# GENERATING HOUSING LAYOUT DESIGNS: FRACTALS AS A FRAMEWORK

**Yoshihiro Kobayashi**
Arizona State University
College of architecture and Environment Design
ykobaya@asu.edu

**Subhadha Battina**
Arizona State University
College of architecture and Environment Design
sbattina@asu.edu

**Abstract**

*This paper introduces a computer-based application for three dimensional (3D) landscape simulations of housing-layout-design using the concepts of fractals with Iterative Function System (IFS). At an early phase of designing housing layouts, the designer assigns specific locations and positions for many house-units on an undeveloped site. From a simple template pattern defined by an XML file, this application generates a wide variety of such layout designs in 3D which consists of multiple residential house-units. Along with the house, the unit can include other design components found in a residential development such as a roads, walls, trees etc. The template defines the transformation rules for the IFS. It includes the information of the geometrical relationships between the stages in the iteration and of the components used in stopping the iteration. The application is formulated, implemented and tested. The results got from the case studies are demonstrated and evaluated.*

## 1. Introduction

Presently, 3D computer graphics (3DCG) modeling and rendering applications are extensively being used for realistic landscape and terrain simulations. However, there are several constraints in such applications when used as a design tool. Also, most CAD applications used during the initial phase of design of housing layouts have basic transforming functions that allow the user to arrange, modify and view houses. Therefore, the process of creating detailed components of the units and understanding the terrain of the site to determine each of their location becomes very tedious. As a part of creating a housing layout, it is also essential to compare as many design alternatives as possible. This can slow the process further. This application expedites and automates this process by generating alternative 3D landscape models. In addition, it allows the user to visualize, understand and test outputs in varying contexts. For example, it helps in observing changes when the number of house-units developed is altered from 500 to1000.

There are several approaches to automate the generation of 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 models having specific characters of the prototypes are required, such as, in case of 3D virtual city modeling. On the other hands, rule-based approaches such as shape-grammar can generate new models by editing each component using rules. It is often used when the constraining rules can be predefined. Based on the constraints applied, the application would allow fast exploration of new, original and efficient design. But, in order to get practical results, these rules need to be controlled in many complicated ways.

By applying the concept of fractals in this tool, it is possible to obtain similar results using simple rules. For example, changing three rules of relation between stages of the IFS can generate 512 different design solutions. Also each rule can also be parameterized very easily. The detail of the units can be continued because each unit's generation can be terminated only by the stop-condition in IFS. The units can also be replaced very easily.

## 2. Fractals

Pioneered by Benoit Mandelbrot in the seventies, the idea of fractals is from medicine to media art and from statistical analysis to mathematics. Because of its geometric nature the idea gained its popularity with designers. Today fractal geometry in architecture is used to analyze complex rhythms of facades of buildings. The research on fractals based on one of the facts that complex large-scale wholes are assembled from tightly interacting subunits. On a larger scale of a city, a part of understanding urban coherence is to understand complex rhythms of fractals. Different components of the urban fabric: streets, shops, etc. connect to generate a successful city, creating an efficient environment. The success of the result depends on geometrical coherence [1]. Also, understanding any natural form of with the help of fractal geometry enables their simulation through computer graphics.

### 3. Methodology

The algorithm used to generate these houses is based on the concept of the IFS to create fractal patterns. These patterns are created by applying specific parameters to polygons in a repetitive manner. The Sierpinsky Gasket is one such example in which the Parent Unit or the main stage in the IFS is transformed into much smaller child units each consisting of similar transformations, creating a fractal. Mathematically, the process can be continued endlessly.

In this tool, each transformation is repeated to create houses on the terrain. The definition of the Parent unit is determined in relationship with the origin in the XY plan of the drawing area.

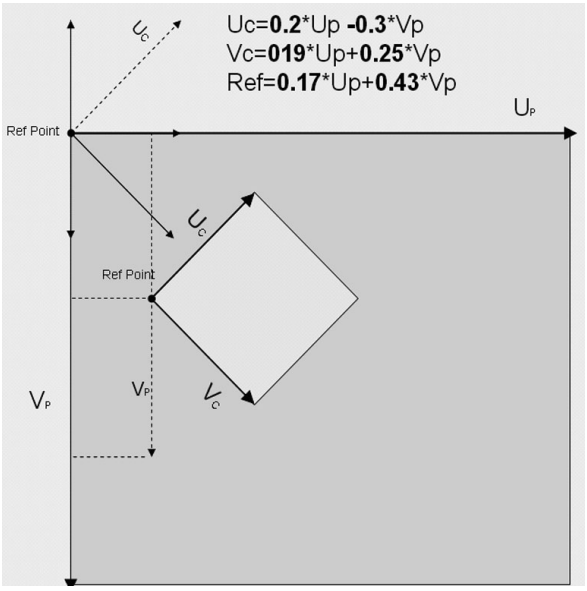


Figure 1 – The Fractal unit

The figure 1 shows how a child element is mathematically referenced to the parent element. A reference point and the inclination of the parent unit are specified by giving the direction and size of two vectors emerging from the reference point. The vector sizes and the directions of the vectors |Vc| and |Uc| is determined with respect to the vectors |Vp| and |Up| respectively. Correspondingly, the child objects are also described with their vector sizes in reference to the parent object. In this application, these formulae apply to all the iteration. The formula below describes the relationship of the vectors child unit with respect to the parent unit's vectors.

**Uc = a * Up + b * Vp**
**Vc = c * Up + d * Vp**
**Ref Point = e * Up + f * Vp**

Where Up & Vp are the directional vectors of the parent unit, Uc & Vc are the directional vectors of the child unit and the RefPoint is the reference point from which the directions and the sizes if the vectors are specified. 'a', 'b', 'c', 'd', 'e' and 'f' are linearly independent constants.

Transformation of the child object is continued until a specified vector length is reached. This helps in limiting the IFS and provides a more realistic output. These vector lengths also determine the sizes of various design elements within the unit. A Part of an XML template file is illustrated below as an example. A class file in the application, which performs the actual creation of three-dimensional units, then reads a similar XML file.

```
<Unit name="Parent">
    <Attribute name="RefPoint" type="Point" valueX="200" valueY="100" valueZ="0" />
    <Attribute name="Udir" type="Point" valueX="300" value Y="0" valueZ="0" />
    <Attribute name="Vdir“ type=”Point" valueX="0" value Y="300" valueZ="0" />
</Unit>
```

Here, the unit is the parent. The Position of the reference point and the directions of the vectors are given in terms of X, Y and Z values. Changing the reference points and the vector angles in the XML file gives out interesting results. Therefore, the template file can be modified to provide various possibilities of outcome. Design elements in the application like the trees or the houses in the algorithm are hard coded to have a specific height.

### 4. System Frameworks

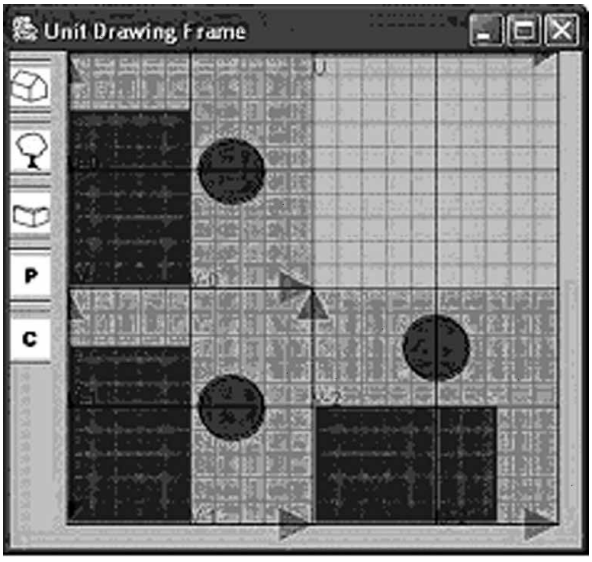


Figure 2 – Screen grab of the graphic area

The application is developed in Java, an object oriented language. The user creates the XML file and imports the file as a process of opening the application. The graphic user interfaces (GUI) consists of two panels -

1. 3D Panel (3DP) to displays the 3D views of the site and the results.
2. Rule Definition Panel (RDP) to define the rules and create/import 3D components used in house-units.

It takes three simple steps for the user to generate the results. First step is to define the unit (U &V vectors and the reference

point) in RDP. The unit could consist of one or more houses. The second is to assign 3D components that are pre-created such as trees, houses, or drive way, etc. The last is to import the site data such as TIN (Triangulated Integrated Network) data in DXF format, and to specify the stop-condition in IFS. The application then generates multiple 3D landscape models of the entire housing layout responding contextually to the topology of the site. Since the terrain is in the TIN format, it consists of triangles with information of the elevation changes in the site. The elevation of each house created in the IFS is aligned with the corresponding elevation of the site. Also certain conditions like reaching the boundaries of the site will also stop the transformations. Hence the houses respond to the shape of the site.

The unit can be defined by the user as the parent and the child element. The figure 2 gives a screen grab of the graphic area in which the user defines the arrangement of houses that could be transformed to create the fractal pattern. The rectangles represent houses in plan. The circles represent the trees. The buttons on the left allow the user to create many such design elements within a housing layout. Each of these units transforms corresponding to the vector length of the parent object.

### 5. Case Studies

The terrain data of Phoenix, Arizona was chosen to experiment with the generation of the houses.

The figure 3 illustrates on example of the output. In table 1 the numbers under the 'Image' column corresponds to the pictures in figure 3. In the second column the numbers indicate the X and Y values of the corresponding Parent and child reference points. The third column gives the direction of the U and the V vectors in terms of X and Y coordinates in xy plane. The Z value as a default is set to zero. The output changes with the values of the reference points & vector directions. Changing the Child-units RefPoint to wider locations from the parent's Refpoint generates locations that are spread out. The opposite is the case when the Refpoints are very closely located. Also by changing the angles of the child elements one can bring out interesting orientations to the houses.

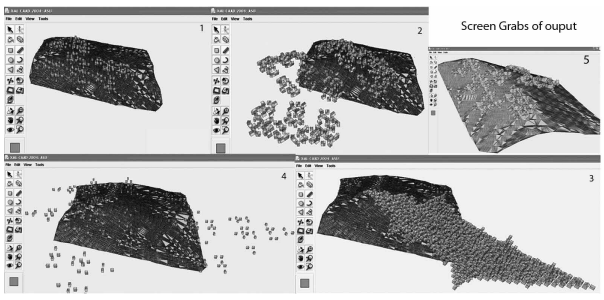

Figura 3 – Screen grab of the Output

Table 1: Description of the output

| Image | Coordinates of reference point X,Y (Z=0) | | Vector directions in terms of X,Y Z=0 | |
|---|---|---|---|---|
| 1 | Parent | 200, 100 | U– 300,0 | V– 0,300 |
| | Child Unit1 | 200, 250 | U – 0,-150 | V – 150,0 |
| | Child Unit2 | 200, 400 | U – 0,-150 | V – 150,0 |
| | Child Unit3 | 350, 400 | U – 0,-150 | V – 150,0 |
| 2 | Parent | 200, 100 | U – 300,0 | V – 0,300 |
| | Child Unit1 | 200,250 | U – 0,-150 | V – 150,0 |
| | Child Unit2 | 200,550 | U – 150,-150 | V – 150,150 |
| | Child Unit3 | 350, 400 | U – 0,-150 | V – 50,0 |
| 3 | Parent | 200, 100 | U – 300,0 | V – 0,300 |
| | Child Unit1 | 100, 325 | U –150,150 | V – 50,-150 |
| | Child Unit2 | 200, 400 | U – 0,-150 | V –150,0 |
| | Child Unit3 | 275, 475 | U – 0,-150 | V – 150,0 |
| 4 | Parent | 200, 100 | U – 300,0 | V – 0,300 |
| | Child Unit1 | 200, 100 | U – 0,150 | V – -150,0 |
| | Child Unit2 | 200, 400 | U – 0,150 | V – -150,0 |
| | Child Unit3 | 275, 475 | U – 150,0 | V – 0,-150 |

### 6. Summary and Future works

Unlike many other probabilistic or rule–based systems which need a great level of expertise, applying fractal concept is a very easy method to get a variety of output. In addition, by using fractals, one can also very easily parametrize the constraining rules. Hence this paper offers an alternative yet simpler option to automate the design process to generate housing layouts. The study provides a means to understand and experiment with the concept of fractals as a design tool. Presently, the case studies demonstrate 3D layouts using only houses and trees. On a larger scale the same concept could be used to involve much more complex elements to generate layouts at an urban level. Also, certain rules in the application allow the houses to overlap one another. In such cases, the overlapping rules may be further refined.

### References


1. Alexander, Ishikawa, Silverstein, Jacobson, Fiksdahl-King, Angel. A pattern language: towns, buildings, construction, New York: Oxford University Press, 1977.
2. Barnsley, M. F. (Michael Fielding), Fractals everywhere, Elsevier Science & Technology Books, Saint Louis, MO, U.S.A., 1988.
3. Batty,Michael & Longley,Paul. Fractal cities: a geometry of form and function, London; San Diego: Academic Press, c1994.
4. Mitchell, William, Liggett, Robin, kvan, Thomas, The Art of computer Graphics Programming Van Nostrand reinhold, Berkshire, England,1987.
5. Peitgen, Jurgens, Saupe, Chaos and Fractals, Springer-Verlag, New York,1992.
6. Salingaros, Nikos, West. B.J. A Universal rule for the distribution of size, Environment and Planning B: Planning and Design 1999, Vol. 26, pg 909-923.