University of Florida
Department of Computer and Information Science and Engineering

CAP 5705 Computer Graphics
Fall 2008

Instructor
Dr. Alireza Entezari

Final Project Report
by Ai-Ti Cheng

Topic:
Creation and rendering of realistic trees
Jason Weber and Joseph Penn

Group member:

Ai-Ti Cheng
66112196

Xingchao Liu
38596016
1 Introduction

The tree rendering is a popular research area in computer graphics in recent decades. *Creation and Rendering of Realistic Trees* by Jason Weber and Joseph Penn proposes a model for rendering realistic trees with efficiency and simplicity. Our goal of this project is to research and to implement this model. After our implementation, we accomplished a comprehensive analysis of the performance.

2 Background

Rendering a realistic tree in computer graphics is not an easy task, so there are many research papers proposed for tree rendering. The challenge of tree rendering is to support rendering various types of realistic trees and to achieve a real-time fashion. In computer graphics, rendering more photo-realism objects means increasing the computation time. There is an obvious trade-off between the levels of realism and the computation time. For meeting the realism and the real-time fashion, there are many tree rendering models proposed.

In 90s, the most popular model is L-system proposed by Lindenmayer [Lindenmayer, 1968]. The L-system is a string-rewriting and rule-based system[1]. The string rewriting is the main idea of the L-system. Under the predefined rules, it defines a complex object by substituting the partial components of an initially simple object. A tree is constructed by applying the substituting rules after several iterations. The advantage of the rule-based system is the easy management of the rules. So, the L-system has been widely applied in current applications.

However, the L-system is somehow limited in producing complicated three-dimensional branches with real-time fashion. So, the parametric-based tree model has been proposed. The goal of the parametric-based model is to apply the predefined a set of specifications for rendering different trees. *Creation and Rendering of Realistic Trees* by Jason Weber and Joseph Penn is a parametric tree model. It provides a set of parameters for four types of trees and specifies the structure of the tree model. The advantage of this model is the efficiency of applying the parameters without adapting the evolution algorithms which are considered time-consuming. So, this method seems more superior in computation efficiency. As a result, we choose this paper as the foundation of our implementation.

3 Methods

This tree model views a tree as a primary trunk with various curved clones. The model defines a set of parameters to specify the number, the probability, and the angle of a branch splitting to multiple branches along its length. It defines this situation as the dichotomous branching. The clones have similar attributes with the remaining length.

The model also defines the child branches which grows on the primary trunk and the clones. These child branches have completely different properties from their parent branch. Although the child and parent branches have different properties, the properties of the child branches are computed mainly based on the attributes of their parent. The child branches are also called the monopodial branches.

From above, the tree branches defined in this tree model are basically composed by the dichotomous branches and the monopodial branches. We discuss the tree creation in detail in the
following subsections.

### 3.1 Tree Creation

The complete tree model proposed by Jason Weber and Joseph Penn is based on two elements: the stems and the leafs. This paper defines the stems as the trunk or branches at any recursion level and each stem has its own coordinate system. For simulating the realistic trees, the model adapts numbers of variation parameters and widely applies random numbers for probabilities. Our implementation is focus on the stem rendering and the leaf rendering is remained as our future work.

### 3.2 The Clone Branch

The first part of stem rendering is the stem splits. This model refers $n$ as the recursive levels and appends a V at the end of a parameter for variations. A stem is composed by $nCurveRes$ segments at recursive level $n$. If $nCurveBack$ is zero, the z-axis of each segment rotates away from the z-axis of the previous segment by $(nCurve/nCurveRes)$ degrees about it's x-axis. If $nCurveBack$ is not zero, the first half of the stem is rotated $(nCurve/(nCurveRes/2))$ degrees and the second half is rotated $(nCurveBack/(nCurveRes/2))$ degrees. A random rotation $(nCurveV/nCurveRes)$ is also added to each segment for more realistic and natural branches[1].

The $0BaseSplit$ defines the number of the dichotomous branches at the end of the first segment of the primary trunk. The frequency of the splitting branches is specified as $nSegSplits$ and the angle between the clones is defined by $nSplitAngle$.

![Figure 1. Stem Split [1].](image)

The first figure has $0CurveRes$ equals to 3 (3 segments of a stem) and $BaseSplits$ equals to 2 (3 split branches at the end of the first segment). The second figure has $0CurveRes$ equals to 3 (3 segments) and $0SegSplits$ equals to 1 (2 split branches in each segment).
3.3 The Child Branch

The split branches, discussed in the previous subsection, simulate the split trunks of the realistic tree. However, the even distribution of split branches cannot really simulate the various branches of the natural trees. So, this model proposes the other component called the stem children. The levels of recursion refer to the levels of the stem children.

The $nBranches$ defines the maximum number of stem children in recursion level $n$. The average number of stem children is computed as

$$\text{stems} = \text{stems}_\text{max} \times (0.2 + 0.8 \times (\text{length}_\text{child}/\text{length}_\text{parent}) / \text{length}_\text{child,max})$$

for the first level children, and

$$\text{stems} = \text{stems}_\text{max} \times (1.0 - 0.5 \times \text{offset}_\text{child}/\text{length}_\text{parent})$$

for further levels of children.

For computing the length of each stem, this model first defines the length of the primary trunk as $\text{Length}_{\text{primary trunk}} = (\text{Scale} + \text{ScaleV}) \times (0\text{Length} + \text{0LengthV})$. The length of a stem children is computed as

$$\text{length}_\text{child} = \text{length}_{\text{trunk}} \times \text{length}_\text{child,max} \times \text{ShapeRatio}(\text{Shape}, (\text{length}_{\text{trunk}} + \text{offset}_\text{child}) / (\text{length}_{\text{trunk}} - \text{length}_\text{base}))$$

for the first level children, and

$$\text{length}_\text{child} = \text{length}_\text{child,max} \times (\text{length}_\text{parent} - 0.6 \times \text{offset}_\text{child})$$

for further level children. The $\text{ShapeRatio}(\text{shape, ratio})$ is a predefined function with two inputs of shape and ratio as follows.

<table>
<thead>
<tr>
<th>Shape</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 (conical)</td>
<td>0.2 + 0.8 * ratio</td>
</tr>
<tr>
<td>1 (spherical)</td>
<td>0.2 + 0.8 * sin( $\pi$ * ratio )</td>
</tr>
<tr>
<td>2 (hemispherical)</td>
<td>0.2 + 0.8 * sin( 0.5 * $\pi$ * ratio )</td>
</tr>
<tr>
<td>3 (cylindrical)</td>
<td>1.0</td>
</tr>
<tr>
<td>4 (tapered cylindrical)</td>
<td>0.5 + 0.5 * ratio</td>
</tr>
<tr>
<td>5 (flame)</td>
<td>ratio/0.7</td>
</tr>
<tr>
<td></td>
<td>(1.0 - ratio)/0.3</td>
</tr>
<tr>
<td>6 (inverse conical)</td>
<td>1.0 - 0.8 * ratio</td>
</tr>
<tr>
<td>7 (tend flame)</td>
<td>0.5 + 0.5 * ratio/0.7</td>
</tr>
<tr>
<td></td>
<td>0.5 + 0.5 * (1.0 - ratio)/0.3</td>
</tr>
<tr>
<td>8 (envelope)</td>
<td>use pruning envelope (see Section 4.6)</td>
</tr>
</tbody>
</table>

**Figure 2. ShapeRatio function [1]**

The $nDownAngle$ defines the angle of the stem child rotating away from the parent and $nDownAngleV$ specifies the variation of the rotation. The $nRotate$ defines the angle between stem children with the same recursive level. These variations of parameters provide the more flexible specifications of trees for simulating the unordered and natural branches.
For computing the radius of the tree branches, this model defines the base radius of the main trunk as $\text{Radius}_{\text{trunk}} = \text{length}_{\text{trunk}} \times \text{Ratio} \times \text{0Scale}$ and the radius of the child branches as $\text{Radius}_{\text{child}} = \text{radius}_{\text{parent}} \times (\text{length}_{\text{child}} / \text{length}_{\text{parent}})^{\text{RatioPower}}$.

**Figure 3. Tree Diagram.** [1]

This figure specifies a clear structure of a tree with one primary trunk and three first level stem children.
4 Design and Implementation

We follow the specifications given by this paper and complete the branch implementation. This section will describe our program design and implementation of this model.

4.1 Data Structure

For implementing the tree model, we define our data structures as follows.

```c
struct Tree
{
    //parameter information
    int Levels;
    float* Length;
    float* LengthV;
    float* Branches;
    float* DownAngle;
    float* DownAngleV;
    float* Taper;
    float* SplitAngle;
    float* SplitAngleV;
    float BaseSize;
    float* tLength;
    float* tBottomRadius;
    float* tTopRadius;
    int TotalStemNum;
    int* StemNum;
    int* CurveRes;
    float* CurveBack;
    float* Curve;
    float* CurveV;
    float* SegSplits;
    float BaseSplit;
    float* Rotate;
    float* RotateV;
    Branch* stems;
};
```

The Tree defines all the parameters specified in this paper and values calculated from the predefined parameter list.
4.2 Program Diagram

For rendering a tree, the important components are the branch splits and the stem children. So, we build two recursive functions for these two components. One is the renderTree(level) and the other one is renderStemSegs(seg). For giving each stem its own local coordinate system, we apply glPushMatrix() and glPopMatrix(). We use cylinder to simulate the geometry of the stems, so the function, gluCylinder, is called when rendering the segments of the stems. The program diagram shows our program procedure.

Figure 4. Program Diagram.
4.3 User Interface

For implementing the user interface, we apply GLUI library. We provides four radio buttons for choosing one tree which we want to render. Each specification of each tree is shown in the text boxes on the GLUI menu.

Figure 4. GLUI Menu.
The menu defines the four radio buttons for different tree choice and the parameters are shown in the text boxes. The start button is used when the user wants to render a tree. The quit button terminates the program.

4.3 Background

The background of our program is Century Tower which is a famous landscape of the University of Florida (Figure 4).
5 Simulation Results

The program runs in our personal laptop and the specifications are as follows.

CPU: T2400 @ 1.83 GHz
Memory: 1 GB
Operation System: Ubuntu 8.04.1
Kernel: 2.6.24-19-generic

This section shows our implementation results.

5.1 Tree One: Quaking Aspen

![Figure 5.1. Quaking Aspen.](image)
5.2 Tree Two: Black Tupelo
5.3 Tree Three: Weeping Willow

Figure 5.3 Tree Three: Weeping Willow
5.4 Tree Four: CA Black Oak

![Tree Four: CA Black Oak](image)

Figure 5.4 Tree Three: CA Black Oak

6 Performance

The stem composed by several segments gives us the realistic tree appearances. The two components, the branch splits and the stem children, achieve the construction of the complicated branch shapes. Our implementation does not achieve the real-time fashion. We analyze the reasons as follows. First, we apply an extremely large number of glPushMatrix() and glPopMatrix() in this program and these function may delay the computation time. Second, we run our program in our personal laptop and do not apply special Graphics Processing Unit (GPU).
7 Future Work

In our implementation, we complete the branch construction and the leaf rendering can be remained as our future work.

8 Contribution List

8.1 Selected Topic:
Ai-Ti Cheng and Xingchao Liu

8.2 Proposal:
Ai-Ti Cheng

8.3 Presentation:
  a. Slides: Ai-Ti Cheng and Xingchao Liu
  b. Main-Presenter: Xingchao Liu
  c. Co-Presenter: Ai-Ti Cheng

8.4 Implementation
  a. User Interface (GLUI): Ai-Ti Cheng
  b. Parametric Tree Model: Ai-Ti Cheng
  c. Background UF Theme: Ai-Ti Cheng

8.5 Final Project Report:
Ai-Ti Cheng

9 Acknowledgment

[1] The png loader used in the tree.c and png.h are from David HENRY, 2007.
[2] Use the GLUI example codes for our user interface implementation.
[4] The bark1.png is from Xingchao's previous project

10 References