GPU Accelerated Method for Constructing and Rendering Trees

-

Progress Report

Thomas Mcloughlin

registration: 100203952

1 Description of the Project

This section will outline the aims and motivation of the project, describing why this system would be useful in a 3D graphics environment application.

1.1 Aims

The aim of this project is to produce an OpenGL module for constructing and rendering trees that can be placed in a 3D environment. These trees should be randomly generated to allow for multiple trees to be rendered and not have them look the same. They should use a suitable algorithm to produce this randomness that will control the structure of branches and leaf placement on those branches.

1.2 Motivation

The motivation for producing this system is to reduce the effort that needs to be dedicated to creating a realistic looking 3D environment. Simulating nature can be difficult and that is shown in the construction of trees and other foliage. Having a designer create each tree model from scratch while trying to make each model look realistic and varied would be a huge undertaking and take a long time. Whereas using this system would allow for quick and easy construction and placement of trees in an environment, with user input allowing for certain aspects of the generated trees to be tweaked to the required specifications.

2 Description and Understanding of Issues and Problems

In this section the various issues and problems that have arisen during the design stage of the project will be listed and explained.

2.1 Branch Structure

The choice of what method to use for branch structure when approaching this project is the issue which will have the largest affect on the result. Different algorithms are likely

to produce different results in the final branch structure so care was put in to understand the various methods and choose one that would be most suitable with a balance between attaining a realistic looking result and ease of implementation.

The choices were separated between methods described in various papers found as part of the Literature Review and after much consideration the decision was made to use L-systems, described by Prusinkiewicz et al. (1996), to create the branch structure in the trees. This would be done by creating preset constructed parts to act as our variables, using a chosen point in the environment as an axiom and providing the rules to construct the tree geometry. Some variation at each level needs to be applied to give a sense of natural growth, this would include a variation in branching angle each time a branch splits off from the previous branch. The user control can be presented through the rules of the L-system, by changing these rules the user could affect how the tree is constructed.

2.2 Leaf Placement

The problem with leaf placement is having the system decide where will be appropriate to place leaves to make the tree realistic looking. Prusinkiewicz et al. (1996) combine leaf placement with the construction of the branches, where some variables in the chosen L-system include adding leaves to the constructed branch or having a terminating branch end in a leaf. This is decidedly too basic for use with a more detailed 3D application and so this idea will be combined with a method presented by Weber and Penn (1995) where leaf placement can be decided based on the level of a branch. The level being the number of parental branches that the current branch has meaning the more parents the branch has usually means that the branch is further from the trunk and more likely to be a terminating branch. Leaf placement will be done only on branches of a certain level and above to give the trees the proper dispersion of leaves across outwards reaching branches.

The issue of how specifically to add leaves to the chosen branches requires another method. Weber and Penn (1995) do describe their method for leaf placement somewhat but only give a formula for deciding the number of leaves to be placed and not specifically how they are placed. For this a method of random placement needs to be inferred and the chosen method involves these steps: decide the number of leaves needed for a chosen branch, choose varying points along the branch where the leaves will be placed,

render the leaves, orient the leaves semi-randomly while taking into account the position of the tree.

3 Design and Planning

As a simulation project this design plan will give a design of the chosen simulation model supported by example pseudo code and diagrams.

The chosen simulation model for this project is the use of L-systems which has been loosely explained but will be described in more detail here.

A Lindenmayer system (L-system) is a system of ordering an alphabet of symbols to make strings. The symbols are formed into the string using production rules that expand each symbol into some larger string of symbols. The expansion starts from an initial axiom string which is some chosen collection of symbols, changing the axiom string is a way to alter the resulting string without changing any of the symbols or rules involved in the system. The number of times you apply the rules to the string (number of recursions) will also give you different strings, and a certain recursion depth may be chosen to give a desired affect.

Prusinkiewicz et al. (1996) outlined multiple examples of using L-systems and below is a simplified explanantion of one of them and how it could be modified to correspond to this project.

For the production of trees with some user input required we would want to use parametric L-systems. The example explained below is that of a deterministic L-system (D0L-system) which would produce the same result every time it was ran. This is not what we want for the context of randomly generating trees so this method would likely be altered to become semi-deterministic with certain randomness in generation being added into the construction process.

Considering the recognisable snowflake fractal we can define the multiple recursions of its formation using a D0L-system with this Axiom and the given Production:

Axiom ω: $F(1) - (120)F(1) - (120)F(1)$
Production p_1: $F(s) \rightarrow F(s/3) + (60)F(s/3) - (120)F(s/3) + (60)F(s/3)$

The axiom draws an equilateral triangle with edges of unit length. Production p_1 replaces each line segment with a polygonal shape where the line segment has been given a convex triangular appendage. The Production p_1 can be visualised as so:

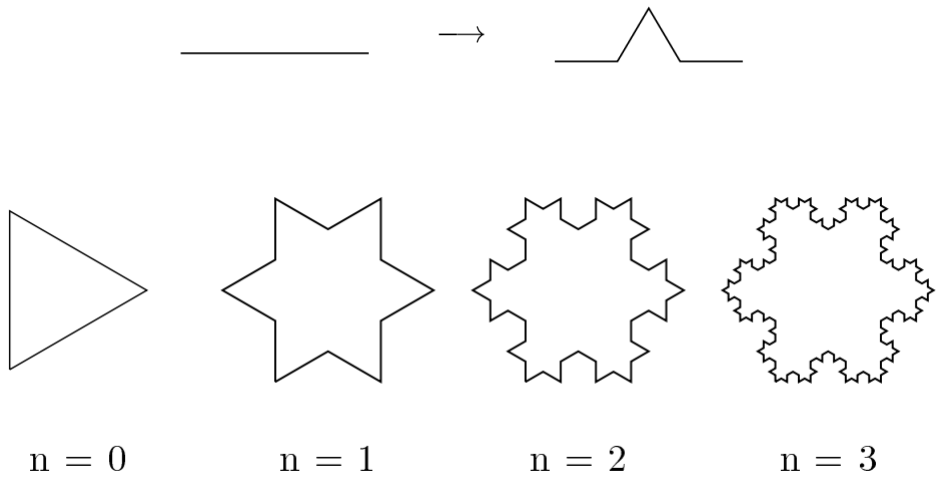

Figure 1: The axiom and first 3 recursions defined by the snowflake D0L-system from Prusinkiewicz et al. (1996) pg.12

Figure 1 gives a basic example providing the foundation to elaborate on the planned design for the tree construction of this project. A semi-deterministic D0L-system will be produced that will allow for the generation of a general branching structure. This would be done by assigning 3D line drawing to the symbols in our L-system that would be formed into a string. This could be done by building from a root point and expanding the tree branches upwards, or by starting with a central trunk and separating it into branches in a similar way to how the triangular snowflake L-system works.

The chosen symbols would represent differing thicknesses of branches as the tree expands from the central trunk and some symbols would include the requirement for leafs to be placed, at which point the chosen leaf placement method would be used to complete the branch.

Once an acceptable default for the D0L-system has been achieved, the system can be parameterised to allow for user alterations of the tree generation which could include facets such as average branch length, number of branch splits, leaf density, tree height and width etc.

To add some random variance to the appearance of the generated trees there can be some semi controlled symbols that will only slightly alter the trees overall appearance. This could include a variation in branching angle to prevent each branch from being perfectly angled from it's parent. This symbol could also be parameterised to allow user control by letting there be a choice over what range this angle deviates.

The issue of adding thickness to the tree branches can also be combined with the L-system implementation. Below are some examples from another parametric L-system explained by Prusinkiewicz et al. (1996)

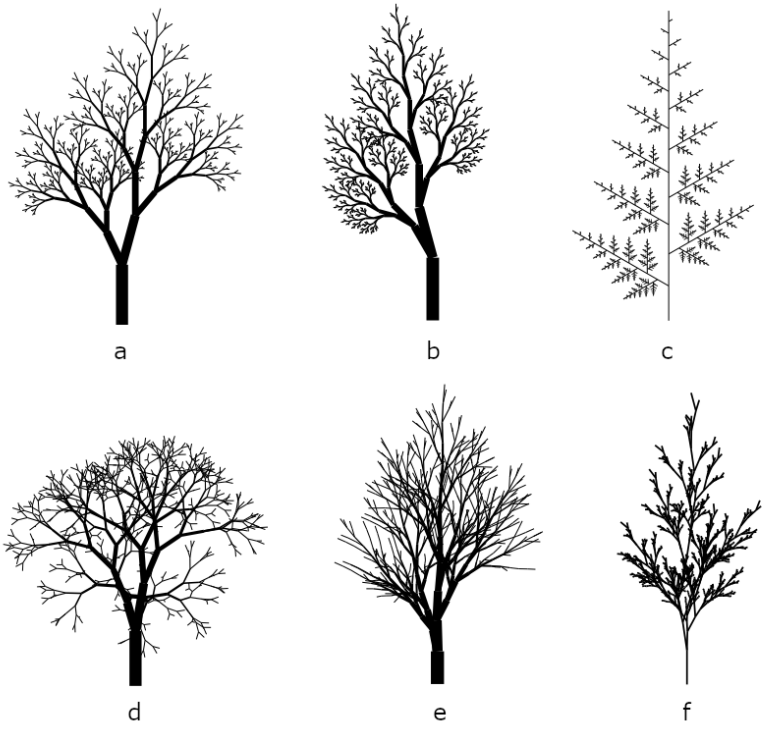

Figure 2: Some example tree constructions showing the results of a parametric D0L-system, altered from Prusinkiewicz et al. (1996) pg.14

The trees in Figure 2 are generated using another D0L-system that includes a parameter to increase the width of branch segments over each recursion which can be seen most apparently in the trees a, b, d and e with their wider trunks and much thinner terminating branches. This same method could be applied to this project by using cylinders for each branch segment instead of line drawing. This would also allow for easier application of texturing to each branch as they will all be formed using the same shape.

The implementation of the project aims will be as follows. Producing an OpenGL module that can be used to insert the tree models into any OpenGL environment. This would consist of a separate "tree" struct that would be given parameters to create the tree, these parameters would include:

- Global x, y, z coordinates for the positioning of the tree, the point given being the base of the tree trunk.
- Maximum tree height and width, to limit how far the branch structure will spread before being pruned.
- The angles between parent and child branches along with the range of variance of those angles.
- The length and thickness of branches along with constants to determine the gradual decrease in length and thickness for each recursion.
- The minimum length for a viable branch which will serve as a cut off point for the recursion algorithm.
- Finally a parameter for directly controlling recursion depth in case the previous parameter does not give the desired results.

After the research carried out in the literature review the decision has been made to change some aspects of the chosen approach, the most impactful of which has been the method of branch structure. The initial plan for branch construction was to follow the method presented by Prusinkiewicz et al. (2007) to produce a line-drawn tree skeleton and then apply the method of generalised cylinders described by Bloomenthal (1985).

After further research into the area of L-systems the decision has been made to forgo using these methods with the benefit of being able to represent the skeleton and branch thickness of the tree within a single L-system as described above. This will still produce acceptable looking tree models while reducing the overall complexity of the process, meaning that the construction and rendering process will be less resource intensive and therefore make real-time applications more stable.

References

Bloomenthal, J. (1985). Modeling the mighty maple. *ACM SIGGRAPH Computer Graphics*, 19(3):305–311.

Prusinkiewicz, P., Hammel, M., Hanan, J., and Mech, R. (1996). L-systems: from the theory to visual models of plants. In *Proceedings of the 2nd CSIRO Symposium on Computational Challenges in Life Sciences*, volume 3, pages 1–32. Citeseer.

Prusinkiewicz, P., Lane, B., and Runions, A. (2007). Modeling trees with a space colonization algorithm. *NPH*, 7:63–70.

Weber, J. and Penn, J. (1995). Creation and rendering of realistic trees. In *Proceedings of the 22nd Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '95, page 119–128, New York, NY, USA. Association for Computing Machinery.