A Survey of Procedural Methods for Terrain Modelling[*]

Ruben M. Smelik, Klaas Jan de Kraker and Saskia A. Groenewegen
TNO Defence, Security and Safety
The Hague, The Netherlands

Tim Tutenel and Rafael Bidarra
Delft University of Technology
Delft, The Netherlands

Abstract

Procedural methods are a promising but underused alternative to manual content creation. Commonly heard drawbacks are the randomness of and the lack of control over the output and the absence of integrated solutions, although more recent publications increasingly address these issues. This paper surveys procedural methods applied to terrain modelling, evaluating realism of their output, performance and control users can exert over the procedure.

1 Introduction

In the last two decades, 3D virtual worlds have advanced from primitive to, at least visually, very advanced and complex. However, the modelling process, tools, and techniques used to create these worlds have not advanced that much: they are still laborious and repetitious in use and require specialized 3D modelling skills.

Procedural modelling has been an active research topic for at least thirty years. The philosophy of procedural modelling is, instead of designing content by hand, to design a procedure that creates content automatically. This approach has been successfully applied to generate, for example, textures, geometric models, animations and even sound clips. A major topic within procedural modelling is the automatic generation of terrain models, which started with natural phenomena such as terrain elevation and growth of plants in the 1980s and 1990s and extended its focus to urban environments at the start of the new millennium.

Despite the encouraging results, procedural modelling is not often applied in mainstream terrain modelling. Several factors limit this transition from manual to automated modelling. For one, both research papers and commercial tools typically focus on one aspect of terrain modelling (for instance, generating interesting elevation profiles) and address other aspects to a limited extent or not at all. The integration and adjustment of existing procedural methods in such a way that they can automatically generate a complete and consistent terrain model remains to date largely unsolved. Another known issue of procedural methods is the lack of control they offer to users. The inherent randomness of the resulting content often forces users to model by trial and error. More recent publications sometimes address this issue with specific solutions, but overall it has not yet received sufficient attention.

This paper presents a short survey of procedural methods applied to terrain modelling. We discuss important properties of the methods, such as the realism of the output, the performance of the algorithm and the facilities it provides users to influence and control the generation process. The purpose of this paper is twofold: to give readers interested in the field of procedural terrain modelling an overview of the research to date, and to assess the extent to which the identified issues of procedural methods are being addressed and what remains to be done.

In (Smelik et al., 2008, 2009), we have identified several requirements a procedural modelling framework should fulfil to be a qualitatively acceptable and more productive alternative to the

[*]This research has been supported by the GATE project, funded by the Netherlands Organization for Scientific Research (NWO) and the Netherlands ICT Research and Innovation Authority (ICT Regie).

current modelling workflow. We described the conceptual design of such a framework. It integrates procedural methods and manages dependencies between terrain features in order to generate a complete, consistent terrain model, which matches a sketch of the rough layout of the terrain made by the user. It involved distinguishing several layers in a terrain model, each containing natural features (earth, water, vegetation layers) and man-made (road and urban layers). We present the design, implementation and results of two of these layers. In this paper, the distinction of terrain layers is also practical to structure the work we surveyed.

The next sections discuss procedural methods for elevation data, water bodies, vegetation, road networks and urban environments, followed by a conclusion on the state of the art.

2 Height-maps

Height-maps, i.e. two-dimensional grids of elevation values, are often used as the basis of a terrain model. There are many procedural algorithms for creating height-maps.

Among the oldest algorithms are the subdivision based methods. A coarse height-map is iteratively subdived, each iteration using controlled randomness to add detail. Miller (1986) describes several variants of the well known mid-point displacement method, in which a new point's elevation is set to the average of its corners in a triangle or diamond shape plus a random offset. The offset's range decreases each iteration according to a parameter that controls the roughness of the resulting height-map.

Height-map generation is nowadays often based on fractal noise generators (Fournier et al., 1982; Voss, 1985), such as Perlin noise (Perlin (1985)), which generates noise by sampling and interpolating points in a grid of random vectors. Rescaling and adding several levels of noise to each point in the height-map results in natural, mountainous-like structures. For a recommended textbook on fractal noise and height-map generation, see Ebert et al. (2003).

These height-maps can be transformed further based on common imaging filters (e.g. smoothing) or on simulations of physical phenomena, for instance erosion. Thermal erosion diminishes sharp changes in elevation, by iteratively distributing material from higher to lower points, until the talus angle, i.e. maximum angle of stability for a material such as rock or sand, is reached. Erosion caused by rainfall (fluvial erosion) can be simulated using, for example, cellular automata, where the amount of water and dissolved material that flows out to other cells is calculated based on the local slope of the terrain surface. Musgrave treats both types of erosion (Musgrave et al., 1989; Musgrave, 1993) and Olsen (2004) discusses several speed optimizations with reduced but acceptable quality. Beneš and Forsbach (2001) introduce a terrain structure suited for more realistic erosion algorithms. Their terrain model consists of stacked horizontal slices of material, each having an elevation value and material properties, e.g. density. It is a trade-off between the limited but efficient height-map structure and a full voxel terrain. The model also allows for air layers, thereby it supports cave structures.

While these erosion algorithms add much to the believability of mountainous terrain, they are also notoriously slow, having to run for hundreds to thousands of iterations. Recent research has focussed on interactive erosion algorithms, often by porting algorithms to the GPU. Promising examples include (Anh et al., 2007) and (Šťava et al., 2008).

The basic noise-based height-map generation delivers results that are fairly random; users control the outcome only on a global level, often using unintuitive parameters. Several researchers have addressed this issue. Stachniak and Stuerzlinger (2005) propose a method that integrates constraints (expressed as mask images) into the terrain generation process. It employs a search algorithm that finds an acceptable set of deformation operations to apply to a random terrain in order to obtain a terrain that conforms to these constraints. Schneider et al. (2006) introduce an editing environment in which the user edits the terrain by interactively modifying the base functions of the noise generator (by replacing the Perlin noise grid with a set of user-drawn gray-scale images). Zhou et al. (2007) describe a technique that generates terrain based on an example input height-map and a user line drawing that defines the occurrence of large-scale curved line features, such as a mountain ridge. Features are extracted from the example height-map and matched to the sketched curves and seamed in the resulting height-map. De Carpentier and

Bidarra (2009) introduce procedural brushes: users paint height-mapped terrain directly in 3D by applying simple terrain raising brushes but also GPU-based brushes that generate several types of noise in real-time (see Fig. 1a). Saunders (2006) proposes a very different method, which synthesizes a height-map based on Digital Elevation Models (DEM) of real world terrain. A user of his system Terrainosaurus draws a 2D map of polygonal regions, each of which is marked to have a certain elevation profile. For realism, the straight boundaries of the region are perturbed and then rasterized in a grid. A height-map is instantiated using a genetic algorithm that selects DEM data that matches the requested elevation profile. Kamal and Uddin (2007) present a constrained mid-point displacement algorithm that creates a single mountain according to such properties as elevation and base spread. Belhadj (2007) introduces a more general system where a set of known elevation values constrain the mid-point displacement process. Possible applications are interpolation of coarse or incomplete DEM's or user line sketches.

An inherit limitation of height-maps is that they do not support rock overhangs and caves. Gamito and Musgrave (2001) propose a terrain warping system that results in regular, artificial overhangs. A recent method (Peytavie et al., 2009) provides a more elaborate structure with different material layers that supports rocks, arches, overhangs and caves. Their resulting terrain models are visually very plausible and natural.

As an illustration of the state of the art in tool support, see Fig. 1d) for a render of a height-map generated by L3DT (Torpy, 2009), one of the commercial tools for creating height-maps.

3 Rivers, Oceans and Lakes

For generating rivers, several authors have proposed algorithms that run either during or after height-map generation. Kelley et al. (1988) take a river network as the basis of a height-map. They start with a single straight river and recursively subdivide it, resulting in a stream network. This forms a skeleton for the height-map, which is then filled using a scattered data interpolation function. The climate type and the soil material influence the shape of the stream network.

Prusinkiewicz and Hammel (1993) combine the generation of a curved river with a height-map subdivision scheme. Of the river's starting triangle, one edge is marked as the entry and one as the exit of the river. In a subdivision step, the triangle is divided into smaller triangles, and the river's course from entry to exit can now take several alternative forms. The elevation of the triangles containing the river is set to be the sum of the negative displacements of the river on all recursion levels (resulting in a river bed), other triangles are processed using standard mid-point displacement. After eight or more recursions, the resulting river course looks reasonably natural. A major downside of the approach is that the river is placed at a constant and low elevation level, and thus carves unnaturally deep through a mountainous terrain.

A more advanced approach described by Belhadj and Audibert (2005) creates a height-map with mountain ridges and river networks. Starting with an empty map, they place pairs of ridge particles at a particular high elevation and move them in opposite directions in several iterations. A Gaussian curve is drawn on the height-map along the particle positions of each iteration. Next, they place river particles along the top of the mountain ridge and let them flow downwards according to simple physics (comparable to hydraulic erosion). The remaining points in between ridges and rivers are filled with an inverse midpoint displacement technique. For this specific type of terrain, i.e. steep mountain ridges with valleys featuring a dense river network, the method is fast and effective.

Except for rivers, procedural water bodies, such as oceans and lakes and their connections, stream networks, deltas and waterfalls, have received too little attention to date. The forming of lakes is typically not considered at all. Oceans are commonly generated setting a fixed water level (e.g. 0 m) or by starting a flooding algorithm from points of low elevation. Teoh (2008) also states that the research in this area is incomplete: several river and coastal features have not been addressed. He proposes fast and simple algorithms for river meandering, deltas and beach forming, which require further work to increase their realism.

4 Plant Models and Vegetation Distribution

Regarding vegetation, authors developed several procedures for generating tree and plant models and methods for automatic placement of vegetation on a terrain model. The former can be used to quickly obtain a set of similar but varying plant models of the same species; the latter saves terrain modellers the laborious task of manually placing all these individual vegetation models in a large forest.

Procedural plant models grow, starting from the root, adding increasingly smaller branches and ending with the leaves. They are based on grammar rewriting. Prusinkiewicz and Lindenmayer (1990) discuss the Lindenmayer-system, or *L-system*, an often used rewriting system. They explain how production rules can be applied in 3D, and present many examples of generated trees together with their grammar.

Lintermann and Deussen (1999) propose a more intuitive system to procedurally model plants, by placing plant components (e.g. a leaf) in a graph. Connected components can be structured in subgraphs (e.g. a twig). The system traverses this graph, generating and placing instances of the components in an intermediate graph that is used for geometry generation. Fig. 1e) shows a tree created with their commercial plant modelling software XFrog.

Deussen et al. (1998) describe an ecosystem simulation model to populate an area with vegetation. The input of this simulation model is the height-map and a water map, several ecological properties of plant species, such as rate of growth, and, optionally, an initial distribution of plants. Based on this and taking into account rules for competition for soil, sunlight and water, a distribution of plants inside an area is iteratively determined (see Fig. 1b), running for several minutes.

Another procedure for vegetation placement by Hammes (2001) is based on ecosystems. He uses elevation data, relative elevation, slope, slope direction and multi fractal noise to select one of the defined ecosystems. Ground vegetation textures are generated at run-time, depending on the level of detail and the ecosystem. The ecosystem also determines the number of plants per species, which are then placed randomly.

Procedural modelling of vegetation delivers believable results and is already applied quite often in modern games, for instance using the commercial package SpeedTree.

5 Road Networks

The generation of road networks for cities can be done using a variety of methods, of which we treat the pattern-based approaches, L-systems, agent simulations and tensor fields. The simplest technique is to generate a dense square grid (as in Greuter et al. (2003)). Displacement noise can be added to grid points to create a less repetitive network, but still the realism of this technique is limited.

A more elaborate method to create roads is via the use of templates, as proposed by Sun et al. (2002). They observe several frequent patterns in real road networks and aim to reconstruct them. For each pattern, there is a corresponding template: a population-based template (implemented as the Voronoi diagram of population centres), a raster and radial template, or a mixed template. The main arteries of the road map are the highways, which are generated first using these pattern templates. Simple rules are applied to check their validity. When encountering impassable areas (e.g. oceans), they are discarded or diverted. Next, the main roads are often curved to avoid large elevation gradients. The regions they encompass are filled in with a raster of streets.

Parish and Müller (2001) use an extended L-system to grow their road network. The L-system is goal-driven and the goals are the population density (the roads try to connect population centres) and specific road patterns. Examples of such patterns are the raster or the radial pattern. Their L-system is extended with rules that have a tendency to connect new proposed roads to existing intersections and rules that check road validity with respect to impassable terrain and elevation constraints. Streets are also inserted into the remaining areas as simple grids.

Kelly and McCabe (2007) introduce the interactive city editor CityGen, in which a user defines the main roads by placing nodes in the 3D terrain. Regions enclosed by these roads can be filled with one of three patterns: Manhattan-style grids, industrial grown roads with dead-ends and organic roads as in e.g. North-American suburbs.

Glass et al. (2006) describe several experiments of replicating the road structure found in South African informal settlements using a combination of a Voronoi diagram for the major roads with L-systems or regular subdivision with and without displacement noise for the minor roads. They were reasonably successful in recreating the observed patterns.

Different from the grammar- and pattern-based approaches discussed above, Lechner et al. (2003) take an agent-based approach, in which they divide the city into areas including not only residential, commercial and industrial areas, but also special areas like government buildings, squares, and institutions. They place two agents, named the *extender* and the *connector*, at a seed position in the terrain map. The extender searches for unconnected areas in the city. When it finds such an area that is located not too far from the existing road network, it finds the most suitable path to connect the area to the network. The connector agent starts from a certain location on the existing network and randomly chooses another spot on the network, within a certain radius. It checks the length of the shortest path to this spot. If the travel time is considered too long, a direct road connection is added to the network. In Lechner et al. (2006), the authors extend this method with, among other things, agents that are responsible for constructing main roads for fast connections through the city, and agents that develop small streets. This method gives plausible results, but a disadvantage is its very long running time.

Chen et al. (2008) propose interactive modelling of road networks by the use of tensor fields. They define how to create common road patterns (grid, radial, along a boundary) using tensor fields. A road network is generated from a tensor field, by tracing the streamlines from seed points in the major eigenvector direction until a stopping condition is met. Next, along this traced curve new seed points are placed for tracing streamlines in the perpendicular (minor eigenvector) direction. Users can place new basis tensor fields, such as a radial pattern, smooth the field, or use a brush to locally constrain the field in a specific direction. Noise can be applied to make the road network less regular and thereby more plausible.

In the discussed methods, the influence of the underlying terrain map and elevation profile is to varying degrees taken into account. Most methods take only basic measures to avoid too steep ascending roads and roads through water bodies. Kelly and McCabe (2007) plan the precise path of their main roads between the user set nodes to have an even change in elevation as much as possible. Still, for rough terrain this measure will not be adequate and the terrain needs to be modified to accommodate for the road. Bruneton and Neyret (2008) propose a simple and effective method for blending road profiles into the height-map using shaders.

6 Urban Environments

Kelly and McCabe (2006) give an elaborate overview of several approaches for generating urban environments. Watson et al. (2008) give a practical overview of the state of the art.

The common approach for procedurally generating cities is to start from a dense road network and identify the polygonal regions enclosed by streets. Subdivision of these regions results in lots, for which different subdivision methods exist, see e.g. Parish and Müller (2001) or Kelly and McCabe (2007). To populate these lots with buildings, either the lot shape is used directly as the footprint of a building, or a building footprint is fitted on the lot. By simply extruding the footprint to a random height, one can generate a city of skyscrapers and office buildings. To obtain more interesting building shapes, several approaches have been devised.

Greuter et al. (2003) generate office buildings by combining several primitive shapes into a floor plan and extruding these to different heights. Parish and Müller (2001) start with a rectangular floor plan and apply an L-system to refine the building. Both approaches are most useful for relatively simple office building models.

Wonka et al. (2003) introduce the concept of the *split grammar*, a formal context-free grammar designed to produce building models. The split grammar resembles an L-system, but is based on shapes as primitive elements rather than symbols. In their system, a specific building style can be acquired by setting an attribute of the start symbol, which is propagated during the rewrite. Within one building model, the style can differ per floor (e.g. an apartment building with shops on the ground floor). Their approach focuses mostly on generating coherent and believable facades

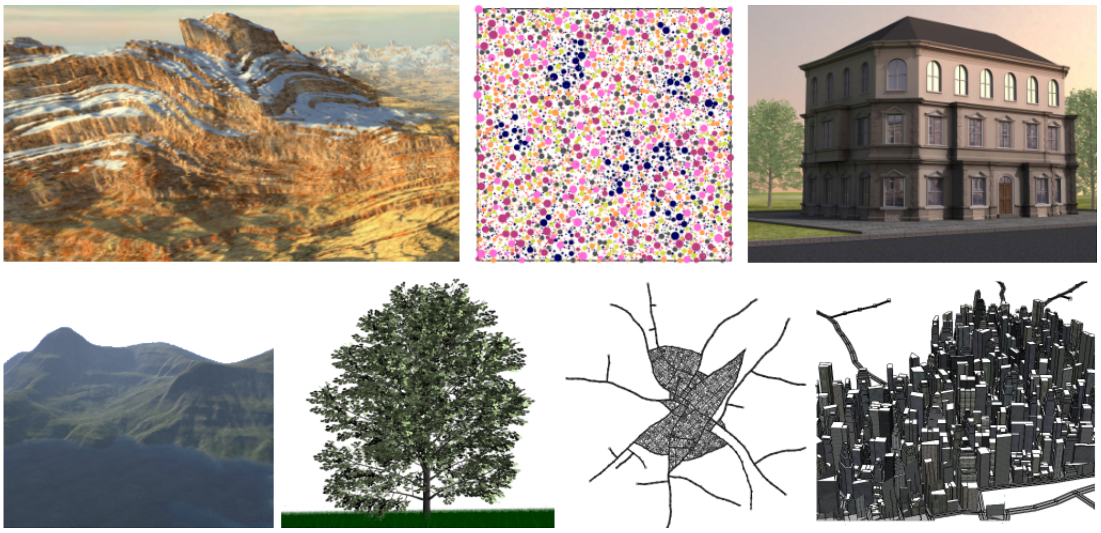

Figure 1: a) Terrain interactively created using procedural brushes (De Carpentier and Bidarra, 2009). b) Plant distribution simulation (Deussen et al., 1998). c) Complex building facade (Finkenzeller and Bender, 2008). d) A 3D render of a height-map generated using L3DT (Torpy, 2009). e) A plant model generated using XFrog (Greenworks, 2009). f) and g) A road network and corresponding city generated using CityEngine (Procedural, inc., 2009).

for relatively simple shaped buildings. Larive and Gaildrat (2006) use a similar kind of grammar, called a *wall grammar*. With this grammar they are able to generate building walls with additional geometric details, such as balconies.

Müller et al. (2006) apply another type of grammar, named *shape grammar*. The main property of a shape grammar is that it uses context-sensitive rules, whereas a split grammar uses context-free rules, which in this case allows the possibility to model roofs and rotated shapes. They start with a union of several volumetric shapes which defines the boundary of the building. This shape is then divided into floors and the resulting facades are subdivided into walls, windows, and doors by means of a grammar system. In a final step, the roof is constructed on top of the building. Fig. 1f shows a road network and Fig. 1g the corresponding city generated by their commercial product, CityEngine (Procedural, inc., 2009). Besides the well known business buildings, the grammar can also model residential buildings, e.g. suburban homes or ancient Roman villas.

Although the shape grammars in Müller et al. (2006) can generate visually convincing building models, Finkenzeller and Bender (2008) note that semantic information regarding the role of each shape within the complete building is missing. They propose to capture this semantic information in a typed graph. Their workflow consists of three steps. Starting with a rough building outline, a building style graph can be applied to this model. This results in an intermediate semantic graph representation of the building, which can be modified or regenerated with a different style. In the last step, geometry is created based on the intermediate model, and textures are applied, resulting in a complete 3D building. Finkenzeller (2008) describes in more detail the generation of facades and roofs in their system (see Fig. 1c)).

Yong et al. (2004) describe a method to create vernacular-style southeast Chinese houses using an extended shape grammar. The grammar is hierarchical and starts at the city level (whereas in other methods a shape grammar is applied to an individual building footprint). The grammar then produces streets, housing blocks, roads, and in further productions houses with components such as gates, windows, walls, and roofs. Through a number of control rules (defining, for instance, component ratio constraints) the validity of the buildings can be asserted. By applying this grammar system, a typical ancient southeast Chinese town can be generated with plausible results, since the building style of these towns is very rigidly structured.

Müller et al. (2007) used a very different approach for constructing building facades. Their method takes a single image of a facade of a real building as input and is able to reconstruct a detailed 3D facade model, using a combination of imaging and shape grammar generation.

Although the above methods give fast and visually pleasing results, the cities they generate often lack a realistic structure. New research incorporates existing urban land use theories and models in the generation process. Groenewegen et al. (2009) present a method that generates a distribution of different types of districts according to land use models of cities in Western-Europe and North-America. It takes into account a large number of relevant factors, including the historic core of the city and the attraction certain types of terrain (hillsides, oceans, rivers) have for e.g. industrial or high-class residential districts. Weber et al. (2009) use comparable (albeit slightly simplified) models for a simulation of expanding cities over time. Their system is fast (about 1 sec. per simulated year) and interactive, meaning that the user can guide the simulation by changing roads or painting land use values on the terrain.

7 Conclusions

Procedural methods for terrain modelling are becoming increasingly attractive for both academia and industry, as a promising alternative to the expensive manual creation of content for virtual worlds. We have classified these methods into five main areas, and discussed a large variety of research approaches and results of each of them: terrain elevation, water elements, vegetation, road networks and urban environments.

From its early years, where the focus was mainly set on height-map generation, until now, with a shift towards more and more realistic urban environments, there is a considerable body of research results available. Many basic procedural methods deploy common building blocks such as noise, rewriting grammars and simple simulation systems, of which a large number of variants, often very much domain-specific, are now being proposed, in particular within the road and urban categories. Of all categories discussed, the water-related area is clearly the most underdeveloped.

Regarding upcoming research, several interesting trends have been identified. Among them, three promising directions can be summarized as follows. First, performance and interactivity of procedural methods will continue to improve, often by means of parallel programming on the GPU. Second, road networks and urban areas will certainly continue to improve in variation and level of detail, but the realism leap will likely be given by deploying more and more semantics in both the procedural generation process and the generated models (Tutenel et al., 2008). And last, the key to a widespread deployment of procedural methods by non-experts (e.g. game designers, artists, scenario designers) will be the integration of procedural methods within a framework, offering among other things, more intuitive controls, tools to generate complete terrain models and non-intrusive mechanisms to maintain the consistency among generated features (Smelik et al., 2009).

References

Anh, N. H., Sourin, A., and Aswani, P. (2007). Physically based Hydraulic Erosion Simulation on Graphics Processing Unit. In *GRAPHITE '07: Proceedings of the 5th International Conference on Computer Graphics and Interactive Techniques in Australia and Southeast Asia*, pages 257–264, New York, NY, USA. ACM.

Belhadj, F. (2007). Terrain Modeling: a Constrained Fractal Model. In Spencer, S. N., editor, *AFRIGRAPH '07: Proceedings of the 5th International Conference on Computer Graphics, Virtual Reality, Visualisation and Interaction in Africa*, pages 197–204, New York, NY, USA. ACM.

Belhadj, F. and Audibert, P. (2005). Modeling Landscapes with Ridges and Rivers: Bottom Up Approach. In *GRAPHITE '05: Proceedings of the 3rd International Conference on Computer Graphics and Interactive Techniques in Australasia and South East Asia*, pages 447–450, New York, NY, USA. ACM.

Beneš, B. and Forsbach, R. (2001). Layered Data Representation for Visual Simulation of Terrain Erosion. In *SCCG '01: Proceedings of the 17th Spring Conference on Computer Graphics*, pages 80–86, Washington, DC, USA. IEEE Computer Society.

Bruneton, E. and Neyret, F. (2008). Real-time Rendering and Editing of Vector-based Terrains. In Drettakis, G. and Scopigno, R., editors, *Eurographics 2008 Proceedings*, volume 27, pages 311–320, Hersonissos, Grèce.

Chen, G., Esch, G., Wonka, P., Müller, P., and Zhang, E. (2008). Interactive Procedural Street Modeling. In *SIGGRAPH '08: Proceedings of the* 35th *Annual Conference on Computer Graphics and Interactive Techniques*, volume 27, pages 1–10, New York, NY, USA. ACM.

De Carpentier, G. and Bidarra, R. (2009). Interactive GPU-based Procedural Heightfield Brushes. In *Proceedings of the* 4th *International Conference on the Foundation of Digital Games*, Florida, USA.

Deussen, O., Hanrahan, P., Lintermann, B., Měch, R., Pharr, M., and Prusinkiewicz, P. (1998). Realistic Modeling and Rendering of Plant Ecosystems. In *SIGGRAPH '98: Proceedings of the* 25th *Annual Conference on Computer Graphics and Interactive Techniques*, pages 275–286, New York, NY, USA. ACM.

Ebert, D. S., Worley, S., Musgrave, F. K., Peachey, D., and Perlin, K. (2003). *Texturing & Modeling, a Procedural Approach.* Elsevier, 3rd edition.

Finkenzeller, D. (2008). Detailed Building Facades. *IEEE Computer Graphics and Applications*, 28(3):58–66.

Finkenzeller, D. and Bender, J. (2008). Semantic Representation of Complex Building Structures. In *Computer Graphics and Visualization (CGV 2008) - IADIS Multi Conference on Computer Science and Information Systems*, Amsterdam, The Netherlands.

Fournier, A., Fussell, D., and Carpenter, L. (1982). Computer Rendering of Stochastic Models. *Communications of the ACM*, 25(6):371–384.

Gamito, M. and Musgrave, F. K. (2001). Procedural Landscapes with Overhangs. In 10th *Portuguese Computer Graphics Meeting*, pages 33–42.

Glass, K. R., Morkel, C., and Bangay, S. D. (2006). Duplicating Road Patterns in South African Informal Settlements Using Procedural Techniques. In Spencer, S. N., editor, *AFRIGRAPH '06: Proceedings of the* 4th *International Conference on Computer Graphics, Virtual Reality, Visualisation and Interaction in Africa*, pages 161–169, New York, NY, USA. ACM.

Greenworks (2009). XFrog. Available from http://www.xfrogdownloads.com.

Greuter, S., Parker, J., Stewart, N., and Leach, G. (2003). Real-time Procedural Generation of 'Pseudo Infinite' Cities. In *GRAPHITE '03: Proceedings of the* 1st *International Conference on Computer Graphics and Interactive Techniques in Australasia and South East Asia*, pages 87–94, New York, NY, USA. ACM.

Groenewegen, S. A., Smelik, R. M., de Kraker, K. J., and Bidarra, R. (2009). Procedural City Layout Generation Based On Urban Land Use Models. In Alliez, P. and Magnor, M., editors, *Short Paper Proceedings of Eurographics 2009*, pages 45–48, Munich, Germany. Eurographics Association.

Hammes, J. (2001). Modeling of Ecosystems as a Data Source for Real-Time Terrain Rendering. In *DEM '01: Proceedings of the First International Symposium on Digital Earth Moving*, pages 98–111, London, UK. Springer-Verlag.

Kamal, K. R. and Uddin, Y. S. (2007). Parametrically Controlled Terrain Generation. In *GRAPHITE '07: Proceedings of the* 5th *International Conference on Computer Graphics and Interactive Techniques in Australia and Southeast Asia*, pages 17–23, New York, NY, USA. ACM.

Kelley, A. D., Malin, M. C., and Nielson, G. M. (1988). Terrain Simulation Using a Model of Stream Erosion. In *SIGGRAPH '88: Proceedings of the* 15th *Annual Conference on Computer Graphics and Interactive Techniques*, pages 263–268, New York, NY, USA. ACM.

Kelly, G. and McCabe, H. (2006). A Survey of Procedural Techniques for City Generation. *Institute of Technology Blanchardstown Journal*, 14:87–130.

Kelly, G. and McCabe, H. (2007). Citygen: An Interactive System for Procedural City Generation. In *Proceedings of GDTW 2007: The Fifth Annual International Conference in Computer Game Design and Technology*, pages 8–16, Liverpool, UK.

Larive, M. and Gaildrat, V. (2006). Wall Grammar for Building Generation. In *GRAPHITE '06: Proceedings of the* 4th *International Conference on Computer Graphics and Interactive Techniques in Australasia and Southeast Asia*, pages 429–437, New York, NY, USA. ACM.

Lechner, T., Ren, P., Watson, B., Brozefski, C., and Wilenski, U. (2006). Procedural Modeling of Urban Land Use. In *SIGGRAPH '06: ACM SIGGRAPH 2006 Research posters*, page 135, New York, NY, USA. ACM.

Lechner, T., Watson, B., Wilensky, U., and Felsen, M. (2003). Procedural City Modeling. In 1st *Midwestern Graphics Conference*, St. Louis, MO, USA.

Lintermann, B. and Deussen, O. (1999). Interactive Modeling of Plants. *IEEE Computer Graphics and Applications*, 19(1):56–65.

Miller, G. S. P. (1986). The Definition and Rendering of Terrain Maps. In *SIGGRAPH '86: Proceedings of the* 13th *Annual Conference on Computer Graphics and Interactive Techniques*, volume 20, pages 39–48, New York, NY, USA. ACM.

Müller, P., Wonka, P., Haegler, S., Ulmer, A., and Gool, L. V. (2006). Procedural Modeling of Buildings. In *SIGGRAPH '06: Proceedings of the* 33rd *Annual Conference on Computer Graphics and Interactive Techniques*, pages 614–623, New York, NY, USA. ACM.

Müller, P., Zeng, G., Wonka, P., and Gool, L. V. (2007). Image-based Procedural Modeling of Facades. In *SIGGRAPH '07: Proceedings of the* 34th *Annual Conference on Computer Graphics and Interactive Techniques*, volume 26, New York, NY, USA. ACM.

Musgrave, F. K. (1993). *Methods for Realistic Landscape Imaging.* PhD thesis, Yale University, New Haven, CT, USA.

Musgrave, F. K., Kolb, C. E., and Mace, R. S. (1989). The Synthesis and Rendering of Eroded Fractal Terrains. In *SIGGRAPH '89: Proceedings of the* 16th *Annual Conference on Computer Graphics and Interactive Techniques*, pages 41–50, New York, NY, USA. ACM.

Olsen, J. (2004). Realtime Procedural Terrain Generation. Technical Report, University of Southern Denmark.

Parish, Y. I. H. and Müller, P. (2001). Procedural Modeling of Cities. In *SIGGRAPH '01: Proceedings of the* 28th *Annual Conference on Computer Graphics and Interactive Techniques*, pages 301–308, New York, NY, USA. ACM.

Perlin, K. (1985). An Image Synthesizer. In *SIGGRAPH '85: Proceedings of the* 12st *Annual Conference on Computer Graphics and Interactive Techniques*, volume 19, pages 287–296. ACM.

Peytavie, A., Galin, E., Merillou, S., and Grosjean, J. (2009). Arches: a Framework for Modeling Complex Terrains. In *Eurographics 2009 Proceedings.* Eurographics Association.

Procedural, inc. (2009). CityEngine. Available from http://www.procedural.com.

Prusinkiewicz, P. and Hammel, M. (1993). A Fractal Model of Mountains with Rivers. In *Proceeding of Graphics Interface '93*, pages 174–180.

Prusinkiewicz, P. and Lindenmayer, A. (1990). *The Algorithmic Beauty of Plants*. Springer-Verlag, New York, NY, USA.

Saunders, R. L. (2006). Terrainosaurus: Realistic Terrain Synthesis Using Genetic Algorithms. Master's thesis, Texas A&M University.

Schneider, J., Boldte, T., and Westermann, R. (2006). Real-Time Editing, Synthesis, and Rendering of Infinite Landscapes on GPUs. In *Vision, Modeling and Visualization 2006*.

Smelik, R., de Kraker, K. J., Tutenel, T., and Bidarra, R. (2009). Declarative Terrain Modeling for Military Training Games. *Submitted for publication.*

Smelik, R., Tutenel, T., de Kraker, K. J., and Bidarra, R. (2008). A Proposal for a Procedural Terrain Modelling Framework. In *Poster Proceedings of the* 14^{th} *Eurographics Symposium on Virtual Environments EGVE08*, pages 39–42, Eindhoven, The Netherlands.

Stachniak, S. and Stuerzlinger, W. (2005). An Algorithm for Automated Fractal Terrain Deformation. *Computer Graphics and Artificial Intelligence*, 1:64–76.

Sun, J., Yu, X., Baciu, G., and Green, M. (2002). Template-based Generation of Road Networks for Virtual City Modeling. In *VRST '02: Proceedings of the ACM symposium on Virtual Reality Software and Technology*, pages 33–40, New York, NY, USA.

Teoh, S. T. (2008). River and Coastal Action in Automatic Terrain Generation. In Arabnia, H. R. and Deligiannidis, L., editors, *CGVR 2008: Proceedings of the 2008 International Conference on Computer Graphics & Virtual Reality*, pages 3–9, Las Vegas, Nevada, USA. CSREA Press.

Torpy, A. (2009). L3DT. Available from http://www.bundysoft.com/L3DT/.

Tutenel, T., Bidarra, R., Smelik, R., and de Kraker, K. J. (2008). The Role of Sematics in Games and Simulations. *ACM Computers in Entertainment*, 6:1–35.

Voss, R. F. (1985). *Fundamental Algorithms for Computer Graphics*, chapter Random Fractal Forgeries, pages 805–835. Springer-Verlag, Berlin.

Šťava, O., Beneš, B., Brisbin, M., and Křivánek, J. (2008). Interactive terrain modeling using hydraulic erosion. In Gross, M. and James, D., editors, *Eurographics / SIGGRAPH Symposium on Computer Animation*, pages 201–210, Dublin, Ireland. Eurographics Association.

Watson, B., Müller, P., Veryovka, O., Fuller, A., Wonka, P., and Sexton, C. (2008). Procedural Urban Modeling in Practice. *IEEE Computer Graphics and Applications*, 28(3):18–26.

Weber, B., Müller, P., Wonka, P., and Gross, M. (2009). Interactive Geometric Simulation of 4D Cities. *Proceedings of Eurographics 2009*, 28:481–492.

Wonka, P., Wimmer, M., Sillion, F., and Ribarsky, W. (2003). Instant Architecture. In *SIGGRAPH '03: Proceedings of the* 30^{th} *Annual Conference on Computer Graphics and Interactive Techniques*, pages 669–677, New York, NY, USA. ACM.

Yong, L., Congfu, X., Zhigeng, P., and Yunhe, P. (2004). Semantic Modeling Project: Building Vernacular House of Southeast China. In *VRCAI '04: Proceedings of the 2004 ACM SIGGRAPH International Conference on Virtual Reality Continuum and its Applications in Industry*, pages 412–418, New York, NY, USA. ACM.

Zhou, H., Sun, J., Turk, G., and Rehg, J. (2007). Terrain Synthesis from Digital Elevation Models. *IEEE Transactions on Visualization and Computer Graphics*, 13(4):834–848.