

Procedural Modeling of Cities

Proseminar-Ausarbeitung von

Robin

An der Fakultät für Informatik
Institut für Visualisierung und Datenanalyse,
Lehrstuhl für Computergrafik

November 23, 2015

Contents

1	Overview	1
1.1	Notable documents	1
1.2	General concepts and methods	1
1.2.1	Procedural modeling	1
1.2.2	Lindenmayer Systems (Prusinkiewicz and Lindenmayer 1990)	1
1.2.2.1	Simple L-Systems	1
1.2.2.2	Parametric L-Systems (Hanan 1992)	2
1.2.2.3	Applications in procedural city modeling	2
1.2.3	Tensor fields (Chen et al. 2008)	3
1.2.4	Gradient Noise (Kelly and McCabe 2006)	3
2	Procedural City Modeling	5
2.1	Input parameters (Parish and Müller 2001, ch. 2)	5
2.1.1	Geographical maps (Elevation maps, Land/water/vegetation maps)	5
2.1.2	Sociostatistical maps (Population density, Zones, Street patterns)	5
2.2	Generating a street network	6
2.2.1	Separate street types	6
2.2.2	Global goals and local constraints	6
2.2.3	Road patterns, (see Parish and Müller 2001, sec. 3.2.2)	6
2.2.4	Constraints and solving conflicts	6
2.2.5	Approach using tensor fields from (Chen et al. 2008)	7
2.3	Splitting areas into building blocks	7
2.4	Computation time	7
2.5	Street network generation approach overview	8
2.5.1	L-system based approach by Parish and Müller (2001)	8
2.5.2	Approach using agents by Lechner, Watson, and Wilensky (2003)	8
2.5.3	Approach from Citygen by Kelly and McCabe (2007)	9
2.5.4	Approach using tensor fields by Chen et al. (2008)	9
2.5.5	Time-based approach by Weber et al. (2009)	10
2.5.6	Approach by Lipp et al. (2011)	11
2.6	Architecture generation approach overview	11
2.6.1	Approach by Parish and Müller (2001)	11
2.6.2	Approach by Wonka et al. (2003)	12
2.6.3	Approach by Müller et al. (2006)	12
2.6.4	Other Approaches	12
	References	13

1. Overview

1.1 Notable documents

The first popular paper about procedural city modeling is from Parish and Müller (2001), which contains a general approach to modeling of the street network and building architecture and is cited in nearly every subsequent document. It is also the basis for (“Esri CityEngine | 3D Modeling Software for Urban Environments” 2015), a professional software application for semi-automated city modeling. Wonka et al. (2003) are the first to describe architecture modeling more precisely. Weber et al. (2009) is the only group using a complicated time based simulation, the others use recursive algorithms or grammars. Vanegas et al. (2010) and Kelly and McCabe (2006) give an extensive overview over most of the other papers. (“Procedural City, Part 1: Introduction. Twenty Sided” 2015) is a detailed article describing an implementation in OpenGL.

1.2 General concepts and methods

1.2.1 Procedural modeling

Procedural modeling is a general term for creating graphics or models automatically or semi-automatically from an algorithm or a set of rules and a pseudorandom number generator.

1.2.2 Lindenmayer Systems (Prusinkiewicz and Lindenmayer 1990)

L-systems are a popular tool for all kinds of procedural modeling, because they allow the description of the generation algorithm as a set of rules.

1.2.2.1 Simple L-Systems

As defined by Prusinkiewicz and Lindenmayer (1990 ch. 1), an L-System is a formal grammar defined as $G = (V, \omega, P)$, where

- V is the alphabet
- $\omega \in V^+$ is the initial word, called the *axiom*
- $P \subset V \times V^*$ is a set of production rules that each map from one letter to a sequence of letters.

Letters that do not have production rules are assigned an implicit identity rule $a \rightarrow a$. Those are called terminal symbols, because once they are reached the letter will stop changing.

In general, an L-system can be deterministic or non-deterministic. It is deterministic when there is exactly one rule for each letter in V .

In contrast to normal formal grammars, the rule application in L-systems is simultaneous. L-systems can be finite, meaning they that after some number of iterations the only matching rules are $a \rightarrow a$. When using non-finite L-systems, rule application is stopped when some condition is reached, for example when a specific number of iterations is reached or when the resulting changes become insignificant.

L-systems can be context-sensitive or context-free. Hanan (1992 ch. 2.2) defines (m) L-systems as L-systems where each rule can access m symbols to the left. The definition of (m, n) L-Systems then has the context of m letters to the left and n letters to the right. The classical Lindenmayer system is thus a 0L-system.

1.2.2.2 Parametric L-Systems (Hanan 1992)

Parametric L-Systems are an extensions to L-Systems allowing the incorporation of arbitrary functions into L-Systems. Each letter can have assigned *parameters* which are real numbers, and can be combined with variables and regular arithmetic operators like $+, *, \geq, \dots$. Rules can then also be conditional on the parameter. As an example, the rule

$$A(t) : t > 5 \rightarrow B(t + 1)CD(t \wedge 0.5)$$

replaces $A(t)$ with $B(t + 1)CD(t \wedge 0.5)$ if and only if $t > 5$.

Additionally, external functions can be called from these rules.

1.2.2.3 Applications in procedural city modeling

Originally used for plant modeling, L-systems can be applied to more complex problems with the above extensions. They are used extensively by Parish and Müller (2001) for creation of the road network and modeling of building architecture, though Sean Barrett (2008) shows the road network L-system can easily be replaced by a simpler algorithm using a priority-queue.

Apart from Coelho et al. (2007), the newer relevant documents avoid L-systems, replacing them with custom algorithms or regular grammars in both modeling of the architecture and of the street network for various reasons:

With regard to the application of L- systems to buildings, we have to consider that the structure of a building is fundamentally different from the structure of plants (or streets)—most importantly, a building is not designed with a growth-like process, but a sequence of partitioning steps. – (Wonka et al. 2003)

An L-system seemed inapt because of the parameter bloat that would result from all the specific exceptions and particularities that may come with a given culture. – (Lechner, Watson, and Wilensky 2003)

While [the approach by Parish and Mueller] creates a high quality solution, there remains a significant challenge: the method does not allow extensive user-control of the outcome to be easily integrated into a production environment. – (Chen et al. 2008, sec. 1)

We chose not to embed the expansion in an L-system framework to make the implementation more efficient. – (Weber et al. 2009)

While parallel grammars like L-systems are suited to capture growth over time, a sequential application of rules allows for the characterization of structure i.e. the spatial distribution of features and components [Prusinkiewicz et al. 2001]. Therefore, CGA Shape is a sequential grammar (similar to Chomsky grammars) – (Müller et al. 2006, sec. 2)

1.2.3 Tensor fields (Chen et al. 2008)

As defined by Chen et al. (2008), a tensor field is a continuous function from every 2-dimensional point \mathbf{p} to a tensor $T(\mathbf{p})$. A tensor is defined as

$$T(p) = R \begin{pmatrix} \cos 2\theta(p) & \sin 2\theta(p) \\ \sin 2\theta(p) & -\cos 2\theta(p) \end{pmatrix}$$

with $R \geq 0$ and $\theta \in [0, 2\pi)$.

There are various basis fields, such as the radial pattern (seen below the river in fig. 2.7) and the constant directional pattern (above the river in fig. 2.7). These are combined into a single pattern by adding them together while scaling them using exponential fall-off from a given center point (Chen et al. 2008, ch. 5.2).

Tensor fields can be used to create a street network, based on the observation that most street network have two dominant directions (Vanegas et al. 2010, 30–31). This also allows easy visualization of the flow direction beforehand. Chen et al. (2008) use them in a graphical user interface to allow interactive modification of the input parameters.

1.2.4 Gradient Noise (Kelly and McCabe 2006)

Gradient noise is created from a map of interpolated random gradients, creating a smooth-looking bumpy surface. It is used in many aspects of content generation in computer graphics to create nature-like effects. It can be used to create or enhance textures and generate terrain (“Terragen 3” 2015). The simple noise is smooth, but can be summed in multiple scales to create fractal noise as seen in fig. 1.1.

A commonly used type of gradient noise is Perlin noise (Perlin 2002), which is easy to calculate and gives good results.

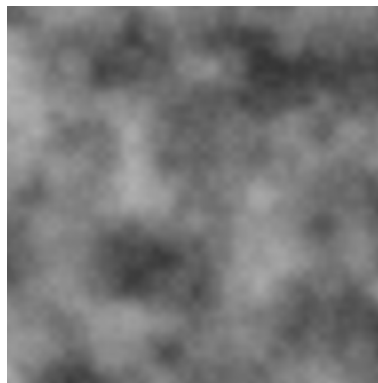


Figure 1.1: “Perlin noise rescaled and added into itself to create fractal noise.” (“File:Perlin.png” 2012)

In the context of procedural cities, noise is mostly used to automate the input parameters like the terrain height map.

2. Procedural City Modeling

The following description is mostly based on Parish and Müller (2001). Differences to other papers are also mentioned, with more detailed descriptions in sec. 2.5.

2.1 Input parameters (Parish and Müller 2001, ch. 2)

Most city generation methods have some form of user input. In (Parish and Müller 2001), the input is a set of image maps containing elevation, water and population density plus some numbers like city block size or maximum bridge length. All further steps are completely automatic. Chen et al. (2008) use similar input maps, but the initial street flow is created based on the map boundaries. The user can then modify the tensor field to change the resulting street graph as seen in (Peter Wonka Research 2008).

Kelly and McCabe (2007) let the user draw the primary streets, which is used as the basis for the secondary street network.

Most of the city modeling systems expect multiple input maps. These are either created from user input, based on real data or also procedurally generated. The commonly used ones are described below.

2.1.1 Geographical maps (Elevation maps, Land/water/vegetation maps)

Water map Water is seen as an obstacle, no buildings are allowed here. Streets are built around water in most cases, sometimes a bridge crossing the water is built.

Elevation map Terrain. Real streets are often aligned to the terrain map to minimize slope. There is also a limit of the maximum slope a street can have. In some cases this map is taken from real world data, in some it is procedurally generated based on noise (see sec. 1.2.4) (Olsen 2004; “Terragen 3” 2015).

Vegetation map Contains vegetation such as forests or parks, used e.g. as obstacles (Chen et al. 2008)

2.1.2 Sociostatistical maps (Population density, Zones, Street patterns)

Parish and Müller (2001) expect a population density input map, which is not found in other programs. Most documents have the local road pattern (see sec. 2.2.3) as inputs, which in real life is determined by the way the city was built.

2.2 Generating a street network

2.2.1 Separate street types

Most systems employ at least two street types. Parish and Müller (2001) introduce highways and streets, where highways connect population centers and streets fill the areas according to given patterns. Kelly and McCabe (2007) follow the exact same method.

Chen et al. (2008) first generate a sparse network of “major roads” and fills the spaces with minor roads. Both major and minor roads follow the same alignment rules. Highways are hand-drawn by the user in this system.

2.2.2 Global goals and local constraints

Parish and Müller (2001) use a complex L-system to produce the road network. The L-system has two external functions: `localConstraints` and `globalGoals`. `GlobalGoals` is used for the general structure of the roads. For the highways this is done by searching for the nearest population centers and navigating in that direction. Normal streets and highways are also directed according to road patterns, described in sec. 2.2.3.

`LocalConstraints` contains more local rules relevant for specific points on the map. In these specific points (described in sec. 2.2.4) the `localConstraints` function can adjust the parameters of the next iteration, or return `FAILED` if there is no fitting solution.

The approaches used in the other documents are described in more detail in sec. 2.5.

2.2.3 Road patterns, (see Parish and Müller 2001, sec. 3.2.2)

Parish and Müller (2001) mention the following three general street patterns, citing (Focas, London Research Centre, and Environment and Transport Studies 1998) (see fig. 2.1):

- Rectangular raster – aligned to one angle and perpendicular to that. This is common in planned cities that are built from the ground up in modern times.
- Radial / concentric – streets go around a center point and perpendicular streets go straight to the center.
- Branching / random – smaller streets branch from larger ones. Common in old cities that have naturally grown. Parish and Müller (2001) interpret this as no restrictions / random layout.

Most other documents use either the same, or very similar types of road patterns.

2.2.4 Constraints and solving conflicts

Streets are grown in parallel and greedily from starting points, branching off randomly.

When a new street collides with an existing street, they are connected, preferably at an already existing intersection if one is near enough. The approach by Kelly and McCabe (2007) instead has a set area (“snap radius”) around the growing street that is searched for available connections.

For other obstacles such as water or mountains, Parish and Müller (2001) propose the following solution:

If the obstacle is shorter than a preset length, it is ignored. The resulting street is marked as special according to the obstacle, e.g. as a bridge or as a tunnel.




Pattern name	Pattern	Example
Basic	No superimposed pattern.	
New York	Rectangular Raster	
Paris	Radial to center	
San Francisco	Elevation min or max	see figure 6

Figure 2.1: An overview of the street pattern used in the CityEngine system with a short description and an example (Parish and Müller 2001, table 1)

Otherwise, the near space is searched up to a maximum rotation for alternative positions that are valid. This allows the roads to work around boundaries just like they would in reality.

If the maximum angle is reached, the road segment is truncated and simply stops at the maximum valid distance.

2.2.5 Approach using tensor fields from (Chen et al. 2008)

This approach is very different from the usual methods. The conflict problem from sec. 2.2.4 does not happen here, because the whole road network is created instantly.

The input tensor field (see sec. 1.2.3) is converted into a road network by tracing hyperstreamlines. These hyperstreamlines are aligned to the eigenvector field of the tensors.

Intersections between real roads tend to have near right angles, because that creates the most efficient street navigation. In tensor fields, the major and minor eigenvectors are perpendicular to each other, so the resulting street layout created from them using the methods described by Chen et al. (2008) is similar to that of real road networks.

2.3 Splitting areas into building blocks

When the network graph is complete, the resulting areas need to be divided into blocks.

First, the streets are expanded to have a width. The resulting blocks between neighboring streets are converted into a polygon. For simplification this polygon must be convex in the approach by Parish and Müller (2001). Then the block is recursively divided along the longest approximately parallel edges until a specified maximum target size is reached. Every resulting area that is too small, or does not have direct access to a street, is discarded. The remaining blocks are interpreted as the base area for the building generation. See fig. 2.2 for an example.

2.4 Computation time

Here are some data points for execution time of the algorithms presented in some papers.

- (Parish and Müller 2001): 10 seconds for the street graph, 10 minutes for building blocks and architecture
- (Chen et al. 2008): 5 minutes for complete generation

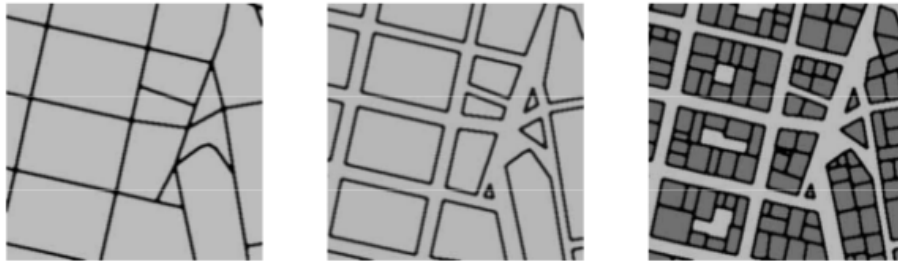


Figure 2.2: The lot division process (Parish and Müller 2001, fig. 10)

- (Kelly and McCabe 2007): real time
- (Lechner, Watson, and Wilensky 2003): real time
- (Weber et al. 2009): 5 minutes for 25 years of simulation

2.5 Street network generation approach overview

2.5.1 L-system based approach by Parish and Müller (2001)

Input

- Elevation map and Land/water map – used as obstacles
- Population density map – used for highways and street density
- Street patterns map – specifies local type of street patterns
- Zone map (residential, commercial or mixed) – used for architecture choices
- maximum house height map – used for architecture

Algorithm

Uses an extended parametric L-system with constraint functions.

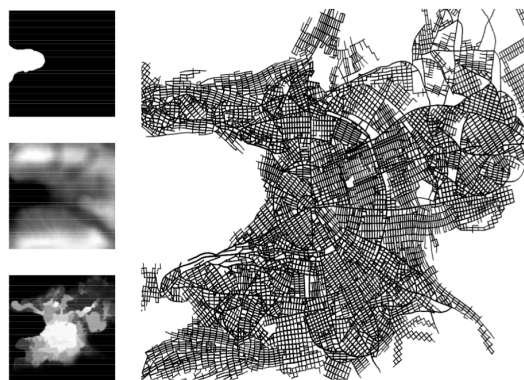


Figure 2.3: Left: Input maps (water, elevation, population density). Right: Sample output from (Parish and Müller 2001, fig. 2)

2.5.2 Approach using agents by Lechner, Watson, and Wilensky (2003)

Input

- Elevation map
- Initial seed position
- Optionally local modifications to the rules

Algorithm

Begins from a single seed. Uses multiple simultaneous agents interacting with their local environment according to a set of rules. The environment consists of rectangular patches. Only generates tertiary roads, no highways.

There are two types of agents. Extender agents search the land for areas that are not reachable from existing streets and create streets according to the elevation maps. Connector agents walk along the road network, choose random near destinations and comparing the current path finding distance to the optimal distance. If there is significant improvement, a new road segment is added.

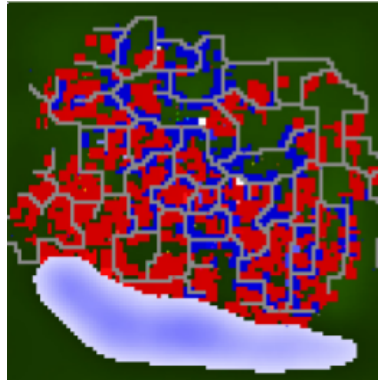


Figure 2.4: Sample output from (Lechner, Watson, and Wilensky 2003, fig. 4)

2.5.3 Approach from Citygen by Kelly and McCabe (2007)

Input

- Corner points of the primary road network

Algorithm

First the primary road network is created according to the general layouts described in sec. 2.2.3. For every resulting enclosed region the secondary road network is created independently and in parallel.

Primary Roads are created with a set start and target point. The road is built using an algorithm that walks in the direction of the destination, with a maximum angle of deviation, sampling the possible next points (see fig. 2.5). The next point is chosen so the elevation difference along the completed road is as evenly distributed as possible.

Secondary roads are generated starting from the middle of the longest sides of the primary road network. The roads grow in parallel, splitting off randomly with some specified angle plus deviation. When encountering existing roads at a maximum distance, the roads are connected, whereby existing intersections are preferred.

2.5.4 Approach using tensor fields by Chen et al. (2008)

Input

- Binary Water map – interpreted as obstacles
- Binary Park and forest map – interpreted as obstacles
- Elevation map
- Population density map

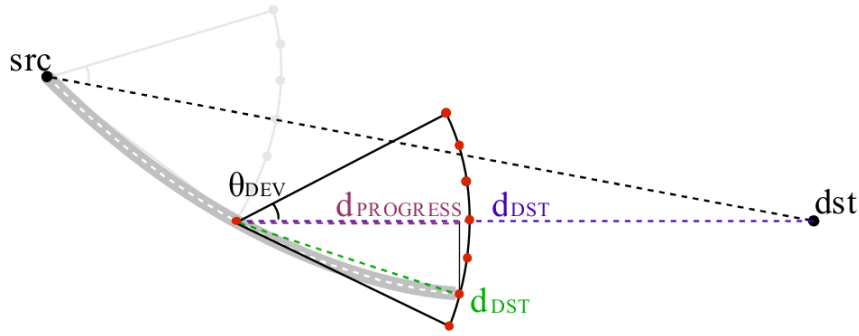


Figure 2.5: Road interval sampling (Kelly and McCabe 2007, fig. 3)

Algorithm

From the set of input maps, the initial tensor field (see sec. 1.2.3) is generated. The obstacle maps are converted to boundaries, the tensors are aligned so the major eigenvectors are in parallel to the boundaries. The elevation map is used to set the major direction to minimize the height difference of the roads. Then the user interactively adds preset tensor fields that correspond to the road patterns from sec. 2.2.3. All of the overlaid basis tensor fields are then blended together.

The tensor field is then converted to a road map by tracing hyperstreamlines (see fig. 2.6).

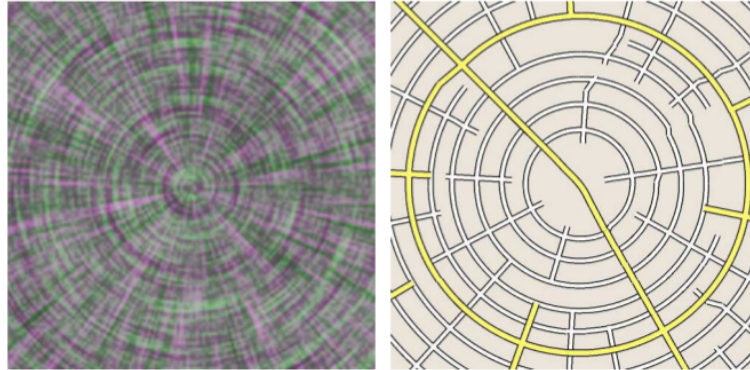


Figure 2.6: Left: radial input tensor field (Green: major, magenta: minor hyperstreamlines), Right: resulting road map (Chen et al. 2008, fig. 5)

2.5.5 Time-based approach by Weber et al. (2009)

Input

- Elevation map
- List of city centers and growth centers
- Percentage of street growth per year
- Average land price per year
- List of street patterns
- Land use type definitions and corresponding use percentages, construction setback values and building generation rules



Figure 2.7: Left: Semi-automatically generated tensor field visualized using hyperstreamlines. Right: tensor field used for creating a street network (Chen et al. 2008, fig. 1)

Algorithm

The system uses three substeps that are iteratively executed for every time step: First the street network is extended and analyzed, then the land use is planned, and finally the new construction plan is generated. The algorithms takes a lot of details into account which are not further described here.¹

Streets are built on demand according to a traffic simulation. The traffic simulation is created by simulating residents that make trips to random targets in the city. Every building and land space has a calculated value. Buildings are built on empty lots randomly and replaced when renovating significantly increases the value of the area / property.

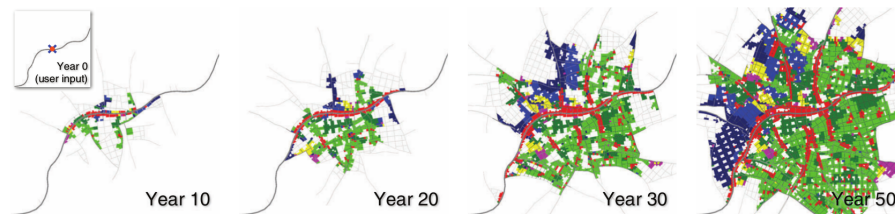


Figure 2.8: Sample output (green: residential areas, blue: industrial zones, red: commercial)(Weber et al. 2009, fig. 4)

2.5.6 Approach by Lipp et al. (2011)

This document only describes methods for interactive modification of street layouts and is not further explained here.

2.6 Architecture generation approach overview

2.6.1 Approach by Parish and Müller (2001)

A parametric, stochastic L-system is used to model the architecture. The buildings are separated into residential buildings, commercial buildings and skyscrapers, determined by the input zone map. The input also contains a building height map that is used to limit the vertical growth of buildings.

¹These slides contain further information: http://www.train-fever.com/data/xian_slides_train_fever.pdf

The rules for the L-system can be either transformation rules (scale or move), branching rules or termination rules. Generation starts with the ground plan and continues until every node only matches a termination rule.

Then, the output of the L-system is transformed into three dimensional geometry using a set of geometric models and textured. The textures are procedurally generated from layered grids describing windows, door positions, and wall materials.

2.6.2 Approach by Wonka et al. (2003)

This paper uses a special type of design grammar called split grammar. In addition to the shape grammar this uses a second grammar, called the control grammar, to ensure symmetry.

Whenever a rule is applied, the resulting elements get a set of common attributes, some copied from the parent and some added by the control grammar, which decides the attributes using the spatial information, like the floor in the building that contains the element. This results in the expected symmetry of building facades.

2.6.3 Approach by Müller et al. (2006)

This paper uses an extension of the split grammar by Wonka et al. (2003), adding rules for transformation (rotations and scaling) and volumetric mass models. It also describes concrete grammars for specific applications (office buildings and single family homes).

2.6.4 Other Approaches

- Coelho et al. (2007) use geospatial L-systems.
- Weber et al. (2009) use preconfigured buildings for specific land uses created from parametric shape grammars described by Müller et al. (2006), replacing buildings as time progresses in the simulation.
- Martin et al. (2010) use an evolutionary approach that iteratively creates buildings, combining them from breeding pools of the last generation, mutating the structure randomly.

References

- Chen, Guoning, Gregory Esch, Peter Wonka, Pascal Müller, and Eugene Zhang. 2008. “Interactive Procedural Street Modeling.” In *ACM SIGGRAPH 2008 Papers*, 103:1–103:10. SIGGRAPH ’08. New York, NY, USA: ACM. doi:[10.1145/1399504.1360702](https://doi.org/10.1145/1399504.1360702).
- Coelho, A., M. Bessa, A. Augusto Sousa, and F. Nunes Ferreira. 2007. “Expeditious Modelling of Virtual Urban Environments with Geospatial L-Systems.” *Computer Graphics Forum* 26 (4): 769–82. doi:[10.1111/j.1467-8659.2007.01032.x](https://doi.org/10.1111/j.1467-8659.2007.01032.x).
- “Esri CityEngine | 3D Modeling Software for Urban Environments.” 2015. Accessed November 8. <http://www.esri.com/software/cityengine>.
- “File:Perlin.png.” 2012. *Wikipedia, the Free Encyclopedia*. <https://en.wikipedia.org/w/index.php?title=File:Perlin.png&oldid=483227863>.
- Focas, Caralampo, London Research Centre, and Environment and Transport Studies. 1998. *The Four World Cities Transport Study*. London: Stationery Office.
- Hanan, James Scott. 1992. “Parametric L-Systems and Their Application to the Modelling and Visualization of Plants.” The University of Regina (Canada).
- Kelly, George, and Hugh McCabe. 2007. “Citygen: An Interactive System for Procedural City Generation.” In *Fifth International Conference on Game Design and Technology*, 8–16. http://procedural.googlecode.com/svn-history/r121/trunk/articles_cities/citygen_gdtw07.pdf.
- Kelly, George, and Hugh McCabe. 2006. “A Survey of Procedural Techniques for City Generation.” *ITB Journal*.
- Lechner, Thomas, Ben Watson, and Uri Wilensky. 2003. “Procedural City Modeling.” In *1st Midwestern Graphics Conference*.
- Lipp, M., D. Scherzer, P. Wonka, and M. Wimmer. 2011. “Interactive Modeling of City Layouts Using Layers of Procedural Content.” *Computer Graphics Forum* 30 (2): 345–54. doi:[10.1111/j.1467-8659.2011.01865.x](https://doi.org/10.1111/j.1467-8659.2011.01865.x).
- Martin, Andrew, Andrew Lim, Simon Colton, and Cameron Browne. 2010. “Evolving 3D Buildings for the Prototype Video Game Subversion.” In *Applications of Evolutionary Computation*, edited by Cecilia Di Chio, Stefano Cagnoni, Carlos Cotta, Marc Ebner, Anikó Ekárt, Anna I. Esparcia-Alcazar, Chi-Keong Goh, et al., 111–20. Lecture Notes in Computer Science 6024. Springer Berlin Heidelberg. http://link.springer.com/chapter/10.1007/978-3-642-12239-2_12.
- Müller, Pascal, Peter Wonka, Simon Haegler, Andreas Ulmer, and Luc Van Gool. 2006. “Procedural Modeling of Buildings.” In *ACM SIGGRAPH 2006 Papers*, 614–23. SIGGRAPH ’06. New York, NY, USA: ACM. doi:[10.1145/1179352.1141931](https://doi.org/10.1145/1179352.1141931).
- Olsen, Jacob. 2004. *Realtime Procedural Terrain Generation - Realtime Synthesis of Eroded Fractal Terrain for Use in Computer Games*.

- Parish, Yoav I. H., and Pascal Müller. 2001. "Procedural Modeling of Cities." In *Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques*, 301–8. SIGGRAPH '01. New York, NY, USA: ACM. doi:[10.1145/383259.383292](https://doi.org/10.1145/383259.383292).
- Perlin, Ken. 2002. "Improving Noise." In *Proceedings of the 29th Annual Conference on Computer Graphics and Interactive Techniques*, 681–82. SIGGRAPH '02. New York, NY, USA: ACM. doi:[10.1145/566570.566636](https://doi.org/10.1145/566570.566636).
- Peter Wonka Research, dir. 2008. *2008 SG Interactive Procedural Street Modeling - Youtube*. <https://www.youtube.com/watch?v=2PcpURiyJFw>.
- "Procedural City, Part 1: Introduction. Twenty Sided." 2015. Accessed November 5. <http://www.shamusyoung.com/twentysidedtale/?p=2940>.
- Prusinkiewicz, Przemyslaw, and Aristid Lindenmayer. 1990. "Graphical Modeling Using L-Systems." In *The Algorithmic Beauty of Plants*, 1–50. The Virtual Laboratory. Springer New York. http://link.springer.com/chapter/10.1007/978-1-4613-8476-2_1.
- Sean Barrett. 2008. "L-Systems Considered Harmful." http://nothings.org/gamedev/l_systems.html.
- "Terragen 3." 2015. Accessed November 5. <http://planetside.co.uk/products/terragen3>.
- Vanegas, C. A., D. G. Aliaga, P. Wonka, P. Müller, P. Waddell, and B. Watson. 2010. "Modelling the Appearance and Behaviour of Urban Spaces." *Computer Graphics Forum* 29 (1): 25–42. doi:[10.1111/j.1467-8659.2009.01535.x](https://doi.org/10.1111/j.1467-8659.2009.01535.x).
- Weber, Basil, Pascal Müller, Peter Wonka, and Markus Gross. 2009. "Interactive Geometric Simulation of 4D Cities." *Computer Graphics Forum* 28 (2): 481–92. doi:[10.1111/j.1467-8659.2009.01387.x](https://doi.org/10.1111/j.1467-8659.2009.01387.x).
- Wonka, Peter, Michael Wimmer, François Sillion, and William Ribarsky. 2003. "Instant Architecture." In *ACM SIGGRAPH 2003 Papers*, 669–77. SIGGRAPH '03. New York, NY, USA: ACM. doi:[10.1145/1201775.882324](https://doi.org/10.1145/1201775.882324).

Erklärung

Ich versichere, dass ich die Arbeit selbstständig verfasst habe und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe, die wörtlich oder inhaltlich übernommenen Stellen als solche kenntlich gemacht und die Satzung des KIT zur Sicherung guter wissenschaftlicher Praxis in der jeweils gültigen Fassung beachtet habe. Die Arbeit wurde in gleicher oder ähnlicher Form noch keiner anderen Prüfungsbehörde vorgelegt und von dieser als Teil einer Prüfungsleistung angenommen.

Karlsruhe, den November 23, 2015

(Robin)