Aug. 2015. Vol. 6, No.4 ISSN 2305-1493
International Journal of Scientific Knowledge
Computing and Information Technology

www.ijsk.org/ijsk

AUTO-SYNTHESIZE LOGICAL 3D LARGE MODEL BASED ON SIMPLE INPUT PRIMITIVE MODEL

[1] KOK-WHY NG, [2] KOK-YAN YONG, [3] MAHDI BABAEI

[1,2] Faculty of Computing and Informatics, Multimedia University, Malaysia.
[3] Faculty of Creative Multimedia, Multimedia University, Malaysia.
E-mail: [1]kwng@mmu.edu.my, [2]sean.kyan92@gmail.com, [3] mahdi.babaei@gmail.com

ABSTRACT

3D modelling is a tedious and time-consuming process in games and entertainment industries. Due to the availability of high performance graphics hardware today, the demand for higher detailed and realistic graphics complicates tremendously the creation of large 3D model such as a city. Several modelling techniques have been introduced to ease the productivity of 3D modelling. However, they only fit for certain models. This paper proposes to auto-synthesize logical and realistic 3D model based on simple input primitive models. The input model will be sub-divided into several parts which the top-most and bottom-most are constrained for rational merging purpose. The body structure is randomly made. Our algorithm is easy to implement and capable to generate variety multifaceted models with only a single touch on a key. Our output is logic and acceptable. It is useful for the artist to have different ways of design.

Keywords: ***Model Synthesis, Procedural Modeling, Modelling by Example, Rendering.***

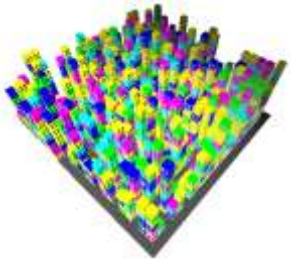
1. INTRODUCTION

Modelling is an important task in Computer Graphics applications. It consumes a lot of time as it requires significant creativity and technical skill from the user. Therefore, model synthesis is introduced as a new approach to 3D modelling that generates a large and complex model based on the simple example model input. Model synthesis can be used to create symmetric models, dynamic models, and models that fit soft constraints with many different objects and environments.

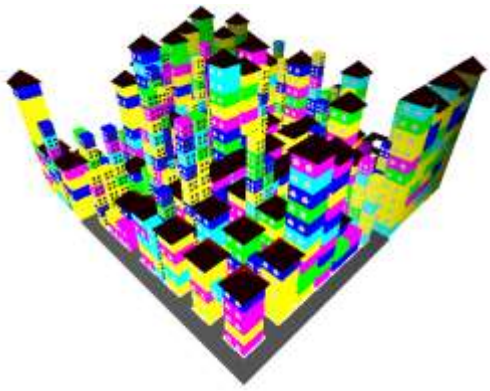
Architectural buildings, landscapes and fractal structures are the crucial models in video games, virtual environments and computer-generated movie (e.g. animations). However, there are problems arise in automated modelling. There are limitations in current 3D computer-aided-design (CAD) and modelling tools in terms of producing complex models and can be cumbersome to use. Modelling architectural buildings, landscapes and fractal structures are often a long and tedious process due to their size and complexity as most exciting visual environments are usually large and intricate.

Model synthesis algorithm accepts a simple 3D model as an input and produces larger complex 3D model that resembles to the input. As model synthesis is closely related to texture synthesis, there have been many techniques developed for synthesizing the texture. Figure 1(a) shows the simple 3D input model from the user and undergoes the 3D model synthesis process to output large and complex models shown in Figure 1(b1 and b2).

(a)



(b1)



(b2)

Figure 1: *Larger and intricate models (b1 and b2) are automatically generated based on the input models (a). In (a), the two models on the left are the roads and two models on the right are the simple block of houses.*

2. RELATED WORKS

Recently, there are wide varieties of techniques been developed and introduced for model synthesis. All of these techniques are designed specifically for certain objects and environments. Models such as plants are usually generated using L-system grammars [10]. Fractals have been used to model terrain [14]. Split grammars have been used to create architectural buildings [13]. However, different designs in each method are only meant to model a specific class of objects.

A Wang Tile [9] consists of non-rotatable color-coded edge square tiles. It provides a framework for generating large form of output based on the small input. The color-coded edges are connected to the same color-coded edges from different square tiles respectively so that it produces a valid periodic tiling of plane [4]. Lloyd's method is an alternative way for creating Poisson disc distributions in order to optimize an initial set of point positions [2]. Voronoi diagram is constructed to satisfy the Poisson disc constraints across the tile boundaries.

Procedural modelling algorithm accepts inputs and produces complex output. The output is able to change and evolve from time to time. Procedural modelling techniques are very diverse and targeted towards a certain type of object and environment based on fractal geometry achieved success modelling natural landscapes [13]. L-systems are a set of rules for updating a given string. L-systems have involved in generation of fractals and mimic certain visual aspects of realistic plant development. L-systems are parallel rewriting systems which can be extended to consider how plants interact with their environment as they grow [7]. Continuous model synthesis technique is implemented by computing an output that has similar connected features and resembles the input. It exploits the connectivity between the adjacent boundary features of the input model [8, 12].

Example-based techniques [3] are widely used for model synthesis. It was inspired by the use of texture synthesis [1, 11] where an example texture accepted as an input and the algorithm outputs a more extensive texture that resembles the example texture [5, 6].

3. PROPOSED METHOD

Different 3D model synthesis project has its own respective approach and solution towards their model creation. There is non-existence of general algorithms or techniques that are able to satisfy different constraints. Therefore, building and road models are chosen in our work. As a result, the program will be able to compute a city of logical and realistic architectural buildings and roads.

3.1 Basic Models

Randomization is the backbone of our program. It requires plenty of logics in order to produce logical output randomly with a touch of a button. Both soft constraints and hard constraints are heavily used in the algorithm. Understanding the environment constraints is relatively important for the program to generate a realistic and functional city with buildings and roads. For instance, a building cannot be built on a road and the road cannot intersect with any of the buildings.

3.2 Shape Analysis

Shape analysis is the first phase that focuses in analyzing the given basic 3D models and categorizes them according to their respective class or category. As mentioned earlier, our work accepts simple model which consists of either a primitive shape or a simple 3D model that can be decomposed and predefined by the user. For example, a robot model can be decomposed into several parts such as the head, the body, the arm, and the leg. These several parts are then identified with an integer. In our work, the buildings and roads are the main 3D models in order to generate the output that looks like a city. Both of the 3D models will be further decomposed into simpler parts.

Generally, buildings are differed by the parameters and dimensions in terms of length, width and height. A standard skyscraper can be further decomposed and predefined into several parts that consist of roof, body and base. However, a twin skyscraper may be bridged in between for pedestrian purposes. These parts will then be further assigned with different integer started from 1, 2, 3 and etc.

A straight road itself is considered the primitive input shape. It is a basic square-like-tile that creates a straight road when two tiles were combined together. Logically, road can be built either vertically or horizontally. Furthermore, roads can be intersected to form an intersection point that makes it a T-junction. Alike building analysis, a standard straight road will be assigned as the default road path, whereas left heading path will be assigned as numeric one, '1', and right heading path will be assigned as numeric two, '2'.

3.3 Constraints Declaration

In order to generate a logical, realistic and structural 3D model city, constraints definition plays an important role in our program. Constraints are mainly categorized into soft constraints and hard constraints. Soft constraints are set of conditions that cause certain variable values to be penalized based on the extent, which conditions on variable that are not satisfied. In short, these set of conditions may not be satisfied and would not affect the logics of the output.

On the other hand, hard constraints are set of conditions that variable values are required and mandatory to be satisfied. The output would be corrupted and non-logical if these sets of conditions were not met. In addition, geometric constraints are used to control the sizes and shapes of the synthesized models. Geometric constraints such as dimensional constraints, algebraic constraints, connectivity constraints, and large-scale constraints are applied in the program besides soft and hard constraints. These constraints will be further explained in the following section.

Soft constraints are set of conditions that remain true unless it contradicts by other conditions. The logics of the output would not be affected even though the set of conditions are not satisfied. In this program, soft constraints are defined for both roads and buildings. For instance, roads can be built at infinite length without violating any conditions. However, interception between two roads could still be accepted if it looks logically. Any stories of buildings are acceptable even though it is almost impossible to build an infinite height of building in reality. However, it should still hold a true statement.

Hard constraints should always be satisfied and no violation of hard constraints is acceptable. For instance, buildings are not supposed to be built on the roads nor buildings could overlap or to be built on top of others resulting an illogical and meaningless output. Besides, rooftop is not acceptable to be a stand-alone building. This is because rooftop itself is not suitable for people to stay in reality. Nevertheless, a base or the levels of buildings should not be built right above the rooftop because rooftop should always stays on top of a building. A connection or pedestrian bridge should be built between two similar heights of buildings. It should not be simply built in the middle of nowhere that does not provide any meaning at all (refer Figure 2).

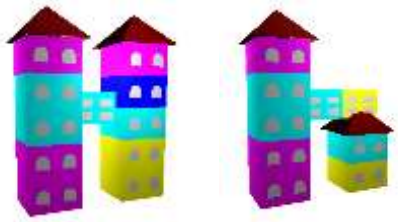


Figure 2: *Logical (left) and illogical (right) building models.*

In our work, all the 3D models are predefined to their dimensions. Roads, building bases, building walls, and rooftops have their own respective width, height and length. Dimensional constraints allow the user to predefine the 3D models dimensions so that the program would not generate any imbalance dimensions of buildings and roads. Roads are built to fit two sizes of cars for a to-and-fro traffic and the ceilings are high enough to fit the average height of an actual man compared directly to the car size. A default building dimension will be used as a guideline to define other 3D models' dimensions.

Some 3D models do not have predefined dimensions. However, it is still important to satisfy the algebraic relationship between 3D models dimensions. For example, the height of a story must be at least twice the car height. The width of the road must be at least twice the width of a car in order to fit two cars for a to-and-fro traffic.

Certain 3D models would look funny and unrealistic if there are unconnected parts.

Connectivity is relatively essential for our work especially the road network. It would be illogical if there is existence of isolated loop of road from the others. Buildings connectivity is also vital. A building should not have any floating or unconnected blocks from the base or a floating rooftop. Furthermore, a connection or pedestrian bridge must always stay connected between two similar buildings. Connectivity constraints declaration is crucial in eliminating all isolated and unconnected roads and buildings.

Before generating the output, a basic floor plan is mandatory for our work to get an idea of how the city would look like. The purpose of these constraints is to ensure the city is built in square blocks accordingly and the buildings are required to be built between roads without any interceptions and overlapping with other 3D models.

3.4 Shape Synthesis

Our program will generate a whole new complex and intricate 3D model based on the basic primitive shapes and the decomposed parts of the 3D models. It will examine the declared constraints and available spaces before constructing the new 3D models on the plane. The purpose of this is to prevent unwanted graphic processing wastage and plane destruction with messy and overlapping 3D models all over the plane.

Once all of the 3D model constraints have been clarified and declared, it will generate the 3D models on a plane. In this stage, an array is initialized and the dimension of the plane is predefined. We apply binary form to represent the 3D models on the plane. Every columns and rows are initialized with number '0', which stands for nothing or nil on the plane. Whenever there is any 3D model to be built on the plane, the particular box number will be set to number '1', which shows that the area of the plane is occupied. The binary representation is used to prevent any interception and overlapping of the 3D models on the plane. The program will generate different types of road randomly (refer Figure 3) meanwhile updating the binary code in the array. It is ready for 3D model generation over the plane with number '0' stored in the array once the road generation process is completed.

Figure 3: *Consistent (left) and inconsistent (right) models dimensions.*

Our program will proceed to model generation in this section. It will gather the information of defined 3D model parts and combine them in a logical yet meaningful intricate 3D model. There are a few important considerations to be pondered about:

a. Constraints of 3D model
b. Dimensions of 3D model
c. Available spaces on the plane
d. Suitable spaces for 3D model on the plane

Once the information of the new 3D model has been collected and analyzed, the program would then finalize the model along with the process of beautifying and detailing. Our program will apply colors or textures onto the 3D model in order to make it looked more realistic. A few additions have been applied to the new 3D models such as windows, rooftops and pedestrian bridge.

4. EXPERIMENT RESULTS

4.1 Model Analysis

A basic road consists of horizontal and vertical orientation. Roads can be categorized into different types of directions. A simple 3D building model could be decomposed into parts such as rooftop and building base. A high rise building could be decomposed into single story without any rooftop, while a landed building could be further decomposed into both single story base and rooftop as shown in Figure 4. The decomposed parts would be further categorized and assigned with a corresponding number.

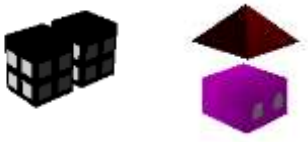


Figure 4: *Decomposed building models.*

4.2 Randomization

As randomization is the crucial part from our work, it is mandatory to generate 3D models randomly in a logical sense. It requires randomization for every complex 3D model in terms of parts and categorizations.

4.3 Model Synthesis

We have applied the foundation of randomization to generate the road. A complicated and logical road network is computed randomly based on the basic road models categorized earlier (refer Figure 5).

Figure 5: *Logical and complicated road network.*

Next, we apply the random functions on decomposed parts of buildings to generate different types of buildings randomly based on the constraints defined (refer Figure 6).

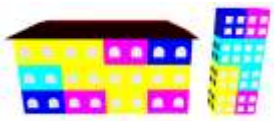
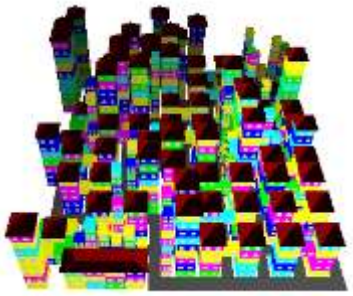


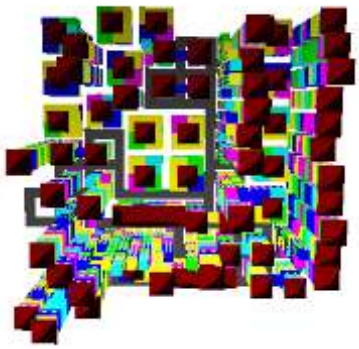
Figure 6: *Computer generated new building models.*

Finally, we applied the constraints and random functions on the buildings to allow different buildings to be built across the city. For instance, there are standard high rise buildings, multiple stories houses, bungalows and connected skyscrapers to be built on the plane without intercepting each other and violating the logics of city building. The models in Figure 7 fulfill the characteristics above.

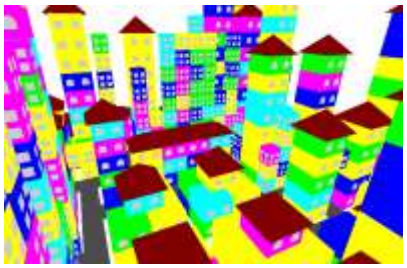
(a)



(b)



(c)



(d)

Figure 7: *(a) Synthesized model 1. (b) Synthesized model 2. (c) Synthesized model at top view. (d) Synthesized model at inner view.*

5. CONCLUSION

We have proposed an algorithm to analyze the given input 3D model, and generate fully randomized, logical and realistic 3D models. This piece of work may assist programmers or designers in creating objects effortlessly and efficiently especially in the field of virtual reality and gaming industries.

6. REFERENCES

1. Bhat P., Ingram S., and Turk G., "Geometric texture synthesis by example," In Proceedings of the Eurographics/ACM SIGGRAPH symposium on Geometry processing, ACM Press, NY, USA, pp. 41-44, 2004.
2. Cohen M. F., Shade J., Hiller S., and Deussen O., "Wang tiles for image and texture generation," ACM Trans. Graph. vol. 22, no. 3, pp. 287–294, 2003.
3. Funkhouser T., Kazhdan M., Shilane P., Min P., Kiefer W., Tal A., Rusinkiewicz S., and Dobkin D., "Modeling by example," ACM Transactions on Graphics (TOG) - Proceedings of ACM SIGGRAPH 2004. vol. 23, no. 3, pp. 652-663, 2004.
4. Grunbaum B., and Shephard G. C., Tilings and Patterns, 1st Edition, W. H. Freeman and Company, 1990.
5. Kwatra V., Essa I., Bobick A., and Kwatra N., "Texture optimization for example based synthesis," ACM Transactions on Graphics (TOG) - Proceedings of ACM SIGGRAPH 2005, vol. 24, no. 3, pp. 795-802, 2005.
6. Legakis J., Dorsey J., and Gortler S., "Feature based cellular texturing for architectural models," In Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques, pp. 309-316, 2001.
7. Lindenmayer A., "Mathematical models for cellular interaction in development I. Filaments with one-sided inputs," Journal of Theoretical Biology, vol.18, no.3, pp. 280-289, 1968.
8. Liu Y. J., Bao F., Li Z.M., and Li H., "3D Model Retrieval Based on 3D Fractional Fourier Transform," International Arab Journal of Information Technology (IAJIT); vol. 10 , no. 5, pp. 421-427, 2013.
9. Lu A., Ebert D. S., Qiao W., Kraus M., and Mora B., "Volume Illustration Using Wang Cubes," ACM Transactions on Graphics (TOG), vol. 26, no. 2, Article no. 11, 2007.
10. Mech R., and Prusinkiewicz P., "Visual models of plants interacting with their environment," In Proceeding SIGGRAPH '96 Proceedings of the 23rd annual conference on Computer Graphics and Interactive Techniques, NY, USA, pp. 397-410, 1996.
11. Merrell P., "Example-based model synthesis," In I3D'07: Symposium on Interactive 3D Graphics and Games, NY, USA, ACM Press, pp. 105–112, 2007.
12. Merrell P. and Manocha D., "Continuous model synthesis," ACM Transactions on Graphics (TOG) - Proceedings of ACM SIGGRAPH Asia 2008, vol. 27, no. 5, Article no. 158, 2008.
13. Muller P., Wonka P., Haegler S., Ulmer A., and Gool L. V., "Procedural modeling of buildings," ACM Transactions on Graphics (TOG) - Proceedings of ACM SIGGRAPH 2006, vol. 25, no. 3, pp. 614–623, 2006.
14. Musgrave F. K., Kolb C. E., and Mace R. S., "The synthesis and rendering of eroded fractal terrains," ACM SIGGRAPH Computer Graphics - Special issue: Proceedings of the 1989 ACM SIGGRAPH conference, volume 23, no. 3, pp. 41 – 50, 1989.