Leopold-Franzens-Universität Innsbruck

Institute of Computer Science
Interactive Graphics and Simulation Group

Geometric Modeling Project

Advanced Computer Graphics WS14
Documentation

Phillip Mildenberger
Stefan Spiss
Cem Okulmus

advised by
Savoye Yann Pierre, PhD

Innsbruck, January 22, 2015

1 Introduction

In this Project, the goal was to use procedural Methods to generate meshes. The idea was to generate a landscape and adding different plants to it. For the generation of procedural plants, L-Systems were used. The landscape generation is based on the erosion of streams. Unfortunately the landscape generation was more time consuming then expected. So we couldn't implement the full procedural landscape as we originally wanted.

2 Procedural plants using L-Systems and Turtle Graphics visualisation

Implemented by: Cem Okulmus

2.1 Technical Overview

The implementation consists of two parts. One is the "LSystem" class, which is responsible for importing a set of rules that define the L-System and consequently generating new strings that represent a word from the language the L-System defined. This is done by applying a formal rewriting system, the nature of which depends on the actual type of the L-System. The end result is a string that contains simple commands on how to "draw" a three-dimensional mesh.
The second part is the generation of a three-dimensional mesh of the wanted plant. This is done by the well known method of turtle graphics. This concept was explained in detail in the lecture, so only a concise summary will be given. The idea is that the "turtle" follows the list of commands sequentially and generates the mesh as it moves through space. To accommodate branching structures, as they are common in nature, a state stack is introduced, a state being the exact position and direction of the "turtle".

2.2 Implementation Detail

2.2.1 L-System types

Three basic types of L-Systems were implemented. Each of them is an extension of the former, so the L-Systems accepted by the first, would also be accepted by the second and third, and so forth. All L-Systems, like all formal languages, define an alphabet. Additionally L-Systems have a set for variables, where variables are a subset of the alphabet. The variables are the symbols which will be replaced in the production rules.

Deterministic and Context-free L-System
: This type of L-System is also known as D0L in literature. All its production rules must define a single symbol to replace, since it is context-free and a string of its alphabet to replace it with. Depending on how often this is iterated, we will get a different word of the language, assuming a fixed starting symbol.

Stochastic and Context-free L-System
: This extends the previous type by allowing multiple rules for the same variable. These will be treated as possible applications. Their probability is uniformly distributed. (E.g.: Three rules means a probability of 1/3 for each of them)

Stochastic and Context-sensitive L-System

This extends the previous type by a, possibly empty, environment, for each variable. This restricts the rule, and will only apply it if the environment can be matched. Matches with multiple applicable rules are resolved by choosing the longest match, then the prior rule in the L-System definition, similar to an LR-Parser.

2.2.2 Three-dimensional Turtle Graphics

The "_3DTurtle" class is very straightforward implementation of a turtle graphics renderer in three-dimensions. The operations on meshes were implemented using OpenMesh[1], which made it very easy to add an arbitrary amount of faces to the final geometry. The needed calculations for rotation (and similar operations) in three-dimensional space were done using the library Eigen[2]. To move in space, the three operations yaw, pitch and roll are needed to orientate the "turtle". A three-dimensional line, with a fixed length, is rendered by connecting two hexagons, as can be seen in Figure 1. Additionally a polygon mode, where the turtle can sequentially follow the shape of any polygon and add it to the mesh, is provided. The interpretation of symbols is taken from a book by P.Prusinkiewicz[4], which also explained the Turtle Graphics model in more detail.

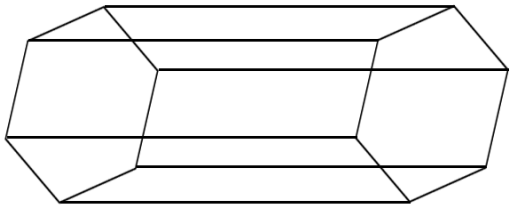

Figure 1: A three-dimensional line produced by "_3DTurtle" class

3 Terrain Generation based on Stream Erosion

Implemented by: Philipp Mildenberger and Stefan Spiss

3.1 Technical Overview

The project is based on the paper: "Terrain Simulation Using a Model of Stream Erosion"[3] . Initially an outline in 2D is representing the drainage basin - the area where the Stream Network is placed, is given. A single link is the basis, and represents the main stream. Figure 2 shows this scenario. The areas connected to a link are called "drainage areas". This area contributes water to a stream which is herein called link.

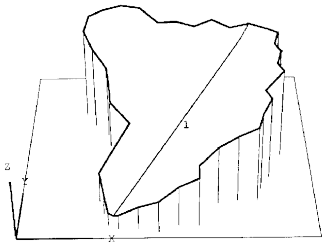

Figure 2: Initial Drainage Basin [3]

3.1.1 The Addition of a stream/link

A recursive algorithm is used to generate streams into this Network, until the drainage areas aren't strong enough to support a new stream. This recursion basis is predefined by a constant C which stands for "channel maintenance". A link is now added until $\frac{A}{L} < C$ where A is the local drainage area of a link and L is the length of an individual link. Since every link has

[1] www.openmesh.org

[2] www.eigen.tuxfamily.org

its own drainage area, drainage divides will be inserted if a link is inserted. Those drainage divides describe a ridge between the links. Every link has an ordinal number, which is in Shreve Order [5]. This means that a link has a magnitude $m + n$, where m and n are the magnitudes of the sublinks. This is defined recursively, until an exterior link is reached, which has magnitude 1. The following Figure 3 explains this concept.

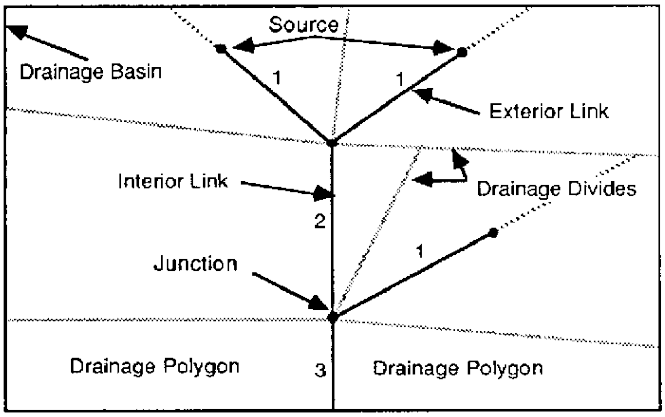

Figure 3: Drainage System [3]

3.1.2 Characteristics of a link

The junction position of a link is determined with the following equotion:

$$Junction = MeanJunction + rand() * DeltaJunction \tag{1}$$

where $MeanJunction$ is a predefined value, $rand()$ a random function with values from -1 to 1 and $DeltaJunction$ the strength of perturbation of the position.
The length of an exterior link is determined the same way with exchanging $MeanJunction$ with $MeanLength$ and $DeltaJunction$ with $DeltaLength$.
$Junction$ and $Length$ both must be between 0 and 1.
The Junction angle is estimated using the Howard geometric model [2]:

$$JunctionAngle = E_1 + E_2 \tag{2}$$

where

$$E_1 = arccos(\frac{S_3}{S_1}) \tag{3}$$

$$E_2 = arccos(\frac{S_3}{S_2}) \tag{4}$$

S_1, S_2 and S_3 are the slope tangients of a link, which are achieved with a simplified form of Flint's equation [1]:

$$S = p(2u - 1)^q \tag{5}$$

where S is the slope tangent, u the Shreve Order magnitude and q is a factor between -0.37 and -0.837.

3.1.3 Drainage Divides

The drainage divides are inserted similar as the links. They are also starting from the junction and splitting the drainage polygon in a certain angle.
These angles are computed according to the Howard geometric model [2]:
There are three different cases for the calculation:

- Angle for the drainage divide between the upper links.
- Angle for the drainage divide between stronger up link and down link.
- Angle for the drainage divide between weaker up link and down link.

For more details look at [2] (page 6-8).

3.1.4 Calculation of the height

After the horizontal drainage basin is generated. The height information must be added. The calculation of the stream heights is done using a recursive algorithm, starting from the predefined outlet (root) at the base elevation. Every recursive step uses following formula:

$$h_{upstream} = h_{downstream} + S * L \tag{6}$$

Where $h_{upsteram}$ is the height of the current junction, $h_{downstream}$ is the height of the downstream junction, S is the slope tangents and L is the length of the current link.
The height of the valley sidewalls are calculated with a similar model using the slope tangents of the valleys.

3.2 Implementation Detail

The Implementation is done with a (binary-)tree like structure, where the root node is the initial link. The leafs of the tree are the exterior links. The main loop of this implementation is done in "buildDrainageNetwork". The first part checks whether a link has to be inserted. If it will be inserted, another function recalculates the mesh based on the meta data saved in each node, like the parametric length or the shreve order. This loop continues until every sublink is saturated with its drainage area.
Unfortunately we could not complete the implementation due to the complexity of the topic and the difficulty of debugging OpenMesh. At the moment the subdivision of the links is almost working. Everything works fine until, after several iterations an unexpected condition occurs while deleting an edge.

References

[1] JJ Flint. Stream gradient as a function of order, magnitude, and discharge. *Water Resources Research*, 10(5):969–973, 1974.

[2] Alan D Howard. Optimal angles of stream junction: Geometric, stability to capture, and minimum power criteria. *Water Resources Research*, 7(4):863–873, 1971.

[3] Alex D Kelley, Michael C Malin, and Gregory M Nielson. *Terrain simulation using a model of stream erosion*, volume 22. ACM, 1988.

[4] Aristid Lindenmayer Przemyslaw Prusinkiewicz, Aristid Lindenmayer, James S Hanan, F David Fracchia, and Deborah Fowler. The algorithmic beauty of plants with. 1990.

[5] Ronald L Shreve. Statistical law of stream numbers. *The Journal of Geology*, pages 17–37, 1966.