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 20, 2015

1 Introduction

This is a really elegant and thoughtful introduction to our project. //introduce the name Lindenmayer-Systems, with abbreviation L-System.

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 a 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, which also explained the Turtle Graphics model in more detail.

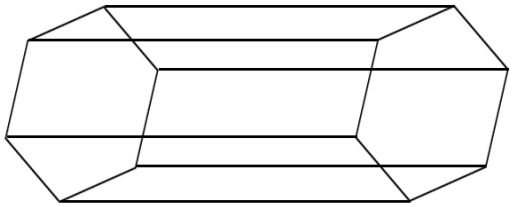

Figure 1: A three-dimensional line produced by "_3DTurtle" class

3 Procedural terain

Implemented by: Stefan Spiss and Phillip Mildenberger

3.1 Technical Overview

3.2 Implementation Detail

4 Results

[1] www.openmesh.org
[2] www.eigen.tuxfamily.org

References