Procedural Generation

Your Thesis subtitle

Artur Alkaim, arturalkaim@tecnico.ulisboa.pt

Universidade de Lisboa

Abstract. Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

Keywords:
Procedural Generation, City, Modeling

1 Introduction (2/3pgs)

As technology evolves and people get new and more powerful devices, they want to take advantage of that with more detailed and complex contents to have more realistic experiences. And this is observable in the graphic contents. With more de definition of the screens and the computational power of the machines beating records, the graphic content have to follow up that characteristics in quantity as well as in quality. The issue is that the manual content generation takes a long work time from artists to achieve this quality. The obvious answer to this problem is to contract more artists to each project to increase the production, but experience have shown that this solution is not scalable, that means that double the number of artists working in a project will not double their overall productivity. And this solution have a big impact on costs, that would take immediately out of the market new producers with less resources.

There is a area of research that tries to solve this problem with *Procedural Content Generation*, or *Procedural Generation*.

Procedural Generation is the algorithmic generation of content in stead of the usual manual creation of content. This can be applied in almost all forms of content, but is mostly used in graphics creation and sound (music and synthetic speech). "The key property of procedural generation is that in describes the data entity, be it geometry, texture or effect, in terms of a sequence of generation instructions rather than as a static block of data."[2] This allows anyone with less resources to produce high detailed, and high quality content.

2 Objectives (1pg)

Clearly explain the project objectives.

The objectives of this project is to study the existing approaches to Procedural Generation and to develop a new tool that will be able to generate large amouts of forms efficiently and according to a particular architectural style .

3 Related Work (17pgs)

3.1 CityEngine [4] [3]

It's a three-dimensional (3D) modeling software developed by Procedural Inc. (now part of the Esri RD Center). It's specialized in the generation of 3D urban environments. With the procedural modeling approach, CityEngine enables an efficient creation of detailed and large-scale 3D city models with a lot of control from the user.

RoadNetwork The first part to procedurally generate a city is to create a road network to become a backbone of the city and provide an overall structure. For that, CityEngine receives as input maps such as land-water boundaries and population density. From that input a network of highways is created to connect the areas off high density population, and small roads connect to the highways. This growth process continues until the average area of each lot is the desired one. The system have a default value, bat it can be set by the user to a different one.

To implement this growth process, it's used an L-System, that computes the road network.

The Figure 1 shows the evolution of this process in a map of Manhattan. The first two on the top shows the process in different phases during the process, the middle line is the result of the process and the bottom line is the real map of Manhattan for comparison.

Buildings To implement the generation of buildings, they created the CGA Shape.

CGA Shape is a Shape Grammar that was introduced by Pascal Muller, Peter Wonka and others, in a paper called "Procedural Modeling of Buildings". It is defined as "a novel shape grammar for the procedural modelling of CG architecture, produces building shells with high visual quality and geometric detail." To do so, this grammar uses a group of well defined production rules.

This tool allows the user to model buildings with an high control and in different ways. It can be done by text, writing production rules from a shape grammar or with a visual language like Grasshopper 3D, that is nice for simple works but it's impossible to work with a slightly more complex work.

Mass Modeling To model a building the first step is to create a mass model of the entire building by assembling basic shapes. With scaling, translation rotation

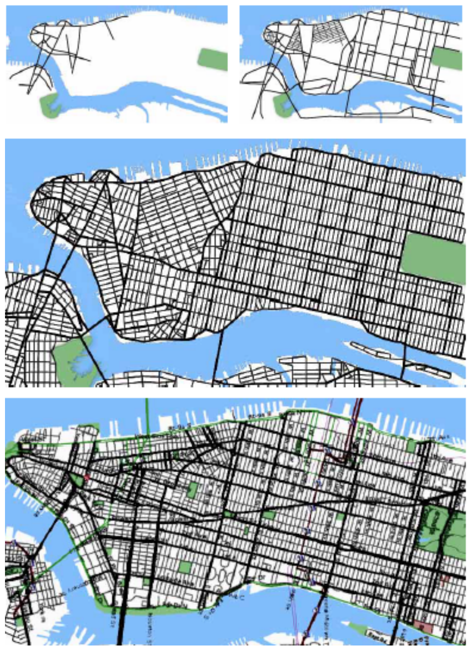

Fig. 1. Road Map growth

and split applied to basic shapes namely I, L, H, U and T as shown in the following picture.

The next step is to add the roof, from a set of basic roof shapes or general L-Systems.

After that, with the application of the grammar rules in the created mass, it's possible to create complexity to the level that is desired, being able to produce high complex buildings like the one in the following picture.

Cities The result can be an city like Figure 2, with approximately 26000 buildings.

City Engine results can be imported by Maya, to achieve better results. Like the Figure 3, that represents a virtual Manhattan.

3.2 Undiscovered City

In the "Real-time Procedural Generation of Pseudo Infinite Cities" paper, from Stefan Greuter et al. presented a system that generates in Real-time pseudo infinite virtual cities which can be interactively explored from a first person perspective. "All geometrical components of the city are generated as they are encountered by the user." As shown in the following image only the part of city that is inside the viewing range is generate.

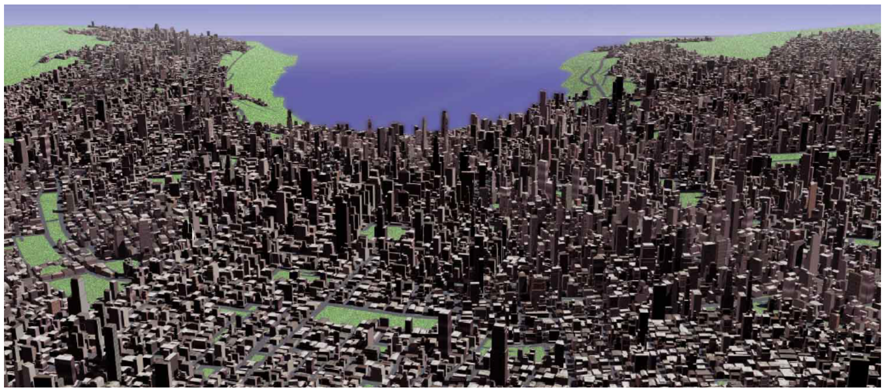

Fig. 2. City with approximately 26000 buildings.

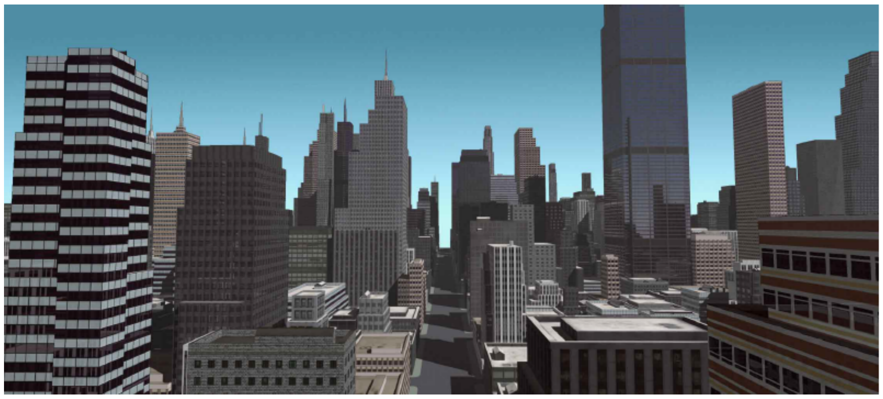

Fig. 3. City rendered with Maya.

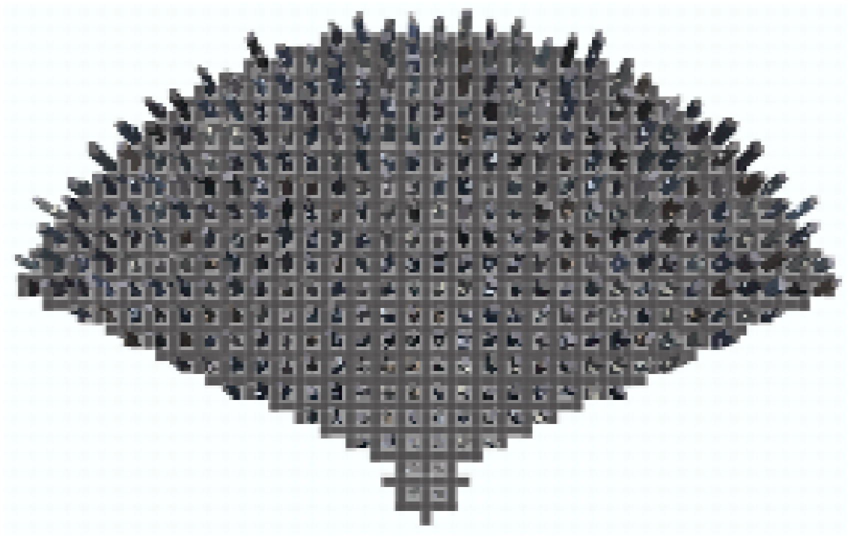

Fig. 4. buildings

Road Network The system uses a 2D grid that divide the terrain into square cells. The cells represent proxies for the content that will be procedurally generated. Before the content of each cell is generated, the potential visibility of it is tested, and after that, only the visible cells are filled with content.

After that the roads are created in a uniform grid pattern. This grid does not feel very natural, and in the continuation of the work, this system evolved into a more realistic one with the join of some of the grids to create a less uniform distribution of the buildings.

Buildings To compute the form and appearance of each building, it is used a "single 32 bit pseudo random number generator seed. The random sequence determines building properties such as width, height and number of floors." Similar sequences of number result in similar buildings. To avoid that, it is used a a hash function to convert each cell position into a seed.

To generate a building is first is generated a floor plan. To do so, it's randomly selected and merged a set of regular polygons and rectangles, then this is extruded. This is an iterative process, that creates sections from the top to the bottom, by adding more shapes to the the initial shape and extruding as shown in the Figure **??**.

Starting from the left, first there is a simple polygon, that is merged with a rectangle and after extrusion, forms the first block that will be the top of the building. After that, another extrusion is made to generate the next block. After that is merged a rectangle to the floor shape and generated a new block and so on.

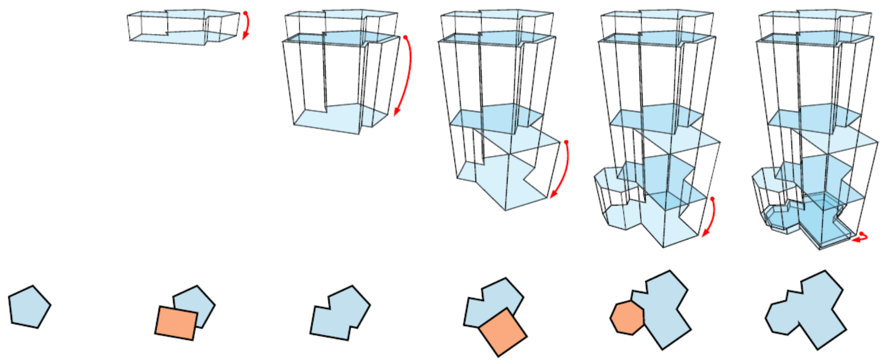

Fig. 5. buildings

3.3 CityGen

[1]

CityGen it's an interactive system that aims to "rapidly create the urban geometry typical of a modern city". The users can interact and control the generation process. The system, like others, is able to generate road networks that act as foundations to the model. It also can generate buildings but can not achieve the complexity and realism of other systems.

Road Network CityGen divided this problem in two steps. First the generation of the "Primary Road Network", and after that, the "Secondary Road Network". This two steps use different methods to generate the roads. Undirected planar graphs are used to represent all roads. Two graphs for the Primary roads and one for each zone to store the secondary roads.

Primary Road Generation

The primary road network uses two graphs, one high level graph that correspond directly to the primary road intersections. It represents the topological structure of the city by it's primary roads, and connections between them. The user is allowed to manipulate this high level graph, to change the high level structure ("topography of the primary road network") of the city . There is also the low level graph that is generated from the other one, and defines the real path that the roads have in the terrain. It have the same nodes as the first graph and many more, that indicate the points on the terrain which the road passes. To generate the low level graph it is used "sampling, plotting and interpolation processes".

Secondary Road Generation

The author defined city cells as districts, that are the areas of terrain that are enclosed by primary roads. The secondary road network is generated inside this cells using a growth based algorithm similar to the L-Systems technique.

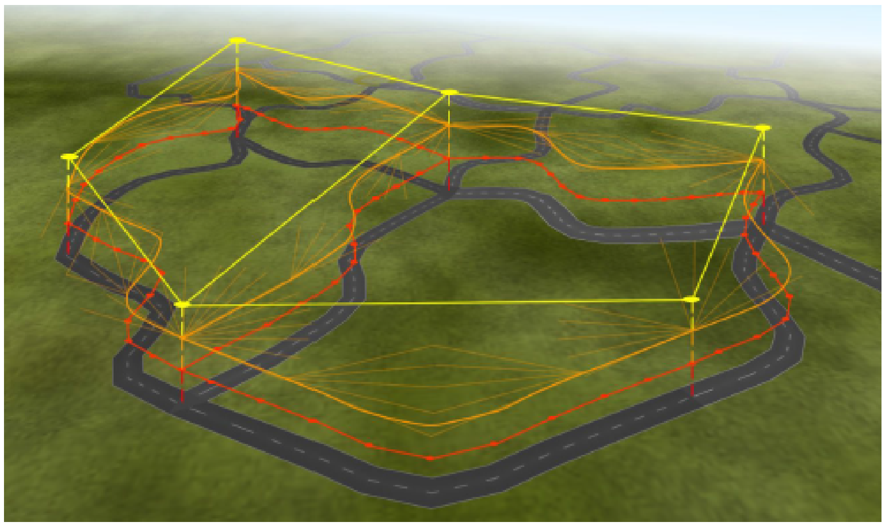

Fig. 6. The lighter graph is the High level graph, and the evolution to the darker Low-level graph

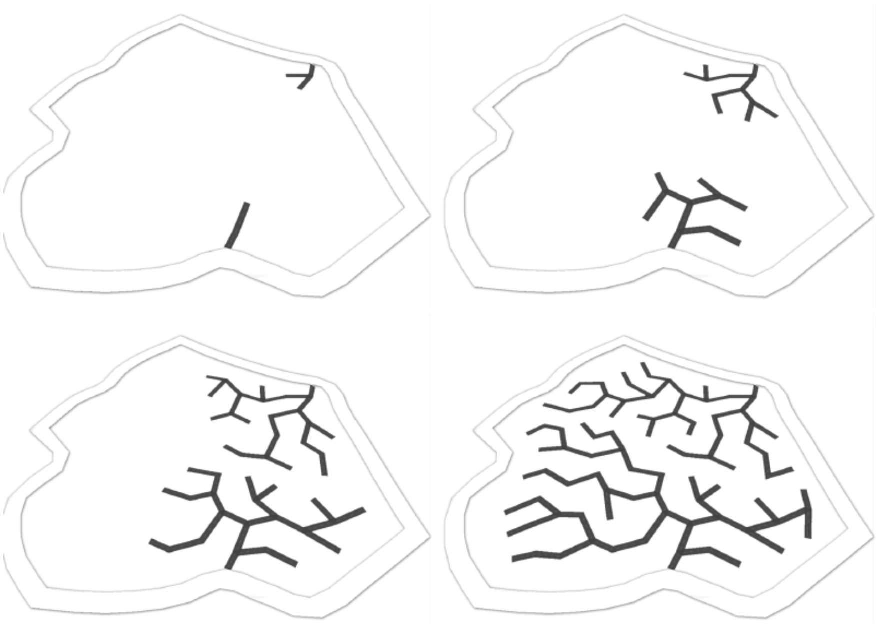

Fig. 7. The lighter graph is the High level graph, and the evolution to the darker Low-level graph

Buildings This system generates buildings also. Each building is created in lots that are identified after the extraction of enclosed regions, called blocks, from the secondary graph. Lots that don't have direct access to the roads are excluded.

Based on the type of block and building footprints are created. With that building geometry is generated by extruding the footprint. The height of each is determined by a height parameter and a noise factor that can be also manipulated. A block is shown in the Figure 8, with only primitive shapes.

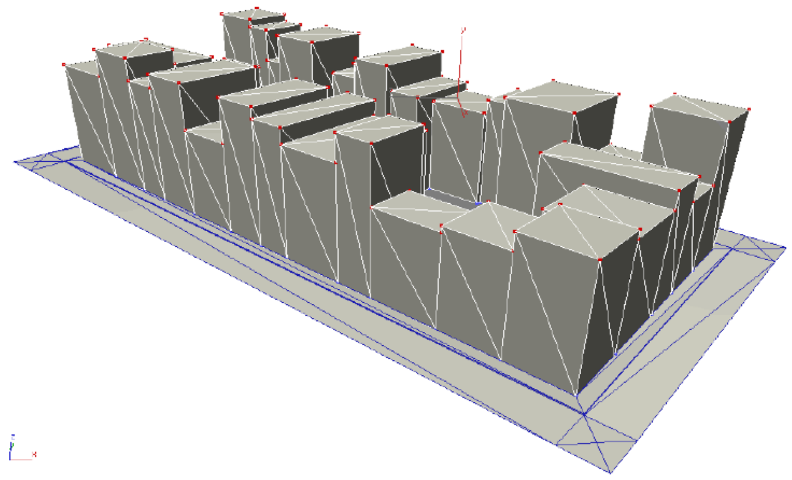

Fig. 8. caption

With this primitive shapes, CityGen uses "advanced materials with shaders to simulate additional geometry".

3.4 Scene City:

[?]

Scene City is an addon for the 3d application Blender.

It varies the colors, materials and procedural elements enough to produce a reasonable approximation of a city at a distance, but realism is not a goal, and it does not operate on real data at all. (http://vterrain.org/Culture/BldCity/Proc/)

3.5 ghostTown:

http://kilad.net/site/ http://www.kilad.net/GTForum/

Its a script plugin for 3DS Max from Autodesk. Ghost Town is a script that procedurally generates cities and urban environments in only a few clicks, and features a number of options. Such as low or high poly buildings, road layouts, vehicles, trees, facades and an easy to use material and texture system. (http://cgi.tutsplus.com/tutorials/create-a-detailed-city-with-3d-studio-max-ghost-town–cg-10090)

3.6 Skyscraper:

http://www.skyscrapersim.com/index.shtml Standalone

Skyscraper aims to be a fully-featured, modular, 3D realtime building simulator, powered by the Scalable Building Simulator (SBS) engine. The main feature SBS provides is a very elaborate and realistic elevator simulator, but also simulates general building features such as walls, floors, stairs, shaftwork and more. Many more things are planned, including gaming support (single and network multiplayer), and a graphical building designer. Skyscraper is written in C++ and uses the OGRE graphics engine, Bullet for collisions and physics, FMOD for sound, the wxWidgets GUI library, and is multiplatform. The current versions aim for a future 2.0 release. from the official site.

3.7 Blended Cities:

http://jerome.le.chat.free.fr/index.php/en/city-engine/ Addon for Blender

Blended Cities is an open-source city generator for Blender. it allows to create quickly a large amount of streets and buildings, with various shapes. B.C. fights against squared things : curved streets and odd or cylindric buildings can be created simply, so you can create old towns, not only modern cities.

4 Architecture (2/3pgs)

Your proposed architecture. Can have lots of pictures and bullet points so it is easy to understand.

5 Evaluation (1/2pgs)

Explain how you are going to show your results (statistical data, cpu performance etc). Answer the following questions:

– Why is this solution going to be better than others.
– How am I going to defend that it is better.

6 Conclusions

Wrap up what you wrote.

A Appendix

Appendix files and refs will go here. Such as your thesis work scheduling.

A.1 Work Scheduling Example

Simple work schedule is presented in Table 1. You can do something more fancy link a Gantt chart or whatever.

Table 1. Work Scheduling

Month	**Work**
February	Do Stuff
February	Do Stuff
March	Do Stuff
April	Do Stuff
May	Do Stuff
May	Do Stuff
June	Do Stuff
July	Do Stuff

References

1. George Kelly. An Interactive System for Procedural City Generation. 2008.
2. George Kelly and Hugh Mccabe. A Survey of Procedural Techniques for City Generation.
3. Pascal Müller, Peter Wonka, Simon Haegler, Andreas Ulmer, and Luc Van Gool. Procedural modeling of buildings. *ACM SIGGRAPH 2006 Papers on - SIGGRAPH '06*, page 614, 2006.
4. Yoav I. H. Parish and Pascal Müller. Procedural modeling of cities. *Proceedings of the 28th annual conference on Computer graphics and interactive techniques - SIGGRAPH '01*, pages 301–308, 2001.