Computer Graphics Seminar WiSe 2012/2013
A Hybrid Multiresolution Representation for Fast Tree Modeling and Rendering [1]

Yongzhi Ong

31 March 2013

Abstract

This report is written for the Computer Graphics Seminar held at the Georg-August University of Goettingen by Dr. Katarina Smolenova and Professor Winfried Kurth during the winter semester of 2012/2013. The paper titled *A Hybrid Multiresolution Representation for Fast Tree Modeling and Rendering* by Lluch et al. [1] is studied and presented in the report for evaluation.

Contents

Chapter 1

Introduction and Background

1.1 General Overview of Paper

A set of techniques for the modeling and representation of trees and in general, plants, are presented in the paper by Lluch et al..

The authors categorize existing techniques for these purposes into two categories - multiresolution techniques and image-based techniques. Multiresolution techniques refer to techniques that employ geometric simplifications on models while image-based techniques use billboards and textures to represent models.

Multiresolution techniques generally fail to capture and retain the structures of natural plants or trees. On the other hand, image-based techniques often require large storage spaces and produce visible artifacts upon close distances to the viewing camera or eye.

A hybrid technique is created by the authors of the paper in order to address the problems given by the above two categories.

1.2 Background

Conventionally, various types and combinations of texture-mapped polygons and billboards are used reduce the number of polygons rendered [9]. A single tree may be represented by a single image, i.e. a single quad polygon, or by a cloud of billboards. Other techniques based on L-Systems [7], components [2] or re-construction from images of real plants [8] have also been used.

In addition to the above mentioned simplified representations, multiple level-of-details (LOD) have also been explored. Some of the mentioned LOD techniques are:

- Degradation at Range and Pixel-based LODs [10]
- Space Partitioning and Multiresolution [3]
- Cluster-based Hierarchical Polygon Decimation and Compression [11]
- Layered Depth Images [4]
- Volumetric Textures [5]
- Bidirectional Textures [6]

Lluch et al. define the term *Multiresolution* as the representation of objects at various LODs. Four main characteristics of good multiresolution models are given:

- Size of model does not increase with number of LODs
- Extraction of LODs is fast enough for interactive rendering
- No loss of information
- Smooth transition between LODs

To justify the usage of L-Systems in their hybrid technique, Lluch et al. highlight that L-System models have the following properties:

- Quick model generation
- Low storage requirement
- View direction and position independent from rendering quality

Their proposed hybrid technique breaks the problem into two:

- Trunk and branches
- Leaves or foliage

Chapter 2

Modeling using L-Systems

2.1 Parametric L-Systems

In this section, an example of parametric L-Systems is given to illustrate the idea of string replacement grammar with parameters. This concept is used by Lluch et al. in their hybrid solution to model the trunk and branches of plants or trees.

An *Axiom* string is defined along with two replacement rules. The conditions for rule application during each iteration are given immediately after the colon symbol. The strings that match the string (along with the parameters) on the left of -> are replaced by the string on the right of ->.

```
Axiom :
A(length)

Rule 1 :
A(l):  itNum < maxIt -> B(l) [A(l/2) A(l/2)]

Rule 2 :
A(l):  itNum = maxIt -> B(l)
```

Before the first application of rules, the *Axiom* string appears as the output string chain itself. *itNum* refers to the number of rule application iterations. *maxIt* refers to the maximum number of rule applications.

```
Axiom :  A(1)
itNum = 0
maxIt = 2


Output Chain:
A(1)
```

On the first rule application, the output chain is replaced by rule 1.

```
Rule 1 :
A(l):  itNum < maxIt -> B(l) [A(l/2) A(l/2)]
itNum = 1
maxIt = 3


Output Chain (Before)
A(1)
Output Chain:
B(1) [A(0.5) A(0.5)]
```

On the second rule application, part of the output chain is replaced by rule 1.

```
Rule 1 :
A(l):  itNum < maxIt -> B(l) [A(l/2) A(l/2)]
itNum = 2
maxIt = 3


Output Chain (Before)
B(1) [A(0.5) A(0.5)]
Output Chain:
B(1) [B(0.5) [A(0.25) A(0.25)]
B(0.5) [A(0.25) A(0.25)]]
```

On the final rule application, part of the output chain is replaced by rule 2.

```
Rule 2 :
A(l):  itNum = maxIt -> B(l)
itNum = 3
maxIt = 3


Output Chain (Before)
B(1) [B(0.5) [A(0.25) A(0.25)]
B(0.5) [A(0.25) A(0.25)]]
Output Chain:
B(1) [B(0.5) [B(0.25) B(0.25)]
B(0.5) [B(0.25) B(0.25)]]
```

2.2 Turtle Interpretation

In this section, an example of turtle interpretation is given to illustrate the idea of interpreting L-system strings geometrically.

Interpreting the character *F* as a *Forward* command for the turtle, and *R* as a *Rotate* command for the turtle, an L-System string can be geometrically interpreted by moving the turtle using the sequence of commands as specified by the string.

For example, on the first *F* command, the turtle is moved forward by 1 unit:

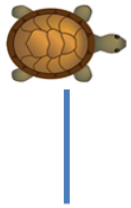
```
F(1) [R(90) F(1)] F(2)
```

On the *[* symbol, the state of the turtle is stored:

```
F(1) [ R(90) F(1)] F(2)
```

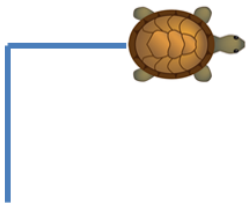
On the *R* command, the turtle is rotate by 90 degrees:

```
F(1) [ R(90) F(1)] F(2)
```



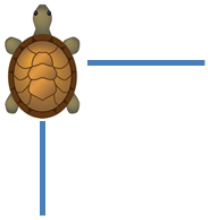
On the next *F* command, the turtle is moved forward by 1 unit again:

```
F(1) [ R(90) F(1)] F(2)
```



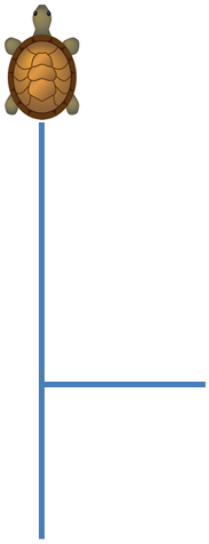
Encountering the *]* character, the state of turtle is restored to the last stored state:

```
F(1) [ R(90) F(1)] F(2)
```



Finally, the last *F* command moves the turtle forward by 2 units:

```
F(1) [ R(90) F(1)] F(2)
```



Chapter 3

Solution and Algorithm

In the previous chapter, basic parametric L-Systems and turtle interpretation of L-System strings are illustrated. In this chapter, the hybrid solution introduced by Lluch et al. is described. An overview of the solution consists of 5 steps:

- Parametric L-System specification
- Construction of *Tree Abstract Data Type (tADT)* from L-System string
- Construction of *Multiresolution Chain* from *tADT*
- Construction of *Bounding Box Hierarchy* from L-System string
- Generating textures representing the foliage from Bounding Box Hierarchy

In section 3.1, the first three steps are described. In section 3.2, the last two steps are described. Finally, in section 3.3, the retrieval of the model for rendering at various LODs is described.

3.1 Trunk and Branch Modeling

In the first step of the solution by Lluch et al., the branches and trunk of trees are specified using parametric L-Systems and turtle interpretation symbols. From the output chain of the parametric L-System string, a *Tree Abstract Data Type (tADT)* is constructed.

Suppose the output chain from the parametric L-System is:

```
C1 [ C2 [ C3 ] [ C4 ] ] [ C5 ]
```

For illustration purposes, we assume that C1, C2, C3, C4, and C5 are turtle interpretation symbols, all of which have a property *length* of value 1. The algorithm begins with a node (the tADT is a tree data structure composed of connected nodes). We begin

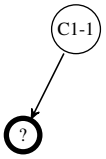
to traverse the output chain (the L-System string) from left to right. For each symbol encountered, the tADT is modified. Using the above output chain, the construction of the tADT then proceeds as follow:

`C1` is encountered. It is added to the list of symbols inside the first node. Length of `C1` is added to the total length of the first node:

```
C1 [ C2 [ C3 ] [ C4 ] ] [ C5 ]
```

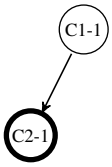
An open bracket is encountered. A new node is added as a child of the current node. The new node is set as the current node (bold circle):

```
C1 [ C2 [ C3 ] [ C4 ] ] [ C5 ]
```



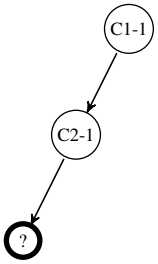
`C2` is encountered. It is added to the list of symbols inside the current node. Length of `C2` is added to the total length of the current node:

```
C1 [ C2 [ C3 ] [ C4 ] ] [ C5 ]
```



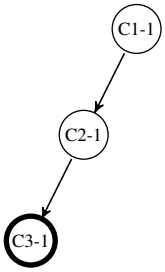
An open bracket is encountered. A new node is added as a child of the current node. The new node is set as the current node (bold circle):

```
C1 [ C2 [ C3 ] [ C4 ] ] [ C5 ]
```



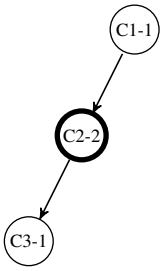
`C3` is encountered. It is added to the list of symbols inside the current node. Length of `C3` is added to the total length of the current node:

```
C1 [ C2 [ C3 ] [ C4 ] ] [ C5 ]
```



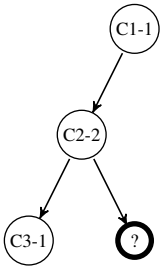
A close bracket is encountered. The total length of the current node (that contains `C3`) is added to the total length of its parent node (that contains `C2`). The parent of the current node, i.e. node that contains `C2`, is set as the current node.

```
C1 [ C2 [ C3 ] [ C4 ] ] [ C5 ]
```



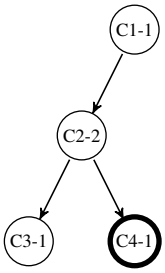
An open bracket is encountered. A new node is added as a child of the current node. The new node is set as the current node (bold circle):

```
C1 [ C2 [ C3 ] [ C4 ] ] [ C5 ]
```



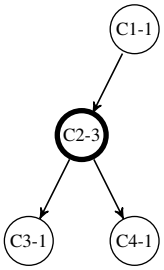
`C4` is encountered. It is added to the list of symbols inside the current node. Length of `C4` is added to the total length of the current node:

```
C1 [ C2 [ C3 ] [ C4 ] ] [ C5 ]
```



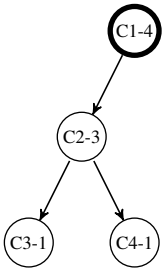
A close bracket is encountered. The total length of the current node (that contains `C4`) is added to the total length of its parent node (that contains `C2`). The parent of the current node, i.e. node that contains `C2`, is set as the current node.

```
C1 [ C2 [ C3 ] [ C4 ] ] [ C5 ]
```



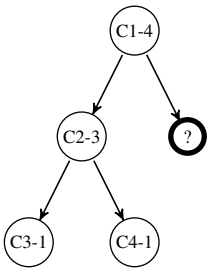
A close bracket is encountered. The total length of the current node (that contains `C2`) is added to the total length of its parent node (that contains `C1`). The parent of the current node, i.e. node that contains `C1`, is set as the current node.

```
C1 [ C2 [ C3 ] [ C4 ] ] [ C5 ]
```



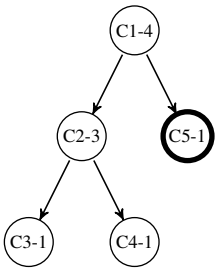
An open bracket is encountered. A new node is added as a child of the current node. The new node is set as the current node (bold circle):

```
C1 [ C2 [ C3 ] [ C4 ] ] [ C5 ]
```



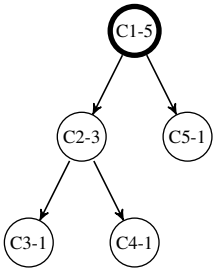
`C5` is encountered. It is added to the list of symbols inside the current node. Length of `C5` is added to the total length of the current node:

```
C1 [ C2 [ C3 ] [ C4 ] ] [ C5 ]
```



A close bracket is encountered. The total length of the current node (that contains `C5`) is added to the total length of its parent node (that contains `C1`). The parent of the current node, i.e. node that contains `C1`, is set as the current node.

```
C1 [ C2 [ C3 ] [ C4 ] ] [ C5 ]
```



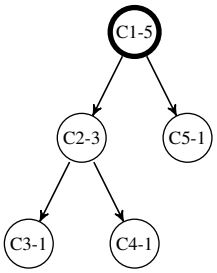
The construction of the *tADT* ends when the end of the L-System string is reached. This marks the end of step 2 of the hybrid technique.

In step 3 (Multiresolution chain construction) of the solution, Lluch et al. attempted to use different metrics or properties of the constructed *tADT* to construct a *Multiresolution Chain*. Attempted metrics include:

- Number of children
- Number of descendents
- Longest path to a leaf node
- **Branching length** (accumulated in each node in during tADT construction)

It is suggested by the authors that *Branching Length* provides the best visual satisfaction in the final visualized results. Continuing from the previous example for *tADT* construction, the construction of the *Multiresolution Chain* proceeds as follow:

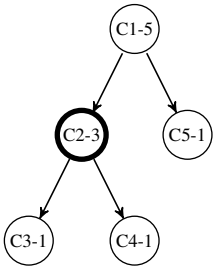
The *tADT* is traversed beginning from the root node, i.e. the node containing `C1`. A node is reached and therefore a new entry `C1` is added to the *Multiresolution Chain*. As the node containing `C1` has more than one child node, a `SAVE(C1)` entry is added to the list:




```
C1 SAVE(C1)
```

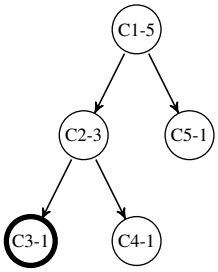
The *tADT* traversal proceeds to the next node that has the highest *Branching Length*. A node is reached and therefore a new entry `C2` is added to the *Multiresolution Chain*. As the node containing `C2` has more than one child node, a `SAVE(C2)` entry is added

to the list:



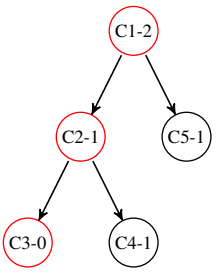
`C1 SAVE(C1) C2 SAVE(C2)`

The *tADT* traversal proceeds to the next node that has the highest *Branching Length*. Since both the children of the node containing `C2` have *Branching Length* 1, the traversal proceeds to the first child of the node, i.e. the node containing `C3`. A node is reached and therefore a new entry `C3` is added to the *Multiresolution Chain*. As the node containing `C3` has no children, no `SAVE` entry is added to the list::



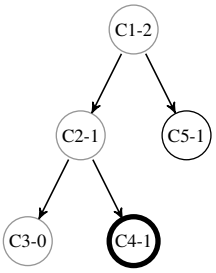
`C1 SAVE(C1) C2 SAVE(C2) C3`

As the node containing `C3` is a leaf node in the *tADT*, the *Branching Length* of all nodes along the path from the root to the leaf node is updated. Each node in the path subtracts its own *Branching Length* and the accumulated *Branching Length* values of subsequent nodes in the path from its *Branching Length* value:



`C1 SAVE(C1) C2 SAVE(C2) C3`

After updating the *Branching Length* values of the path, control of the *tADT* traversal is restored to the last saved point (with remaining children nodes), i.e. node containing `C2`. This is indicated in the *Multiresolution Chain* by a `RESTORE(C2)` entry. Traversal then continues to the next child node with the highest *Branching Length* value again:




```
C1 SAVE(C1) C2 SAVE(C2) C3 RESTORE(C2) C4
```

The *Multiresolution Chain* construction continues this way until all the *Branching Length* values in the *tADT* have been subtracted till zero. The final *Multiresolution Chain* is:

```
C1 SAVE(C1) C2 SAVE(C2) C3 RESTORE(C2) C4 RESTORE(C1) C5
```

3.2 Leaves and Foliage

Step 4 of the hybrid technique by Lluch et al. reads the parametric L-System string from left to right and performs turtle interpretation. A bounding box hierarchy (in the form of a tree data structure) corresponding to the L-System branching is constructed. The root of the bounding box hierarchy contains the axis aligned bounding of all the foliage geometry. Subsequent levels of the bounding box hierarchy contain boundings of subsets of the foliage geometry, with each subsequent level containing the bounding of a smaller subset than the previous.

In the final step of the hybrid technique, textures are generated from the bounding box hierarchy. Considering the bounding box hierarchy as a graph, a set of textures corresponding to the six diagonal planes of each node in the graph is projected, generated and saved. All textures are 128 by 128 pixels in dimension. To prevent generating too many textures (in consideration of available storage space), the ratio of the bounding volume at each node to the bounding volume at the root node is compared against a certain threshold. Textures for a node is not generated if the threshold is exceeded.

3.3 Combined Execution

The precomputed *Multiresolution Chain*, bounding box hierarchy and textures are used together as a hybrid of the two main categories mentioned in section 1.1. In particular, the *Multiresolution Chain* can be categorized under multiresolution techniques while the bounding box hierarchy and textures can be categorized under image-based techniques.

The desired LOD for rendering is extracted from the *Multiresolution Chain* by traversing the chain from the beginning. Each `RESTORE` entry in the chain represents the next LOD with further refined details. For the foliage, each further level in the bounding box hierarchy represents the next LOD with further refined details.

The extraction technique, e.g. based on pixel error, appears to be out of the scope of this paper and is not discussed.

Chapter 4

Results and Conclusion

Lluch et al. presented the following results of the hybrid technique:

- 100 Trees
  - Geometric Model: 3 fps (min)
  - Hybrid Multiresolution Model: 69 fps (min)
- 2000 Trees
  - Geometric Model: 0.1 fps (min)
  - Hybrid Multiresolution Model: 6 fps (min)

Potential Improvements to the technique include:

- Smooth transitions between LODs
- Reduce memory required for textures
- Wind movements

Bibliography

[1] L. Javier, C. Emilio, H. Jose Luis, and V. Roberto. A Hybrid Mutiresolution Representation for Fast Tree Modeling and Rendering. *Procedia Computer Science*, 1(1):485–494, 2010.

[2] B. Lintermann and O. Deussen. Interactive Modeling of Plants. *IEEE Computer Graphics and Applications*, 19(1):56–65, 1999.

[3] D. Marshall, D. S. Fussell, and A. T. Campbell. Multiresolution Rendering of Complex Botanical Scenes. In Wayne A. Davis, Marilyn M. Mantei, and R. Victor Klassen, editors, *Graphics Interface*, pages 97–104. Canadian Human-Computer Communications Society, 1997.

[4] N. Max. Hierarchical rendering of trees from precomputed multi-layer z-buffers. In *Proceedings of the Eurographics Workshop on Rendering Techniques '96*, pages 165–174, London, UK, UK, 1996. Springer-Verlag.

[5] A. Meyer and F. Neyret. Interactive Volumetric Textures. In George Drettakis and Nelson Max, editors, *Eurographics Workshop on Rendering Techniques (Rendering Techniques'98)*, pages 157–168, Vienna, Autriche, 1998. Eurographics, Springer Wein.

[6] A. Meyer, F. Neyret, and P. Poulin. Interactive Rendering of Trees with Shading and Shadows. In Steven J. Gortler and Karol Myszkowski, editors, *12th Eurographics Workshop on Rendering Techniques*, pages 183–196, London, Royaume-Uni, 2001. Springer.

[7] P. Prusinkiewicz, A. Lindenmayer, and J. Hanan. Development Models of Herbaceous Plants for Computer Imagery Purposes. *SIGGRAPH Computer Graphics*, 22(4):141–150, 1988.

[8] L. Quan, P. Tan, G. Zeng, L. Yuan, J. Wang, and S. B. Kang. Image-based Plant Modeling. *ACM Trans. Graph.*, 25(3):599–604, July 2006.

[9] H. Romeijn, F. Sheth, and C.J. Pettit. An Evaluative Review of Simulated Dynamic Smart 3D Objects. *ISPRS Annals of Photogrammetry, Remote Sensing and Spatial Information Sciences*, 1(4):125–130, 2010.

[10] J. Weber and J. Penn. Creation and Rendering of Realistic Trees. In *Proceedings of the 22nd annual conference on Computer graphics and interactive techniques*, SIGGRAPH '95, pages 119–128, New York, NY, USA, 1995. ACM.

[11] X. Zhang, F. Blaise, and M. Jaeger. Multiresolution Plant Models with Complex Organs. In *Proceedings of the 2006 ACM International Conference on Virtual Reality Continuum and its Applications*, VRCIA '06, pages 331–334, New York, NY, USA, 2006. ACM.