Housing Layout Design Using Fractals

A Computer Tool and its Practical Use

KOBAYASHI Yoshihiro and BATTINA Subhadha
College of Architecture and Environmental Design, Arizona State University, USA

Keywords: fractal, housing layout, virtual reality, generative design system, Java3D

Abstract: This paper introduces a computer-based tool for three dimensional (3D) landscape simulations of housing-layout-design using the concepts of fractals with Iterative Function System (IFS). Housing layout design is defined as a design to allocate many house-units in the undeveloped site. The tool generates a variety of layout designs consisting of multiple dwelling house-units from manual inputs or a template pattern defined as an XML file. Each unit can contain any detailed 3D components found in the residential development such as a house, roads, walls, trees etc. The template defines the transformation rules for IFS including the information of geometrical relationships between the stages in the iteration and of the components used in stop-conditions. The application tool is formulated, implemented and tested. The results in the case studies using several practical sites are demonstrated and evaluated based on the experiments in the design studio.

1 INTRODUCTION

Currently 3D computer graphics (3DCG) modelling and rendering applications have been used for landscape/sight-seen simulations. It is very helpful to observe and understand the landscape models from the realistic images and animations created by the applications. It is also important to compare alternative designs, especially for designing housing layouts. However, it takes a lot of time and labor not only to create each detailed component but also to create alternative designs modifying the location of each component one by one. Therefore, it is useful to develop a tool to reduce the time and labor in creating components and to generate alternative landscape models easily and automatically.

There are several approaches to automatically generate virtual models for landscape simulations. Some approaches based on probability can generate new models using the attributes extracted from the existing models. The value of each attribute is calculated probabilistically. This approach is useful when several similar yet slightly different models are required such as 3D virtual city modelling (Kato et al. 1999). However, it is difficult to create many alternative designs at the same time. On the other hand, some rule-based approaches such as shape-grammar (Stiny and Mitchell 1980, Mitchell 1989), L-system, etc. can generate new models by editing each component using a set of rules. This approach is often used when the constrained rules and satisfactions can be predefined. However, it needs more expert knowledge to handle the rules in order to get more practical results.

In the initial phase of designing housing layouts, it is useful to compare to alternative layouts as many as possible rather than a few practical or similar layouts. Since most of the sites are undeveloped, it is more important for the designers and clients to visualize the outputs quickly and generate alternatives easily. For example, it is important to observe how a design changes the landscape when 1000 house-units are developed in a mountain area. In that case, the more alternative designs are tested, the better a solution can be achieved. From this point of view, it is considered to use the concepts of fractals, which is known as an iterative function system (IFS), for generating alternatives. One of the reasons to use fractals is that each rule can be parameterised very simply with the relationship between a parent shape and a child shape. Another reason is that it is possible to generate a lot of alternatives from a few rules. For example, three relation rules can generate 512 different designs, though the number depends on the symmetries of units. In addition, any detailed units can be reused because the location and direction of each unit are assigned only at the stop-condition in IFS. In other words, once a designer creates detailed house-units, it is possible to generate alternative detailed landscape models only by replacing the units. The objective of this study is to develop a tool to generate housing layout designs quickly and easily with simple interface and without expert knowledge.

2 RELATED FIELDS

The concept of fractals was pioneered by a mathematician, Benoit Mandelbrot, in the seventies. Though it is not a new concept, the idea of fractals is explored in a variety of ways from medicine to media art and from statistical analysis to mathematics. Because of its geometric nature, the idea gained its popularity with both mathematicians and designers (Peitgen, Jürgen, and Saupe 1992).

Broadly speaking, there are two ways that fractal concepts can be used in architecture and design. First, the fractal dimension can be measured and used as an analytical tool for design (Bovill 1996). Second, fractal organization principles can be used to generate complex rhythms for use in design.

The concept of fractals has been already used as a common tool in 3D computer graphics (CG) fields for creating 3D tree models, mountains, water surfaces, clouds, etc. Many commercial packages of modelling and rendering for CG such as 3D Studio Max, Maya, and Form-Z support the functions.

3 METHODOLOGY

The core algorithm used to generate the housing layout designs is based on the concept of the Iterated Function Systems (IFS). The Sierpinsky Gasket, introduced by the Polish Mathematician Waclaw Sierpinski, is one such example in which the parent unit is subdivided into smaller units (Peitgen, Jürgen, and Saupe 1992). Each unit has three variables, a reference point P and two direction vectors U&V. Since the parent's UV directions and children UV directions are both linearly independent, the children UV can be represented with parent UV. By passing the portion from parent UV to children UV iteratively, one unit will be subdivided or transformed to several units until it reaches the stop condition. The details are explained in the proposal paper to use fractals in designing housing layouts (Kobayashi and Battina 2004). In addition to the regular IFS methods, a new idea of selecting an option from scalable-unit or static-unit in the stop condition is proposed in this research. If the scalable unit is selected, the 3D objects will be scaled in the stop condition. Otherwise, they will be allocated without scaling. The reason why these options are added is that there are some components that can be scaled and some that should not be scaled in designing housing layouts. For example, streets, ground, etc. can be scaled, but houses and trees should not be scaled corresponding to the site's size. The following is the pseudo code to generate a housing layout design.

```
Generate (unit, parameters){
        if (the length of unit-U or unit-V direction < stop length of the unit){
              Initialize a new 3D object, 3Dobj, from the 3DS file;
              Get the reference point, 3Dr, to put the 3D object;
              Get the UV directions, 3Du and 3Dv;

              if (unit is scalable)
                  Scale the 3Dobj corresponding to the area created with 3Du and 3Dv;
              else if (unit is static)
                  Continue;

              Move the 3Dobj to the 3Dr point;
              Rotate the 3Dobj matching the 3Du to X direction;
        }
        else{
              Create a new set of Child-units using parameters;
              for each Child-unit {
                   Generate (Child-unit, parameters);
              }
        }
      }
```

4 SYSTEM FRAMEWORK

To generate 3D landscape models using this tool, there are 4 steps: 1) import terrain data, 2) define the parent box, 3) generate 2D fractal patterns and decide the design, and 4) generate a 3D model from a 2D fractal pattern (Figure 1).

The application tool is implemented in Java, one of the object-oriented programming languages, with Java3D API. It is easy to communicate with XML files. The user creates the transformation rules as a template file in an XML format, or specifies the rules manually by drawing in the tool. It consists of two main panels, one for viewing and opening the site and the other for defining the child units. The tool can import the terrain data of a DXF file, and export the 3D output model as DXF, OBJ, VRML or 3DS files.

The following is one example defining a child unit in the template file.

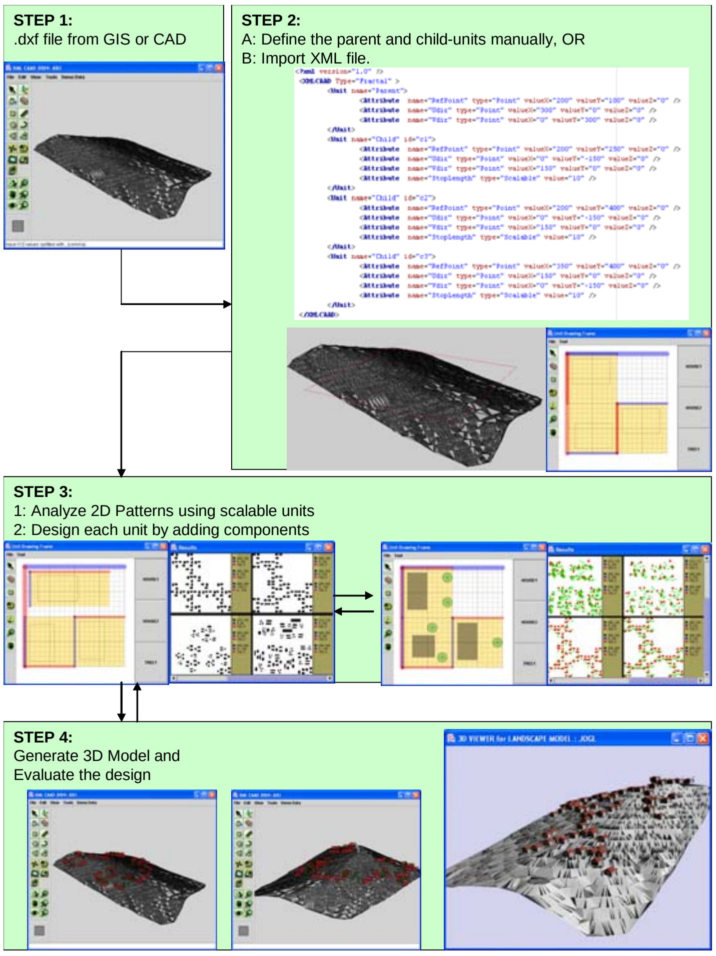
```
<Unit name=”Child” id=”c1”>
   <Attribute name=”RefPoint” type=”Point” valueX=”200” valueY=”400” valueZ=”0” />
   <Attribute name=”Udir” type=”Point” valueX=”0” valueY=”-150” valueZ=”0” />
   <Attribute name=”Vdir” type=”Point” valueX=”150” valueY=”0” valueZ=”0” />
   <Attribute name=”Stop” type=”Scalable” value=”10” />
</Unit>
```

The XML file describes the information about the reference point, and two direction vectors, *u* and *v*, for one parent unit and multiple child units. In this case, the child unit is defined as a scalable unit with the reference point (200, 400, 0), the *u* direction vector (0, -150, 0), and *v* direction vector (150, 0, 0). Its stop condition is when the length of *u* or *v* directions becomes less than 10.

On the other hand, in order to specify the rules manually, the user defines the boundary box as a parent unit after importing terrain data. It is shown in red at step 2 in Figure 1.

In the third step, the user defines the child units and its components in 2D panel. For each unit a reference point, u direction, and v direction must be defined. The reference point is represented as a gray circle, and u and v directions are represented as a light blue bar and a light red bar respectively. The unit is defined as a scalable unit in default. If the user imports 3D objects into the unit, the unit is defined as a static unit. When multiple units are defined, it is allowed to use static and scalable ones at the same time. Switching between these two choices enables the user to make more flexible designs even with the same rules.

The 3D landscape model is automatically generated from the selected fractal pattern in step 3. The main panel displays the results in the three dimensional view. It allows the user to zoom, rotate and make other small transformations to the resulting output.


STEP 1:
.dxf file from GIS or CAD
STEP 2:
A: Define the parent and child-units manually, OR
B: Import XML file.
STEP 3:
1: Analyze 2D Patterns using scalable units
2: Design each unit by adding components
STEP 4:
Generate 3D Model and
Evaluate the design
3D VIEWER for LANDSCAPE MODEL : JOGL

```
<?xml version="1.0" ?>
 <XMLCAAD Type="Fractal" >
       <Unit name="Parent">
              <Attribute  name="RefPoint" type="Point" valueX="200" valueY="100" valueZ="0" />
              <Attribute  name="Udir" type="Point" valueX="300" valueY="0" valueZ="0" />
              <Attribute  name="Vdir" type="Point" valueX="0" valueY="300" valueZ="0" />
       </Unit>
       <Unit name="Child" id="c1">
              <Attribute  name="RefPoint" type="Point" valueX="200" valueY="250" valueZ="0" />
              <Attribute  name="Udir" type="Point" valueX="0" valueY="-150" valueZ="0" />
              <Attribute  name="Vdir" type="Point" valueX="150" valueY="0" valueZ="0" />
              <Attribute  name="StopLength" type="Scalable" value="10" />
       </Unit>
       <Unit name="Child" id="c2">
              <Attribute  name="RefPoint" type="Point" valueX="200" valueY="400" valueZ="0" />
              <Attribute  name="Udir" type="Point" valueX="0" valueY="-150" valueZ="0" />
              <Attribute  name="Vdir" type="Point" valueX="150" valueY="0" valueZ="0" />
              <Attribute  name="StopLength" type="Scalable" value="10" />
       </Unit>
       <Unit name="Child" id="c3">
              <Attribute  name="RefPoint" type="Point" valueX="350" valueY="400" valueZ="0" />
              <Attribute  name="Udir" type="Point" valueX="150" valueY="0" valueZ="0" />
              <Attribute  name="Vdir" type="Point" valueX="0" valueY="-150" valueZ="0" />
              <Attribute  name="StopLength" type="Scalable" value="10" />
       </Unit>
 </XMLCAAD>
```

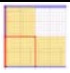
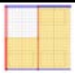
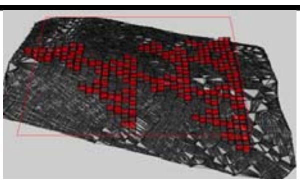
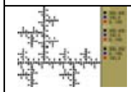
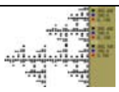
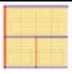
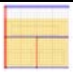
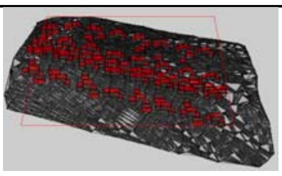
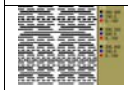
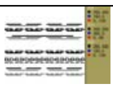
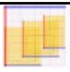
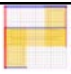
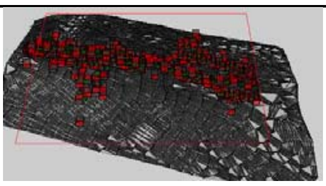
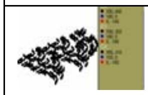
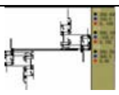
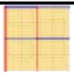
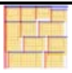
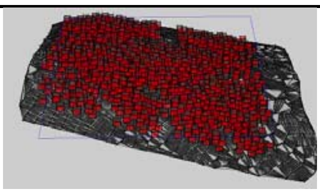
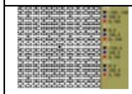
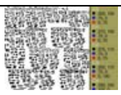
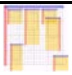
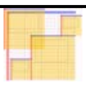
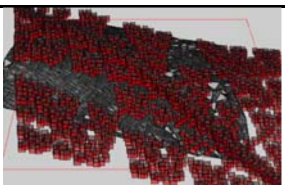
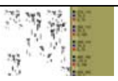
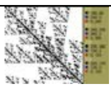
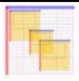
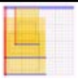
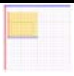
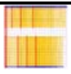
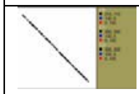
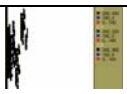
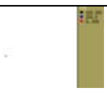
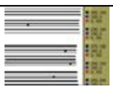
Figure 1 System Framework

5 CASE STUDY

This section first shows some of the generated results from the exercise for students to use the tool and to create housing layout designs (Table 1). The roughly categorized 6 classes, which were useful for the beginners in the exercise, are explained in section 5-1. Section 5-2 shows 3D views of one result and Section 5-3 describes some techniques of how to get the expected results.

Table 1 shows the screen shots of the results. The results are categorized into 6 classes: Basic, Scale, Overlap, Dense, Extend, and Poor. In the table, several results are posted for each category. The first two sub-columns of the Screen Shot Images field show the results using scalable units. The top cell has the image of definition of child units and the bottom cell has the 2D fractal pattern generated with the rules defined above. The last sub-column of the Screen Shot Image field shows the 3D view of the result using static units.

Table 1 The Results of Case Studies

Categories	Screen Shot Images			
Basic				
				
Scale				
				
Overlap				
				
Dense				
				
Extend				
				
Poor				
				

As explained in the previous section, the user needs to understand the fact that fractal is beyond the imagination. In short, it is fun for users to play around with the tool, but it is difficult to acquire enough skills to manage the parameters in order to get the results close to the expected one.

5.1 Categories

Here explains the 6 categories. Through the exercises using this tool in the design studio, it was found that it was very difficult to discuss the inputs and the outputs without any categories because it can generate a huge amount of alternatives. In similar problems to this, the roughly or subjectively defined categories often help a lot in the design process. For example, when 4 or 5 colors must be chosen to design the web pages, the combination of colors is almost infinite. However, a good index catalogue of color combinations such as the book, “Color Index”, published by Jim Krause (2002), enables designers to find similar solutions easily by looking into the book. From the same point of view, the following 6 categories are defined roughly.

1) Basic is a category in which the houses are transformed by a ninety-degree rotation or a reflection with squares. These transformation rules could be used to create more rigid and symmetric patterns. Also the houses are not densely packed. So they can be used for designs on flat terrain data.
2) Scale is a category used to generate houses with a particular configuration over a wide area on the site. Also the transformation rules may be useful for the creation of road patterns.
3) Overlap is one class in which the transforming rules can be used when a specific area is required to be denser than the rest. It needs to pay attention not to create too dense of a design. Sometimes, it generates very interesting patterns with Scaling like a pattern of streets and roads.
4) Dense category is of housing layouts with dwelling units that are placed very close to each other. They are more useful for housing designs on hilly locations.
5) Extend-transforming rules can be useful when the housing layout is designed in phases. The designer can get an idea of what the houses would look like in the next phase of design.
6) Poor class designs are extreme cases of transforming rules, which do not generate outputs that are feasible for housing layout designs. When it has a single unit, all units reference points are on a line, or all units are crowded in the narrow area, the results are categorized into this class.

5.2 3D Results

Figure 2 shows one housing layout design of the Basic category. The output can be observed from the different angles in real time using the technology of OpenGL. Especially, the function to import 3DS files is useful for the students because it is easy for them to model components in the commercial CG/CAD package.

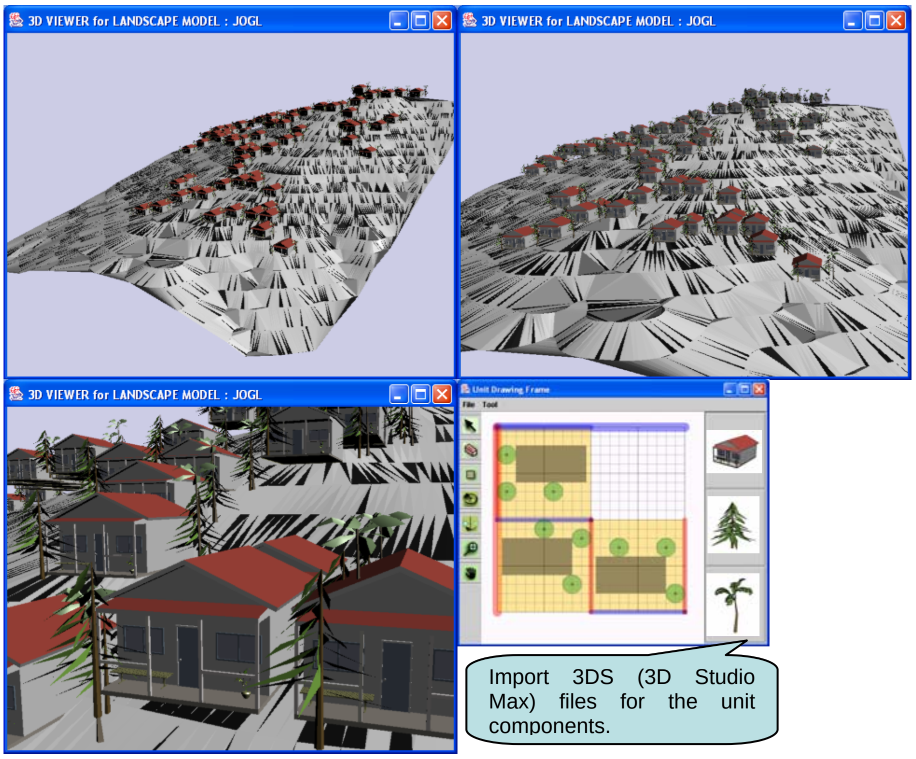

3D VIEWER for LANDSCAPE MODEL : JOGL
3D VIEWER for LANDSCAPE MODEL : JOGL
3D VIEWER for LANDSCAPE MODEL : JOGL
Unit Drawing Frame
File Tool
Import 3DS (3D Studio Max) files for the unit components.

Figure 2 3D-View of Housing Layout Design

5.3 Tips

From the experiments using this tool, more than 10,000 different housing layout designs are examined. Here shows some techniques to find the solutions of housing layout design.

- For increasing the density of trees, there are two ways. One is to add a child unit with only tree components. Another solution is to add trees more in the big units rather than the small ones;
- For increasing/decreasing the housing density, it is better to make the child-unit area bigger/smaller than to add/delete some child units;
- When it is required to create the street patterns together, it sometimes generates good solutions by creating a small gap among child units or to add an overlapping scaled unit only for a street component.

6 CONCLUSION AND FUTURE WORK

This paper offered a new approach for generating housing layout designs in which the distribution of units was done according to fractal patterns. The study provides a means to understand and experiment with the concept of fractals as a design tool.

Based on the case studies, we could demonstrate the potential to use the concept of fractals for generating housing layout designs. Since the rules can be represented in one GUI (Graphical User Interface), the user could handle the generated results among the different settings easily. A variety of alternative designs were made smoothly because the user could reuse the predefined 3D files and XML template. Though we could demonstrate this tool's abilities of generating alternative designs quickly and easily with simple interface and without expert knowledge, there is a need to evaluate this tool as a practical housing layout design tool by comparing it to the other professional approaches and by collaborating with the experts. It also takes a little more time for testing and examining whether the IFS can be valid as a tool for housing layout. Because of the space and time limitations, the evaluation is entrusted to the next opportunity.

Some people point out that this tool ignores the site contents such as slope, orientation, soil types, vegetations, views, etc. in generating patterns. It is important to consider those contents in practical site planning. However, as described in the introduction, one of the objectives is to develop a tool to generate alternative designs using simple inter faces without expert knowledge. In order to integrate the rules corresponding to the site contents, it is required to add more functions to control them. Expert knowledge is also required to control the functions. Therefore, the effects from the site contents were deliberately not integrated in this study. In addition, it is considered that it is better to integrate the rules not in this framework but in the process to assign the house-unit in stop conditions. For example, the direction, size, materials, and other features of house-unit will be controlled when the unit is assigned to a location. Though each unit must be defined as a parametric model for the case, this approach can be the solution and interesting challenge in future.

The followings are considered as areas of future research:

- Find a solution of how to generate housing layout designs when the unmovable infrastructures such as, bridges, streets, etc. exist in the site;
- Find an application for the more practical stage such as a decision-making system for residential house developers;
- Find a more educational use of this application.

REFERENCES

Batty, Michael, and Paul Longley. 1994. *Fractal cities: a geometry of form and function*. London: Academic Press.

Bovill, Carl. 1996. *Fractal geometry in architecture and design*. Boston: Birkhäuser

Kato, N., A. Okano, H. Kanoh, and S. Nishihara. 1999. Building Layout Using a Genetic Algorithm for Virtual Cities, In *The proceedings of System and Information*, vol. J82-D-II No. 10:1766-1774. Tokyo: The Institute of Electronics, Information and Communication Engineers.

Kobayashi, Yoshihiro, and Subhadha Battina. 2004. Generating Housing Layout Designs: Fractals as a Framework. In *The Proceedings of Sigradi 2004*: 221-223. Brazil.

Kurause, Jim. 2002. *Color Index: Over 1100 Color Combinations, CMYK and RGB Formulas, for Point and Web Media*, Ohio, HOW Design Books.

Mitchell, William. J. 1989. *The Logic of Architecture: Design, Computation, and Cognition*. Massachusetts: MIT Press.

Peitgen, Heinz Otto, Hartmut Jürgen, and Dietmar Saupe. 1992. *Chaos and Fractals: New Frontiers of Science*. New York, Springer.

Stiny, George, and William J. Mitchell. 1980. The grammar of paradise: on the generation of Mughul gardens*. Environment and Planning B*, 7: 209-226