Procedural Generation of Infinite Cities

Jiri Danihelka† and Jiri Zara

Czech Technical University in Prague
Faculty of Electrical Engineering

Abstract

We present a novel technique for generation of pseudo-random infinite cities in a real-time. The generated cities can have arbitrarily oriented streets and building blocks can be arbitrarily shaped. The shapes of city buildings are determined using a pseudo-random generator that uses building coordinates as the initial generator seed.

Categories and Subject Descriptors (according to ACM CCS): I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism —Virtual reality, Hidden line/surface removal

1. Introduction and related work

City environments are generally complex, because they are both detailed and huge. To save storage space or Internet bandwidth procedural generation can be used. Buildings can be generated from their lots on-line when the buildings are in the view frustum using a grammar. Currently the most advanced approach for procedural building generation was published by Müller et al. [MWH*06] in 2006. The lot and street geometry can be also generated procedurally. First such algorithm for finite cities was published by Parish and Müller [PM01].

In 2003 Geuter at al. [GPSL03a, GPSL03b] presented an algorithm for generating infinite cities in regular rectangular grid. In their approach the street network has to be aligned with main axis and all building lots must have the same square shape and size (see figure 4). According to the viewing frustum the visible buildings are determined and procedurally generated. Each building lot gets an integer number according to its coordinates using a hash function. This number is used as a seed for the pseudo-random building generation of that building lot. The generated buildings are saved into cache to save system resources.

In this paper we present a novel algorithm for generation city street network for infinite cities. Our approach is the first that can have the streets arbitrarily oriented and the street network is not periodical.

† danihjir@fel.cvut.cz

2. Algorithm

Our algorithm follows the standard city-generation workflow proposed by Parish and Müller [PM01] that was initially specified for finite cities. The workflow starts with the city streets network and the buildings are generated afterwards. (Workflow phases: 1. street network, 2. building blocks/lots, 3. building geometry) We primarily describe our approach in generation of the city street network, because the other phases remains almost the same as in the case of finite cities. When working with infinite structures we assume computing only those values that are needed due to intersection with the view frustum.

Input:
view frustum, optional generator parameters
Output:
all parts of infinite street network that intersects with the frustum; The street geometry has to remain consistent in case of multiple overlapping queries.

2.1. Construction of the infinite network

Step 1: Suppose we have a plane dedicated for the city generation. Create a regular infinite square grid on the plane. (The segment length of the grid should be configurable generator parameter. Let's denote the length as d.) The grid will divide the infinite plane space into squares. Each of these squares has X and Y coordinate (positive or negative). Now we can use a hash function to assign a generator seed from the coordinates for each square.

Step 2: Using the seed create a pseudo-random generator. Generate a pseudo-random position inside the square with uniform distribution and place a street network node on that position. Those nodes represent crossings of the future streets. (see figure 1)
Step 3: Now it is necessary to create connection between the nodes using streets. We have to carefully choose such method that need only local surrounding of the nodes that are processed, because all nodes cannot fit into the memory. Delaunay triangulation has such properties. A triangle belongs to the Delaunay triangulation if and only if there are no other nodes inside its circumcircle. Delaunay triangulation also maximizes minimal angles which is a pleasant feature.

The described street network has the following properties:
Lemma 1: Any circle with radius bigger than $\sqrt{2}d$ has a node inside it. **Proof:** Such big circle has at least one whole grid square inside. Therefore it contains also the node corresponding to that square.
Lemma 2: No triangulation edge may be longer than $2\sqrt{2}d$.
Proof: Every edge has to be a part of a triangle with circumcircle radius smalled than $\sqrt{2}d$.

2.2. Construction of a finite part of the network

Step 1: Take the view frustum and enlarge it by $2\sqrt{2}d$ in all directions.
Step 2: Take all squares that intersect with the frustum and generate their corresponding nodes. Filter out the nodes that are not inside the extended frustum.
Step 3: Create Delaunay triangulation on those nodes. Lemma 2 guarantees that we do not miss any edge that intersects with the original frustum. Filter out edges that are outside the original frustum and nodes with no edges.

Now we have acquired the part of the infinite street network that is visible in the view frustum.

3. Implementation

To verify our approach we have implemented a Silverlight application (.Net equivalent of a Java applet) that interactively generates infinite street networks according to generator parameters. The generated street network can be imported with other tools used for city modeling like CityEngine [Pro] (see figure 2) and converted to polygonal model (see figure 3). The buildings block are pseudo-randomly subdivided according to Paris and Müller [PM01]. For generation of building geometry we used pseudo-random procedural approach based on grammars that was described by Müller et al. [MWH*06].

We currently do not have an implementation for on-line generating of infinite cities, however we do not expect any principal problems there. The main issue would be on-line generation of buildings, because current tools for procedural modeling of buildings do not support extensions or building generation on demand from other programs. Such features should appear in future versions of CityEngine [Pro].

4. Conclusion

We have created an algorithm for generation of infinite pseudo-random non-periodical arbitrary-oriented street network that can run on-line in real-time. Our appearance of an infinite city looks more realistic than in previous approach made by Geuter et al. [GPSL03a, GPSL03b] (compare figures 3 and 4).

5. Future work

Currently we work only with cities on a flat terrain. We believe that our approach can be combined with existing techniques for procedural terrain generation to create more realistic infinite cities.

Our generator allows only limited customization and it is hard to achieve a look similar to a particular real city. Aliaga et al. [AVB08] have presented an algorithm for synthesizing urban layouts by example. We will try to combine their approach with ours to automatically generate infinite cities with structure that matches to maps of existing cities.

On-line real-time generated cities have to deal also with a lot of other issues like occlusion detection or level-of-detail of procedurally generated models. The main problem is in the missing model post-processing that is almost impossible to do for infinite models. These issues have not yet been adequately addressed so far.

6. Acknowledgement

This work was supported by the MSMT under the research program LC-06008 (Center for Computer Graphics) and by the Grant Agency of the CTU Prague, grant No. SGS10/291/OHK3/3T/13.

References

[AVB08] ALIAGA D. G., VANEGAS C. A., BENES B.: Interactive example-based urban layout synthesis. In *ACM SIGGRAPH Asia 2008 papers* (Singapore, 2008), ACM, pp. 1–10. 2

[GPSL03a] GREUTER S., PARKER J., STEWART N., LEACH G.: Real-time procedural generation of 'pseudo infinite' cities. In *Proceedings of the 1st international conference on Computer graphics and interactive techniques in Australasia and South East Asia* (2003), GRAPHITE '03, ACM. 1, 2, 3

[GPSL03b] GREUTER S., PARKER J., STEWART N., LEACH G.: Undiscovered worlds–towards a framework for real-time procedural world generation. In *Fifth International Digital Arts and Culture Conference, Melbourne, Australia* (2003). 1, 2, 3

[MWH*06] MÜLLER P., WONKA P., HAEGLER S., ULMER A., VAN GOOL L.: Procedural modeling of buildings. *ACM Transactions on Graphics (TOG) 25*, 3 (2006), 614–623. 1, 2

[PM01] PARISH Y., MÜLLER P.: Procedural modeling of cities. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques* (2001), ACM, pp. 301–308. 1, 2, 3

[Pro] PROCEDURAL: CityEngine. http://www.procedural.com. 2

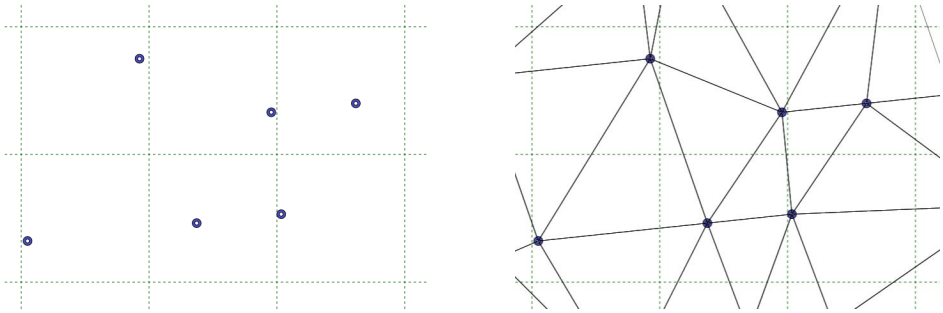

Figure 1: *Left: Randomly generated points in an infinite grid; Right: Delaunay triangulation that forms the streets*

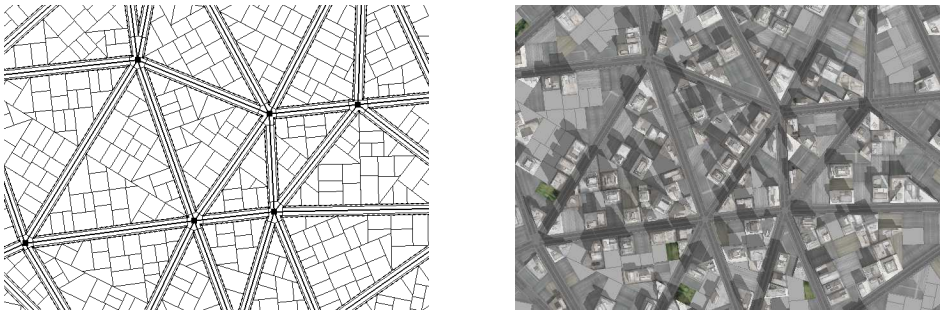

Figure 2: *Left: Building blocks/lots generated according Parish and Müller [PM01]; Right: Generated city - view from above*

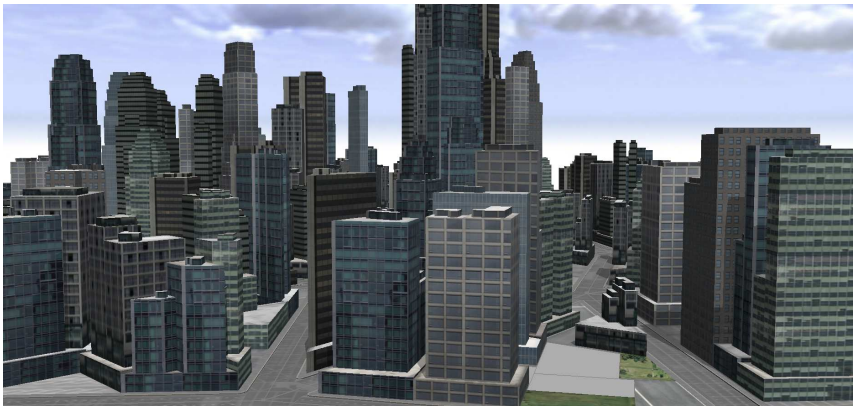

Figure 3: *Our approach displayed from the street level – real-time rendering without scene post-processing, generated offline*

Figure 4: *Previous approach in infinite-city rendering published by Greuter et al. [GPSL03a, GPSL03b] displayed from the street level – real-time rendering, generated online; Note the regular rectangular shape of the street network.*