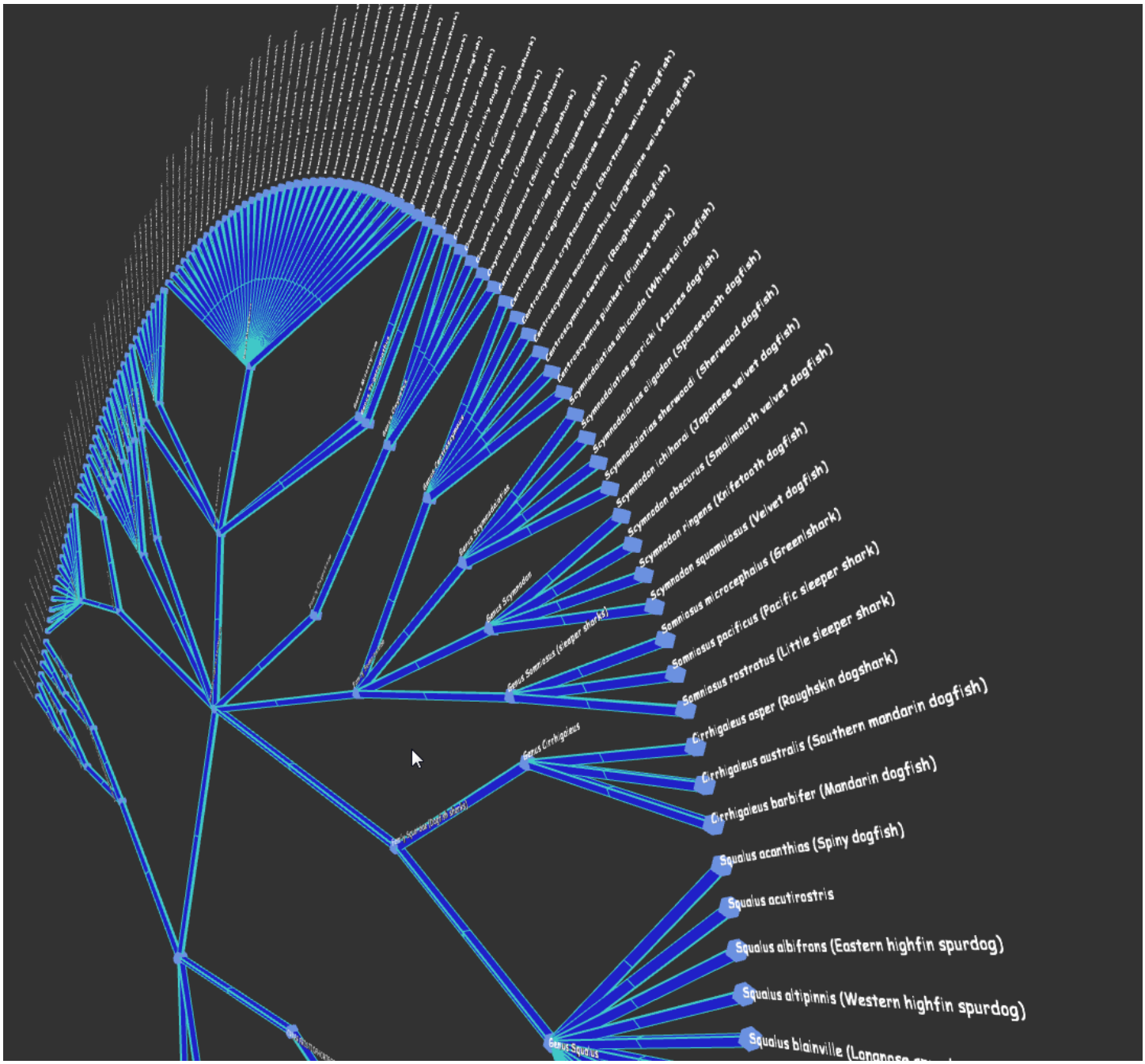
A tree of all countries with cities containing more than 1M people (a shortened version of the cities tree)



Side view of tree of all cities with population greater than 1M



The tree of all sharks, from the Tree of Life (my first Tree of Life drawn tree)



The tree of all sharks (different view), from the Tree of Life

Conclusions

I conclude (from a lot of trees drawn) that yes, it is possible to effectively visualize large, complex trees. I have not only created a program that can just draw one tree, I have created a program that can draw a large variety of trees. The ability to rotate and zoom in allows one to observe a specific part of the tree, and thereby help us better understand that section. The next major challenge is importing more data for the Tree of Life.

References

Reas, Casey, and Ben Fry. *Processing: A Programming Handbook for Visual Designers and Artists*. Cambridge, MA: MIT, 2007.

Migdalski, Edward C., George S. Fichter, and Norman Weaver. *The Fresh & Salt Water Fishes of the World*. New York: Greenwich House, 1983.

Shiffman, Daniel. *Learning Processing: A Beginner's Guide to Programming Images, Animation, and Interaction*. Amsterdam: Morgan Kaufmann/Elsevier, 2008.

Tudge, Colin. *The Variety of Life: A Survey and a Celebration of All the Creatures That Have Ever Lived*. London: Oxford UP, 2000.

processing.org (December 4, 2011)

http://www.openprocessing.org/ (December 4, 2011)

http://tolweb.org/tree/ (December 4, 2011)

Most Significant Original Achievement

My most significant original achievement is my program’s capability to draw trees. My program can draw any tree where all branches are of the same length (given enough memory, and that the input file is in the right format). There are other pieces of code that can draw trees, but those programs are designed to draw specific trees or draw small trees. My program is also special because it prints information about the structure of the tree.

Acknowledgements

I would like to thank:

Pieter Swart for:

- Helping me with my reports and Tree of Life data
- Giving me inspiration
- Helping me when I get stuck

My teachers Mrs. Chrien and Mrs. Hermes.

My sponsor Mrs. Selvage

All the people that organized the supercomputing challenge

All the judges, for their advice and feedback

Software

I used the Processing language, designed by Casey Reas & Ben Fry, in 2001, at MIT. Processing is a visual, interactive language that borrows from Java. The code is open source and available for many architectures at processing.org. Processing is becoming very popular with artists and graphic designers for visualizations. There are many libraries and books available, and I learned most of my skills from openprocessing.org. It is outstanding for animation, which is why I am using it. I also used several powerful libraries, including: OpenGL, Shapes3D, and PeasyCam. OpenGL allows smoother lines in 3D. Shapes3D allows more 3D objects. PeasyCam allows you to zoom in and out and rotate. My program works great except that when animating a tree that requires more than 2Gb causes it to eventually crash because of a memory leak, supposedly because of some strange interaction between the Shapes3D and PeasyCam libraries.

Source Code

Ryan Swart New Mexico Supercomputing challenge
2012, final report
This code draws trees with labels in 3D

(attached separately)