A Survey on Procedural Modeling for Virtual Worlds

Ruben M. Smelik[1], Tim Tutenel[2], Rafael Bidarra[2] and Bedrich Benes[3]

[1]Modelling, Simulation & Gaming Department, TNO, The Netherlands
[2]Computer Graphics and Visualization Group, Delft University of Technology, The Netherlands
[3]Department of Computer Graphics Technology, Purdue University, USA

Abstract
Procedural modeling deals with (semi-)automatic content generation by means of a program or procedure. Among other advantages, its data compression and the potential to generate a large variety of detailed content with reduced human intervention, have made procedural modeling attractive for creating virtual environments increasingly used in movies, games, and simulations.
We survey procedural methods that are useful to generate features of virtual worlds, including terrains, vegetation, rivers, roads, buildings, and entire cities. In this survey, we focus particularly on the degree of intuitive control and of interactivity offered by each procedural method, because these properties are instrumental for their typical users: designers and artists. We identify the most promising research results that have been recently achieved, but we also realize that there is far from widespread acceptance of procedural methods among non-technical, creative professionals. We conclude by discussing some of the most important challenges of procedural modeling.

Categories and Subject Descriptors (according to ACM CCS): Computer Graphics [I3.5]: Computational Geometry and Object Modelling—Geometric algorithms, languages, and systems

1. Introduction

Procedural modeling (PM) has been an active research topic for over thirty years and it is applied to a wide variety of areas such as modeling of textures, plants, terrain, buildings, urban areas, road networks, rivers, or art creation. There is not a single definition of procedural modeling; it encompasses a wide variety of generative techniques that can (semi−)automatically produce a specific type of content based on a set of input parameters. A widely accepted generic definition is that PM provides a content by means of a procedure or a program [EWM*03]. Procedural modeling relates to many different areas, the closest one being computational simulation. Many procedural models are essentially generative representations either of processes inspired by nature, such as plant development, or of man-centered processes, such as urban simulations. PM also relates to physics-based simulations.

One of the main advantages of PM, and probably the main reason for its large attractiveness, is its *data amplification* capabilities [Smi84]. Shortly stated: a simple set of input parameters or a few generation rules of the procedural model yield a wide variety of models.

Another essential property of PM is *data compression*: a rather complex geometric model can be represented by a compact procedural model and a set of parameters, while the actual geometry is generated only when needed. This advantage of PM is becoming even more relevant because of the evolution of computer hardware, by which more functionality is being shifted to the GPU's tessellation and geometry shader units. Instead of storing the data, one can simply run instructions to generate it.

PM has the potential to drastically reduce the amount of modeling effort required to create digital content. Furthermore, because its methods are often stochastic, PM can create a variety of results from one set of input parameters; e.g., a large number of tree models, all of the same species and age, but each with a unique structure and shape.

However, despite promising a high productivity gain, a compact representation, and a seemingly endless variation in content, most current PM methods still do not offer a suitable alternative to manual modeling. The main reason is the poor controllability of most procedural models. They require users to manipulate complicated PM rules and parameters whose effects on the output can hardly be predicted. More-

Published in COMPUTER GRAPHICS *Forum*.
Available at http://doi.org/10.1111/cgf.12276.

over, procedural models are often complex and the generative process follows quite complicated paths. As a result, systems offering PM are usually deployed in a very conservative way, where the designers mostly reuse provided models, or randomly tweak their parameters until they achieve a more or less acceptable output.

The benefits of PM make it particularly attractive for creating virtual environments. Virtual worlds are important for many applications, including a wide variety of (serious) games and simulations. However, current ways of manually creating virtual worlds are too labor-intensive and costly, as the amount and level of detail required of their 3D content continually increases. There is a growing consensus that generative content creation techniques can help solve this issue.

In recent years, the field of PM has been receiving increased attention in the research community. Many new papers are being published yearly, and we felt that a comprehensive overview was missing. We therefore survey PM methods for generating both natural and man-made structures that can be found in virtual worlds. In particular, this survey has the following goals:

- introduce the main relevant PM methods,
- classify these methods according to their underlying technique,
- provide an overview of the most recent PM advancements,
- evaluate them with the most current, relevant criteria, and
- identify open issues and promising research directions.

We intentionally limit this survey to procedural methods in the context of *features of virtual worlds*. We will therefore not consider procedural generation techniques for other types of content, such as textures, sound effects, music, or very specific types of game levels (e.g., dungeons for role playing games or map generators for real-time strategy games). Although *procedural noise* generation is an important part of procedural modeling, it will not be considered in this survey. For a reader, as a starting point we suggest the seminal paper [Per85b]. The book [EWM*03] presents an overview of existing techniques, and the recent survey [LLC*10] describes the latest advances in procedural noise generation. An important recent paper [GLLD12] not included in the survey [LLC*10] discusses Gabor noise generation.

There are many conceivable perspectives from which to analyze PM techniques, from expressive power to output quality, from application area to degree of popularity, from type of user interaction to computational complexity. Consistent with the main reasons listed above for the lack of widespread application of PM methods, and in line with the most recent trends in PM research, we chose the following three criteria as the preferential and most adequate *lenses* through which to look at the methods we are about to survey:

1. intuitiveness and ease of use,
2. degree of user control, and
3. classifications (stochastic, artificial intelligence, simulations, grammars, data-driven, and computational geometry) and application areas (terrain, vegetation, roads, water, buildings, and cities).

This survey has two parts. The first part provides an overview of existing PM methods for the generation of specific virtual world features, such as terrain (Section 2), vegetation (Section 3), water bodies (Section 4), road networks (Section 5), urban layout (Section 6), buildings (Section 7) and building interiors (Section 8). For each feature, an overview will be given of its most relevant methods, however the main focus will be on recent developments related to the feature. The part concludes with a short discussion on a selection of remarkable (commercially or freely) available PM systems (Section 9). Throughout the first part, whenever possible, we will concentrate on evaluating each method or tool based mainly on the three criteria introduced above. The second part (Sections 10 and 11) analyzes and discusses the state of the art. In this discussion, we will provide a thorough classification of all methods presented in the first part, identifying which underlying PM techniques have been used for which features. This, in turn, will help us identify existing open issues, possible missing pieces, and new directions.

2. Terrain

The most common data representation of terrains is a regular height field (a 2D grid, where the vertex value represents the elevation at that location). Height fields, also called height maps, are easy to implement and process; in addition, they can be efficiently compressed and stored on GPUs. Terrain elevation can be generated procedurally, since mountainous elevation profiles resemble shapes produced by fractals [Man82], a topic first explored at the inception of PM research in the 1980's. An inherent limitation of height fields is their inability to provide overhangs and caves, so alternative techniques for storing terrains are layered data structures [BF01], voxel data, or 3D meshes. Gamito and Musgrave propose a terrain model warping system that results in regular overhangs [GM01]. More recently, Peytavie et al. developed a model with different material layers that support loose rocks, arches, overhangs and caves [PGGM09], see Figure 1.

Early height map generation algorithms were based on subdivision methods, by which a coarse height map is iteratively refined, each iteration introducing a controlled amount of randomness to generate elevation detail. The first subdivision algorithm used for terrain generation is known as the *midpoint displacement* method [FFC82, Mil86] in which every time a new point is generated, its elevation is set to the average of its neighbors in a triangle (or a diamond) shape plus a random offset. The offset's range decreases at each iteration according to a parameter that controls the roughness of the resulting height map. Terrain generated by this method is a fractional Brownian motion surface [PJS92].

A common way of controlling the algorithm consists of adjusting the number of subdivision steps and the initial offset

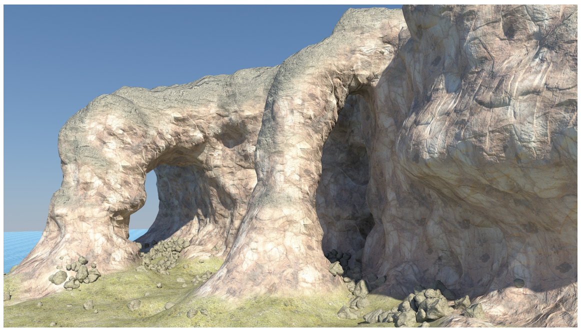

Figure 1: *A canyon with loose rocks created by the Arches framework [PGGM09]. Courtesy of A. Peytavie, E. Galin, S. Merillou, and J. Grosjean.*

range; increasing either of them increases the overall roughness of the terrain. However, these methods provide no way to influence where features, such as mountains or valleys, will occur.

There are many other stochastic methods for height map generation, mostly based on noise generators, such as Perlin noise [Per85a]. Scaling and summing several octaves of noise of increasing frequency into a height map results in natural, mountain-like structures [MKM89]. There are many flavors of this noise, also called fractional Brownian motion (fBm), suitable for generating particular terrain features, such as ridges or rolling hills. Higher dimensional noise functions allow for generation of volumetric structures such as marble, wood, or procedural clouds. If the higher dimension is interpreted as time, the structures can be animated. See Ebert et al.'s book [EWM*03] for a review of these methods.

Similar to the mid-point displacement method, control of such stochastic height map generation is limited to choosing initial generation parameters. The meaning of those parameters can be non-intuitive, and often only a limited range of possible values yield plausible results. Moreover, expressing a designer's intent in this way is almost an impossible task. The performance of fBm noise algorithms is excellent; it is well suited for parallel processing, as each grid point can be computed independently of the values of neighboring points.

Terrains provided by noise generators are usually homogeneous and do not provide local variation of features. They can be further adjusted and edited using common signal processing filters (e.g., localized smoothing adding features by manually adjusting height values). Bruneton and Neyret enhanced Digital Elevation Models (DEM [LZG10]) with continuous data of roads and rivers in [BN08].

A good way to further modify the terrains is by using physics-based algorithms that simulate erosion or weathering. Musgrave et al. introduced thermal weathering and simple hydraulic erosion [MKM89]. Thermal weathering 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 the granular material), is reached. Hydraulic erosion was simulated using 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 elevation profile. Their work has been extended in many directions including corrosion [WCMT07] and full 3D hydraulic erosion [BTHB06].

As erosion is typically implemented as a global operation, its amount of control is again limited to the initial configuration of the simulation. Furthermore, while erosion adds much to the believability of mountainous terrain, early CPU-based implementations of erosion simulations are notoriously slow and require hundreds to thousands of iterations for plausible results. However, the advent of the use of GPUs paved the way for interactive terrain modeling [ASA07, VBHv11].

While most of the above-described approaches use grid-based techniques, the method of Krištof et al. uses smoothed particle hydrodynamics (SPH) efficiently implemented on the GPU to generate full 3D erosion of large-scale terrains [KBKv09]. Although SPH is by its nature adaptive because particles are presented only where the fluid is located, the number of particles needed to achieve realistic effects is rather high.

Some of the proposed extensions provide a way to constrain the generation process in a non-interactive manner by new forms of user input. Stachniak and Stürzlinger propose a method that integrates constraints expressed as mask images [SS05]. It employs a search algorithm that finds an acceptable set of deformation operations to apply to a procedurally generated terrain in order to have it conform to those constraints. However, this method is computationally expensive.

Zhou et al. [ZSTR07] described a technique that generates a height map based on an example and a user line drawing that defines the occurrence of large-scale curved line features, such as mountain ridges. Features are extracted from the example height map, matched to the curves and stitched into the resulting height map. The input is intuitive and provides more control over the placement and shape of large terrain features. However, this method does not allow for small terrain changes and, similar to every example-based method, it is limited by the provided input set.

Doran and Parberry propose a simulation approach using agents, each creating a specific landform (e.g., coastline, beach, mountain) [DP10]. Although the method allows one to control the frequency of a specific landform, it does not offer any direct control on where they occur; additionally, its performance is not interactive.

Saunders proposes an AI-based method that synthesizes a height map based on DEMs of real-world terrain [Sau06]. A user draws a 2D map of polygonal regions, each of which is marked to have a certain elevation profile that is rasterized. A height map is instantiated using a genetic algorithm, which selects DEM data that matches the requested elevation profile in each region. However, the generated transitions at the boundaries between regions can be abrupt. Similarly to the constraint-based methods discussed above, the method offers control in the scale of large terrain features. It is fairly intuitive for designers to draw the input map, but results cannot be obtained interactively.

Kamal et al. present [KU07] a constrained mid-point displacement algorithm that creates a single mountain according to such properties as elevation and base spread, and Belhadj introduces a more general system [Bel07], where a set of known elevation values constrain the mid-point displacement process. Possible applications include interpolation of coarse or incomplete DEMs or user line sketches.

To provide users with more control over the exact appearance of mountain ranges, Gain et al. introduce a *sketch-based* height map generation method in which users sketch the silhouette and bounds of a mountain in a 3D interface, and the generator creates a matching mountain using noise propagation [GMS09]. Using diffusion equations, Hnaidi et al. allow a designer to draw 3D curves that control the shape of the generated terrain [HGA*10]. Bernhardt et al. present an efficient CPU/GPU setup to present real-time feedback to designers using this method [BMV*11]. Even more fine-grained control over the terrain shape is provided by the interactive procedural brushing system introduced by de Carpentier and Bidarra [dB09]. These GPU-based procedural brushes allow users to interactively sculpt a landscape in 3D using several types of noise.

With the purpose of facilitating even more intuitive interaction for terrain creation, Smelik et al. [STdB10b] proposed procedural sketching, another GPU-based method by which one paints a top view of the terrain by coloring a grid with ecotopes (a small area of homogeneous terrain and features), which encompass both elevation information (elevation ranges, terrain roughness) and soil material information (sand, grass, rock, etc.).

3. Vegetation

The approaches to procedural vegetation generation can be divided according to the level of detail they produce in a) individual plant organs, b) plants, and c) complete plant ecosystems. The different approaches include varying levels of interaction of the users, such as pure interactive modeling using Xfrog [LD99] or SpeedTree [Int13], plant reconstruction methods from LiDAR scans [LYO*10] or fully autonomous procedural models; only the last approach will be the focus of this survey. In the field of plant modeling, procedural models have been inspired by biological approaches. In the following text, we will use the words procedural, growth, and developmental to describe the same type of model.

One of the first approaches to plant simulation is the work of Honda [Hon71], who attempted to create branching structures using a set of simple input parameters. Lindenmayer described a linear cellular subdivision mechanism as a rewriting system of terminal and non-terminal symbols of a grammar [Lin68]. This approach, later named in his honor Lindenmayer's systems, or L-systems, is extended by Prusinkiewicz by syntactic sequence and graphics representation that allows for generation of branching structures and 3D interpretation of the string of generated modules [Pru86]. L-systems are a very powerful system, sometimes compared to a programming language, that allows for simulation of individual plants [PLH88], trees [Pru97, WP95], or entire ecosystems [Pru00]. The two most important extensions of L-systems are differential dL-systems that support continuous simulation of plant development [PHM93], and Open L-systems that extended the plant simulation to allow for integration of exogenous flow and environmental sensitivity [MP96], as can be seen in Figure 2. L-systems can generate the entire scale of plant models from cellular subdivision, individual plant organs, to entire ecosystems.

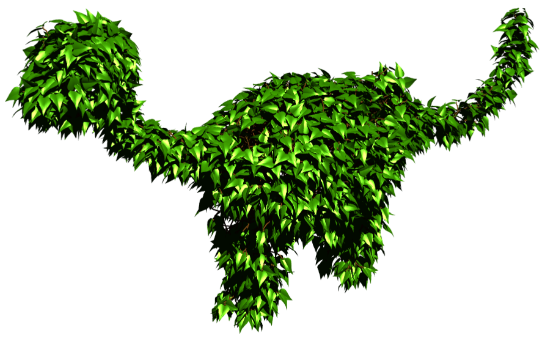
Figure 2: *A virtual topiary model created by Open L-systems [MP96]. Courtesy of R. Mech and P. Prusinkiewicz.*

Pure PM usually does not allow for the communication exchange between the plant and its environment (exogenous control). However, exogenous control, namely self shadowing and collision detection, are among the most important factors defining the final shape of the plant. This has been addressed by the environmental sensitive automata of Arvo and Kirk [AK88], where plants are simulated as autonomous particles that compete for resources. The traces of each particle define the individual branches. Particles can branch and also produce leaves. A similar approach was proposed by Greene [Gre89], where a voxel space is used as an additional data structure for quick illumination evaluation and collision detection. This approach was later used for interactive plant modeling by Benes et al. [BM02, BAv09]. Recently, simulation for resources has been used for modeling leaf venation patterns [RFL*05], and later extended for simulation of plant competition to create individual plants and small plant colonies [RLP07, PHL*09], where plants compete by

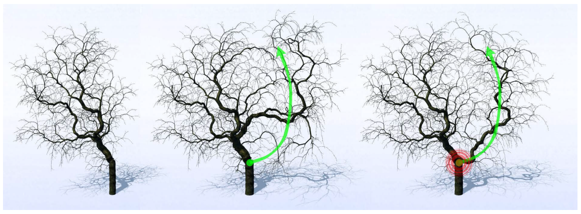

Figure 3: *Comparison of non-selective and selective growth in a tree created with the TreeSketch framework [LRBP12b]. Courtesy of S. Longay, A. Runions, F. Boudon, and P. Prusinkiewicz. © 2012 EG, reprinted with permission.*

growth for a predefined random set of local attractors in space. Most recently, *plastic trees* allow for an interactive modeling of arbitrary input trees that react to external conditions as if they were grown in the given environment [PSK*12].

Deussen et al. [DHL*98] described a first ecosystem simulation model to populate a height map with vegetation using competition for resources on the level of individual plants. The virtual plants grow and seed and over time their local area of influence increases. If two or more plants collide, their survival is evaluated and one plant is eliminated. This approach has been extended in different directions, such as adding virtual agents that affect the ecosystem [AD05], simulating vegetation in urban areas with artificial management of certain areas [BMJ*11], or combining with real digital elevation maps [Ham01].

PM can be combined with sketches, as shown in several methods for sketch-based modeling of floral diagrams [IOI06a, IOI06b]. Recently, a tablet-oriented approach for sketch-based procedural plant modeling, extending the work of Palubicky et al. [PHL*09], was presented by Longay et al. [LRBP12b] (see Figure 3). Such sketch-based PM methods are sound examples of the power of PM, providing a good control over the design process. However, extensive knowledge of PM is still required in order to use such a system to its full extent.

4. Water bodies

The topic of procedural generation of water bodies, such as rivers, lakes, streams, oceans and waterfalls, is somewhat under-addressed in PM literature. However, several authors have proposed algorithms specifically for generating rivers. Typical strategies for river generation can be divided into two categories: generating a river network as part of a height map generation algorithm, or as a post-processing step on an existing height map. For the former, a generated river network forms a basis from which a height map is inferred. For the latter, a height map is analyzed to find potential stream routes from mountains into valleys.

Kelley et al. were the first to generate a river network as the basis for a height map [KMN88]. They start with a single river path and recursively branch and subdivide it, resulting in a stream network. This network then forms a skeleton for the height map, which is filled using a scattered data interpolation function. The climate type and the soil material influence the shape of the stream network.

Prusinkiewicz and Hammel combine the generation of a curved river with a height map subdivision scheme [PH93]. On 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. A downside of the method is that the river is placed at a constant elevation level, and thus carves deep through a mountainous landscape. Moreover, this method is fully automatic and can only be controlled by setting the input parameters.

An approach that does not suffer from these limitations, described by Belhadj and Audibert, creates a height map with mountain ridges combined with river networks [BA05]. Starting with an empty map, they place pairs of ridge particles at a particularly high elevation and move them in opposite directions in several discrete steps. 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 between ridges and rivers are filled with an inverse midpoint displacement technique. This is a fast and effective method for a specific type of terrain, e.g., steep mountain ridges with valleys featuring a dense river network, however its application to other types of terrain is limited. Also, this method does not provide rivers that are hydrologically valid.

The above-described methods provide little control on the resulting river path. The interactive method presented by Huijser et al. proposes a convenient way of defining and controlling a river path [HDBB10]. A line defines the river path and, together with the specification of typical cross section profiles, meant to be imposed and interpolated along that path, the method results in a detailed geometric representation of the river and its banks. The price of this additional control, however, is that the method does not always guarantee physical nor geographical plausibility of the resulting river path. In this sense, a solution that better combines control and plausibility of river paths is the procedural sketching method presented by Smelik et al. [STdB11]: a designer can input a sequence of control points on a terrain as an indication of the desired path, and the algorithm will derive and embank in the terrain a feasible and plausible path that approximates that input sketch.

Recently, Genevaux et al. [GGG*13] presented an ap-

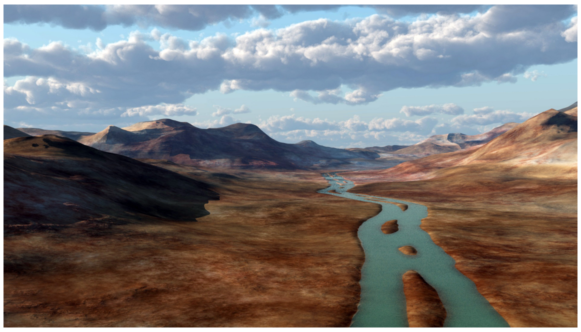

Figure 4: *A landscape shaped by a river network, created by the hydrology-based method by Genevaux et al. [GGG*13]. Courtesy of J.D. Génevaux, E. Galin, E. Guérin, A. Peytavie, and B. Benes.*

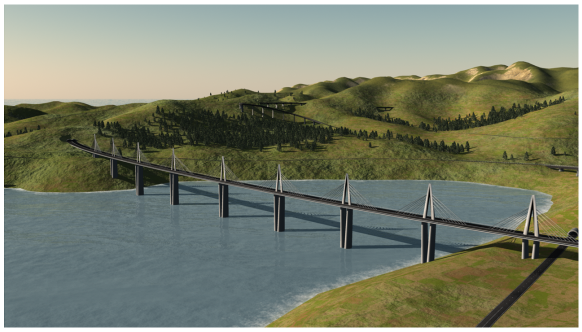

Figure 5: *A hierarchical road network produced by the method of Galin et al. [GPGB11]. Courtesy of E. Galin, A. Peytavie, E. Guérin, and B. Benes.*

proach that follows the idea of Kelly et al. [KMN88] by creating the river network first and then completing it by terrain, closely following the rules from hydrology, as shown in Figure 4. Procedural blocks are then used to create a 3D model of the terrain that is stored as a CSG-like tree where leaf nodes represent the features of the terrain and internal nodes correspond to procedural operations. This approach also suffers from a limited user control.

5. Roads

Procedural road generation has primarily been addressed in the context of procedural cities, so the generation of interstate and country roads still requires further attention. Important requirements for interstate or country road generation are that the trajectories fit well with the local terrain and the curvature of the roads is constrained, in order to allow vehicles to travel at constant speeds. Procedural road generation methods in this category are either a) user-assisted, where road 3D splines that minimize local elevation changes are fitted in between sketch strokes or control points, or b) based on path finding techniques from AI research, such as A*, where cost-functions can encode the desire to maintain constant elevation and curvature.

McCrae and Singh present a method for converting sketched strokes to 3D spline-base roads that are automatically fit to the terrain [MS09]. Their system also creates junctions and viaducts for crossing roads. Kelly and McCabe plan the precise path of their main roads between the user set nodes to have an even change in elevation as much as possible [KM07]. The influence of the underlying elevation profile is taken into account with varying degrees of precision as both methods take only basic measures to avoid too steep roads and roads through water bodies.

It is not enough to simply place a road on the terrain: the landscape needs to be properly modified to accommodate for the road. Early work by Amburn et al. already addressed the problem of fitting roads with terrain: on a coarse level, the road follows the elevation profile of the terrain, and on a fine level, the terrain must be modified to match locally with the road embankment profile [AGW86]. This specific integration problem was also addressed by Bruneton and Neyret, who propose a shader-based system for real-time integration of Geographic Information System vector features, such as road and rivers, into a DEM [BN08]. They create a road profile displacement texture based on footprint geometry, and integrate the profile by blending this texture with a height map texture. Galin et al. first extend this by also removing any vegetation along the road [GPMG10]. Subsequently, Galin et al. [GPGB11] proposed an A*-based road generation method that uses a cost function to encode the influence of slope, water bodies, and vegetation on the trajectory of the road; an example of the result is shown in Figure 5.

6. City layout

A procedural city is a complex and often hierarchically structured model. Its generation procedure typically operates in a top-down fashion, possibly starting at a very high level by generating a broad division of the available land into city zones, such as city center and outskirts, and ending with placing individual building parcels. The topic of procedural cities has received much attention in the last decade, starting with the seminal work of Parish and Müller [PM01]. Many of the earlier procedural city generation methods take a road network, often generated by combining typical road patterns, as the basic city structure. In recent years, more advanced methods considers aspects as urban land use, traffic models or even agent simulations. Kelly and McCabe present a survey of several approaches for generating urban environments [KM06]. A practical overview of the state of the art in procedural modeling of cities can be found in [WMV*08].

Road networks for cities can be generated using a variety of methods, including pattern-based approaches [GPSL03, SYBG02], L-systems [PM01], agent simulations [LWW03, LRW*06], and tensor fields [CEW*08]. The simplest pattern-based technique is to generate a dense square grid, as seen in the work of Greuter et al. [GPSL03] with random noise to create a less repetitive network. However, the realism and expressiveness of this technique is inherently limited.

A more elaborate method to create roads is by means of templates, as proposed by Sun et al. [SYBG02]. They observe several frequent patterns in real road networks and use them as basic building blocks. For each pattern, there is a corresponding template: a population-based template (implemented as the Voronoi diagram [Vor08] of population centers), a raster and radial template, or a mixed template. To create the skeleton of the road network, highways are first generated using the pattern templates. Next, highways are curved to avoid large elevation gradients and the regions they encompass are filled with a grid of streets.

Similar to plant models, a road network can be viewed as a growing structure, and can be simulated by a rewriting system. Parish and Müller use an extended Open L-system to grow a road network [PM01]. The L-system is goal-driven; its goals are population density (roads try to connect population centers) and specific road patterns, for example the raster or the radial pattern. This 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. Smaller streets are inserted into the remaining areas using a grid. This method is fully automatic and provides limited control only by defining density maps of population.

Vanegas et al. [VABW09] use a simulation-based approach to allow for high-level user control over the design of the generated city. They steer the city design by 'painting' jobs, city population, and main roads. A simulation engine interactively creates the corresponding city geometry. This method provides better control than the methods above. However, the control is limited to a high level of abstraction and, therefore, small features cannot be manipulated. Similar to this approach is the work of Weber et al. who also use simulation to address the problem of expanding cities over time [WMWG09]. Their cities grow their road network into nearby available land. Both approaches are fast and interactive so that the user can guide the simulation by changing roads or painting land use values on the terrain.

An example-based system for city modeling was presented by Aliaga et al. [AVB08]. The examples from existing cities are first analyzed and a stochastic procedural model is inferred. The procedural model generates 2D models that have statistically identical road and intersection distribution and road geometry. Once the street patterns are generated, parcels are generated and filled by warped images from the input examples. This approach allows for city layout overlay, extension, and interpolation. Small features can be added manually.

Lechner et al. introduce an agent-based approach, in which they divide the city into areas including not only residential,

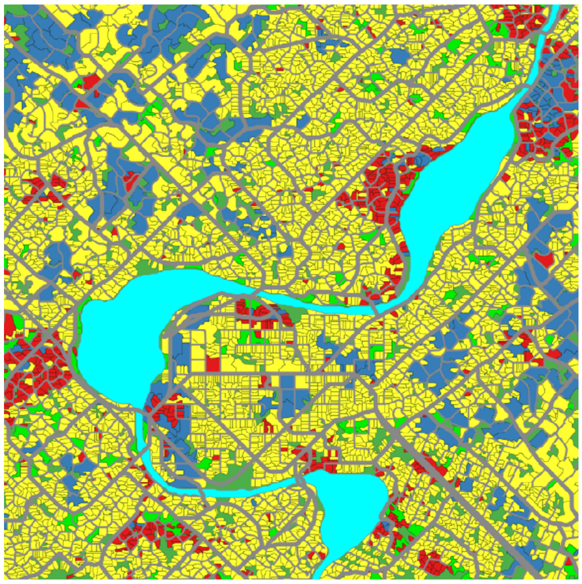

Figure 6: *A procedural city model created with the agent-based technique of Lechner et al. [LRW*06], showing residential (yellow), commercial (red), industrial (blue) and park (light green) development. Courtesy of T. Lechner, B. Watson, and U. Wilensky. © 2006 ACM, reprinted with permission.*

commercial and industrial areas, but also special areas like government buildings, squares, and monuments [LWW03]. They place two agents, named the extender and the connector, at a seed position in the virtual world. The extender searches for unconnected areas in the city and the connector tries to connect them. In their follow-up work, the authors extend this method with agents that are responsible for constructing main roads for fast connections through the city, and agents that develop small streets [LRW*06]. This method gives plausible results, but a disadvantage is its very long running time. A vectorized output can be found in Figure 6. This method gives some direct control over the trajectory of a generated road. As districts, blocks and parcels are defined by the city's road network, a typical method for a designer to influence the city structure is by interactively manipulating the road network.

Kelly and McCabe introduce the interactive city editor CityGen, in which a user defines the main roads by placing nodes on the 3D landscape [KM07]. 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.

A similar system by de Villiers and Naicker [dVN06] allows users to create a road network and city blocks using sketch strokes, and interprets a set of sketch gestures that modify the properties of the city blocks (e.g., population size, function). Lipp et al. present two graph merging operations for city road networks [LSWW11]. The first technique is

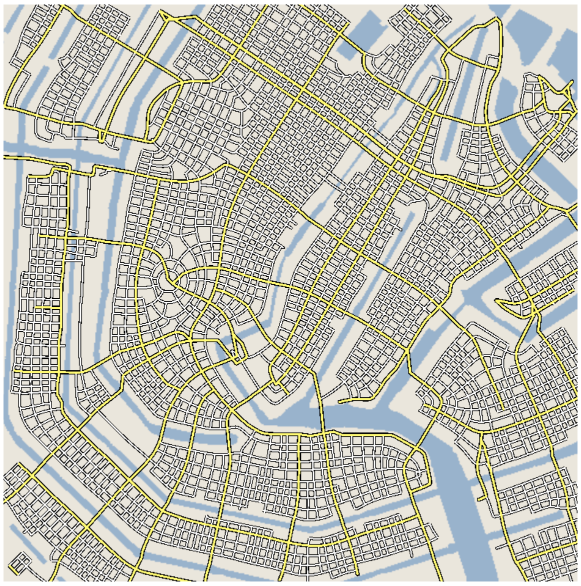
Figure 7: *A street graph traced from a complex tensor field [CEW*08]. Courtesy of G. Chen, G. Esch, P. Wonka, P. Müller, and E. Zhang.*

specialized for locally repairing the road network, after a designer has made a small change to a single road. The second approach merges two road network layers using the graph-cut algorithm. The first layer contains the part of the network that was changed by the designer, and the second contains a procedurally generated network. By using proper merge priorities for roads in the cut, they are able to merge both layers into one network with plausible transitions. This graph-cut merging technique can also be used to lock a subset of the road network.

Chen et al. propose an interactive modeling method for road networks by the use of tensor fields [CEW*08]. Common road patterns (grid, radial, along a boundary) are generated from a tensor field by tracing the streamlines from seed points in the major eigenvector and perpendicular direction until a stopping condition is met. 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. An example of a generated road network is shown in Figure 7.

Most of the above-described algorithms generate the city layout by first creating the road network and then by generating lots and parcels, which can subsequently be populated by procedural buildings as described in Section 7. Lots and parcels are defined by polygonal regions enclosed by the streets and roads. Subdivision of these regions results in building lots, for which different subdivision methods exist, see e.g., [PM01] or [KM07]. A recent work by Vanegas et al. introduces two new parcel subdivision methods [VKW*12]: a splitting technique based on 2D oriented bounding boxes, and a subdivision method that starts from the straight skeleton of the parcel polygonal shape.

Groenewegen et al. 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 [GSdB09]. The method takes into account a large number of relevant factors, including the historic core of the city and the attraction/repulsion that certain types of features (e.g. hillsides, coastlines, rivers) can have on certain types of districts (e.g. industrial or high-class residential districts).

The above methods are designed for generating large and structured urban environments, and, as such, are not suitable for generating small villages, farm lands and informal settlements. Glass et al. describe several experiments of replicating the road structure found in South African informal settlements using a Voronoi diagram for the major roads, in combination with either L-systems or regular subdivision for the minor roads [GMB06]. They were able to recreate the observed patterns. For small villages and agricultural settlements, Emilien et al. propose an iterative process [EBP*12] that grows a specific type of village, considering the local terrain constraints. Starting from an initial road network skeleton and depending on the desired type of village, in each iteration, a number of building seeds are placed at locations that score well on a number of weighted criteria: terrain height, slope, road accessibility, distance to neighbors, and special buildings (churches, forts). These seeds are connected to the existing road network, reusing existing roads and introducing cycles where possible. Finally, a building parcel shape is constructed from each seed point.

7. Buildings

Procedural generation of buildings is one of the best-developed PM areas. Most methods in this category use some form of formal rewriting system, such as an L-system, a split grammar or a shape grammar, as the basis for generating a 3D building model out of a 2D parcel shape. These methods can be employed to create detailed and convincing buildings, but require much authoring effort. Some alternative methods attempt to automatically reconstruct grammars from real world datasets, such as photographs of building facades.

A special sub-area of building generation is procedural facade generation [MZWG07]. Facades are usually modeled by using 2D split grammars and various forward [XFT*08] and inverse approaches [BSW13, ZXJ*13] exist.

Parish and Müller start with a rectangular floor plan and apply an L-system to refine the building [PM01], and Greuter et al. generate office buildings by combining several primitive shapes into a floor plan and extruding these to different heights [GPSL03]. Both approaches are useful for relatively

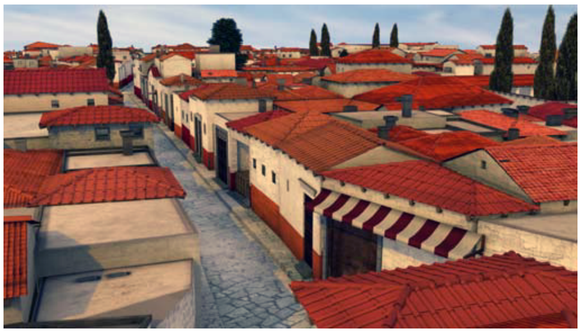

Figure 8: *A typical street in the procedural Pompeii model [MWH*06]. Courtesy of P. Müller, P. Wonka, S. Haegler, A. Ulmer and L. van Gool. © 2006 ACM, reprinted with permission.*

simple office building models. Coelho et al. [CdSF05] propose an urban modeling process that is based on L-systems and generates a tree-like description of the overall scene structure from external data. L-systems are used to generate detailed building models that emerge from the abstract set of data.

Wonka et al. introduced the parametric context-free *split grammar* designed to produce building models [WWSR03]. Split grammars explicitly associate a geometric shape to each symbol and their parameters can control the rewriting or the style of the generated building. Within one building model, the style can differ per floor (e.g., an apartment building with shops on the ground floor). The method focuses mostly on generating coherent and believable facades for relatively simple shaped buildings. Müller et al. extended the split grammars to Computer Generated Architecture (CGA) [MWH*06], (see Figure 8). CGAs are *shape grammars* [SG71] specifically designed for building facades. Shape grammars have been used and described before, especially in the architectural domain [KE81, Cag96, Kwo03]. Architects have used shape grammars as languages of design, supported by a vocabulary of shape rules. Shape rules are specified as spatial relations, where one or more shapes on the right-hand side of the rule is produced and replaces the symbol on the left-hand side (which conditions when the rule can be applied). The CGA shape grammar is specialized for modeling 3D buildings and its operations include the possibility of creating roofs and rotated shapes. It typically starts with extruding a building lot polygon into a volumetric shape, which is divided into floors. The resulting facades are further subdivided, through shape rules, into walls, windows and doors. Variation can be created using conditional or stochastic rule application, shape parameters and random number generation. Shape grammars are presently the most developed, used and compact method for building representation.

Although CGA shape grammars can generate visually convincing building models, Finkenzeller and Bender note that they miss semantic information regarding the role of each shape within the complete building [FB08]. They propose to capture this semantic information in a typed graph. In related work, Finkerzeller presents in more detail the generation of facades and roofs in this system [Fin08].

Yong et al. describe a method to create vernacular-style Southeast Chinese houses using an extended shape grammar [YCZY04]. 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. present an approach for reconstructing building facades from photographs [MZWG07]. Their method takes a single image of a facade of a real building as input, and reconstructs a detailed 3D facade model using a combination of imaging and shape grammar generation.

Shape grammars are a versatile and often successfully employed method for automatic creation of building facades. Still, defining a suitable shape grammar is complex and requires experience and in-depth knowledge of its geometry derivation technique. Addressing this, Lipp et al. propose a shape grammar editing system in which the effects of new rules are interactively visualized [LWW08]. Their approach, however, still requires the awareness of a grammar structure. This drawback is addressed by dataflow graph representations [Pat12, SMBC13].

8. Building interiors

To create a complete 3D building, both its exterior facade and its interior must be generated. Procedural methods to generate building interiors are very different in nature to the often grammar-based approaches for generating facades, and therefore are treated separately here. Within this topic, one can discern *floor plan generation* and *furniture layout solving*. The procedural generation of building floor plans has been the focus of several researchers, which has resulted in grammar, subdivision, graph layout, constraint-solving, and even machine learning approaches. Furniture layouts have been generated using either data-driven or constraint-based methods.

Rau-Chaplin et al. used shape grammars to generate floor plans in [RCMLS96]. Shape grammars are used to create a plan schema containing basic room units that are grouped into functional zones like public, private, or semi-private spaces. Individual functions are then assigned to each room, which are filled with furniture by fitting predefined layout tiles from a library of individual room layouts. Variation of this method is limited by the available predefined tiles.

Hahn et al. present a subdivision method for real-time generation of office buildings [HBW06]. The initial building structure is split into a number of floors and further subdivisions create a hallway zone and individual rooms. All areas are only generated when necessary, i.e., when in view. Furthermore, changes made in such a building are persistent: they are stored and executed again when an area is regenerated.

Marson and Musse introduce a different room subdivision method, based on *squarified treemaps* [MM10]. Starting with a 2D building outline and a set of rooms with desired area and functionality (which are chosen by the user), they recursively subdivide the outline into smaller areas, e.g., building shape, functional zones, and rooms. In a post-processing step, corridors are automatically created to connect unreachable rooms.

Instead of starting with a building outline and rewriting or subdividing this space into rooms, Martin first composes a graph of the connectivity of individual rooms in a building before transforming the graph into the spatial layout [Mar06]. Nodes of the graph represent the rooms and edges correspond to connections between rooms (e.g., doors and walls). The graph is transformed to a spatial layout, and for each node, a specific amount of 'pressure' is applied to make the room expand to the desired size. This method allows users to decide on types of rooms, size and connectivity necessary in the final layout.

Charman gives an overview of constraint solving techniques that can be applied to room layout generation, regarded as a space planning problem [Cha93]. The proposed planner works on the basis of axis-aligned 2D rectangles with variable position, orientation and dimension parameters, for which users can express geometric constraints, possibly combined with logical and numerical operators. Although feasible, the use of constraint solving techniques is quite complex for novice users and is not intuitive.

More recently, Merrel et al. proposed a method for generating residential building layouts [MSK10]. They use a Bayesian network, trained with real-world data, to expand a set of high-level requirements (e.g., number of rooms) into a complete architectural program (e.g., room adjacency, area, and aspect ratio). These architectural programs are then realized into the 2D shapes of the floor plans, through stochastic optimization over the space of possible building layouts.

Tutenel et al. applied a generic semantic layout solving approach to expansion-based floor plan generation [TBSd09], where every type of room is mapped to a class in a semantic library and for each of these classes, relationships can be defined. In this context, relationships define room-to-room adjacency. In addition, other constraints can be defined as well, e.g. place the kitchen next to the garden, or the garage next to the street. For each room to be placed, a rectangle of minimum size is positioned at a location where all defined relation constraints hold, and all these rooms expand until they touch each other. As in previous methods, the user can decide on rooms, size and constraints, however the use of semantics allows for a more intuitive way to express this information. The number of rooms and necessary room types do not need to be fixed, but can be defined based on the family size, characteristics and needs of the household, the size of the building lot and house area, number of floors, etc. Using the same approach and based on a semantic description, furniture can be added to the rooms, as shown in Figure 9.

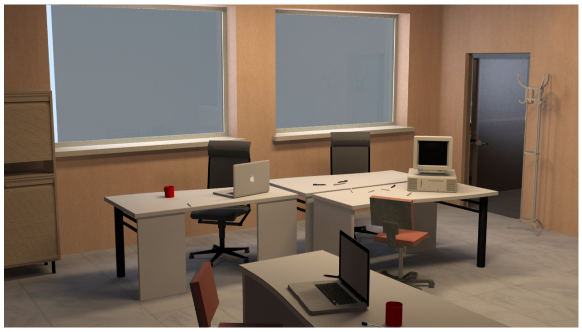
Figure 9: *An office scene furnished based on semantic scene descriptions [TBSd10]. Courtesy of T. Tutenel, R. Bidarra, R.M. Smelik, and K.J. de Kraker.*

Merrell et al. created an interactive furniture layout system that assists users by suggesting arrangements based on interior design guidelines [MSL*11]. The system incorporates the layout guidelines as terms in a density function and generates layout suggestions by rapidly sampling the density function. An example of a dining/living area is in Figure 10.

Yu et al. automatically synthesizes furniture layouts based on sensibly furnished example scenes [YYT*11]. They extract hierarchical and spatial relationships for various furniture objects from these examples, encoding them into variables associated with ergonomic factors, such as visibility and accessibility. These are assembled into a cost function that is optimized.

9. Some available systems

In the previous sections, we reviewed both seminal and recent research on PM of features for virtual worlds. To complement this review, in this section we discuss a selection of PM tools, either commercial or freely available, which we feel are representative of the state of the art. With this, we aim at giving some insight into how PM research results are already being applied in the industry, and what the focus, capabilities and limitations of the current procedural tools are. The tools we will discuss are mostly aimed at the generation of terrain heightmaps, plants or ecosystems and urban environments.

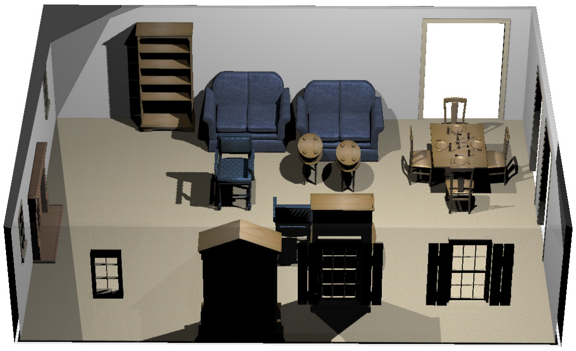

Figure 10: *A basic generated layout, which users can build on to create their desired scene [MSL*11]. Courtesy of P. Merrell, E. Schkufza, Z. Li, M. Agrawala, and V. Koltun. © 2011 ACM, reprinted with permission.*

Numerous procedural tools exist for generating height maps. From this large selection, we review three tools that have been around for several years: TerraGen [Pla13], GeoControl [Ros13], and L3DT [Bun13]. We selected these tools because they have a wide user base and advanced editing capabilities.

A scene in TerraGen is stored as a network of nodes, where each node maps to an operation, such as noise generation, a filter, or a mathematical function [Pla13]. A designer composes and configures the network in such a way that it generates the desired elevation profile. The expressive power of this system is significant. However, in order to use this tool effectively, in-depth knowledge of mathematics and noise generation is needed. Therefore, it is most suitable for designers with extensive technical expertise, focusing on creating aesthetically pleasing landscapes.

GeoControl [Ros13] is a height map editor that iteratively generates elevation data using a subdivision algorithm. Designers define the noise characteristics in each subdivision step. Additionally, filters, such as erosion or smoothing, can be applied on top of this basic noise algorithm. Users can control the generation by defining an isoline, setting its elevation, and the noise characteristics of the transition zone around it. A mountain ridge with these properties is generated along this line that blends in with the existing height map. GeoControl's isolines can, with practice, be used to draw height profiles that adequately match designer's intent. Still, the modeling process can be quite complex and the quality of the results depends on knowledge of the effect of parameters and the dependencies between generation steps.

L3DT [Bun13] allows a user to design a height map by drawing on a grid map using a brush. The brush consists of a set of generation parameters such as the elevation, the amount of erosion, the roughness of the terrain, whether it is a source of water, and a climate profile. Each grid cell is expanded to 64×64 height map points in the resulting height map by applying noise, erosion and water flooding algorithms. Climate profiles are used for generating a large texture that is mapped on the height map, by specifying the types of soil material (e.g., grass, rock) and the conditions under which they occur (e.g., elevation range, slope range, water level). After the height map is generated, a scoring mechanism determines the placement of materials based on the climate profile. L3DT provides a very high level of interaction making it a versatile tool.

Another feature for which some successful tools have been developed is vegetation. XFrog is a procedural plant modeler [Gre13], which is based on the modeling method by Linterman et al. [LD99]. XFrog combines procedural model definition with a strong user interaction. The basic modeling structure is a mathematical tree with nodes that represent procedural or geometrical features. The user can edit topological and geometrical properties and can preview all aspects of the model. The set of modeling parameters can be located per instance or can be inherited, allowing for local or global operations. XFrog also supports level of detail and ecosystems modeling.

SpeedTree [Int13] is a middleware system for modeling and rendering of large amounts of detailed procedural vegetation. The modeling application can procedurally generate trees in various levels of detail, based on a number of input parameters, such as branch length and angles, or based on a pre-defined species template from a library. Furthermore, designers can edit generated trees using hand drawn features. The generated vegetation is efficiently rendered by the SpeedTree engine plugin, which animates and manages the level of detail of the vegetation. The SpeedTree package has widespread application in the entertainment gaming and movie industries.

Plant Factory, by e-on software [eos13b], is a tool to model, animate and render 3D vegetation. Using a graph-based editor, plant or tree species can be created that can generate instances of any age and at any season. The user can manually tweak these instances, e.g. by changing the general shape, adding or prunning branches, or adding some vines on the surface of the tree. By parametrizing wind speed and direction, animation can be automatically added to the plant models.

Another product by e-on software, called VUE [eos13c], is a more integral tool to easily create full 3D landscapes including terrain, soil, vegetation and water surfaces. This product integrates several procedural generation techniques that help modelers create the landscape they need. Some of the features of this product are: *Eco Painter*, which allows users to paint trees, plants, bushes or moss on top of a terrain (or other structures like houses); *EcoSystem*, which populates entire areas with matching vegetation and terrain textures; and *Zephyr*, which adds animation to the scene based on wind parameters. Finally, e-on also developed Carbon Scatter [eos13a], a software tool to randomly scatter around objects picked from a given set, based on both user-defined and random parameters. This is mostly useful for placing plants and trees on landscapes, but could also be used to spread other objects in a scene.

CityEngine [Pro13] is a commercially available city generator based on the CGA grammar [MWH*06]. On one hand, CityEngine allows for interactive modeling by providing various features, such as intelligent resizing of a part of a city or a building, automatic content generation, copy and paste, etc. On the other hand, the system allows for user assisted writing of CGA rules and their interpretation.

A generic approach to PM is provided by the Houdini modeling tools [Sid13]. With Houdini, designers create procedural generators out of small procedural building blocks. These building blocks are often basic mathematical or geometric operations. A visual editor is used to compose the individual building blocks and to connect them as nodes of a network. A graph of primitive operations can be encapsulated into a single operation node, which allows designers to create reusable, high-level operations. Houdini is a versatile tool for creating new procedural methods in a visual way; however, to be used effectively, one must have advanced knowledge on how to design such a procedure.

10. Discussion

PM has a long history that has delivered many high-quality results and is an increasingly active research field. We have also seen that PM tools are available in both the public and commercial domains, and that they are successful for specific areas and purposes. However, contrary to what one might expect, current application of PM in practice is still very limited for, at least, the following two main reasons:

1. Procedural methods are often defined by a set of *non-intuitive* rules and input parameters, which can be hard to grasp and require in-depth knowledge of the technique internals. Their effect on the output is not always clear and predictable. A small modification in a rule or parameter value can cause a chain reaction of changes that propagates through the entire model. As a result, PM methods typically provide rather *limited user control*. To some extent, using procedural methods often comes down to trial and error, which becomes even more cumbersome when their performance hinders interactive use.
2. Heterogeneous procedural methods typically do not work together at all, as they yield *specialized output*, i.e., restricted to generate models of one specific class of features. Therefore, integrating these models still involves an increasingly large amount of manual effort.

In this section, we discuss how current research is addressing these challenges, identifying the main open issues and the most promising research directions (Subsections 10.2 and 10.3). To facilitate this, we start by summarizing and analyzing what has been done using which PM techniques for which features (Subsection 10.1).

10.1. Categorization of procedural methods

Table 1 gives a categorization of the PM methods discussed in this survey, classified according to their main feature of application and their underlying technique family or category. We opted for identifying six such categories, represented by the columns of Table 1, although not all PM methods fit perfectly into this scheme, as they can, in some cases, combine techniques from several families.

Some procedural approaches are based on purely stochastic methods; most of them use noise generators to create patterns or fractal structures. A typical representative of these techniques is the work of Musgrave et al. [MKM89], using fractional Brownian motion to create terrains or textures. Pure stochastic methods find their application mostly in terrain generations and, recently, also in optimization of interior designs.

Artificial intelligence techniques, such as path finding or planning algorithms, complement various procedural algorithms. As can be seen in Table 1, these techniques have been applied in terrain generation, vegetation, road and city layouts, and buildings.

Simulations are on the edge of purely procedural methods and some other areas such as physics, urban modeling, or biology. Simplified simulations, frequently reduced to purely procedural models, have been successfully applied in terrain modeling using erosion, vegetation simulation by competition for resources, and city layout design by simulation of city growth and resource allocations.

Grammar-based methods can be considered pure procedural approaches and their applications can be found in nearly every application area. The two most important approaches are L-systems, used for vegetation modeling and simulation, and shape grammars, used for building modeling.

In some cases, PM techniques are driven by examples or patterns found in existing content or real-world data.

Various other PM methods employ (a combination of) computational geometry algorithms, directly using some programming language. Often these involve subdivision algorithms, space filling methods, or techniques inspired by image processing. Except for buildings, such spatial layout techniques are used for every feature. These methods can be further classified into subdivision, expansion (growth) and optimization-based.

One can also compare procedural models from the perspective of the data they process. For example: noise generators can work with virtually arbitrary data sets; terrain generators are usually restricted to DEM, meshes or volumetric representations; and more specialized algorithms, e.g. for building structures, may also require specialized data to work with.

	Stochastic	Artificial Intelligence	Simulations	Grammars	Data-driven	Computational Geometry
Terrain	FFC82, Per85a, Mil86, MKM89, EWM*03, SS05, KU07, dB09, STdB10b	Sau06, DP10	MKM89, Mus93, BTHB06, WCMT07, ASA07, ŠBBK08, KBKv09, PGGM09, VBHv11, GGG*13	PH93	Sau06, ZSTR07	SS05, KU07, Bel07, GMS09, HGA*10, BMV*11
Vegetation		AD05	Hon71, AK88, Gre89, DHL*98, BM02, RFL*05, RLP07, BAv09, PHL*09, PSK*12, BMJ*11	Lin68, PHM93, Pru86, PLH88, MP96, Pru97, LD99, Pru00, IOI06b, IOI06a, LRBP12b	LYO*10	Ham01
Water bodies	PH93, BA05			KMN88		PH93, BA05, HDBB10
Road networks		GPMG10, GPGB11		PM01		AGW86, KM07, MS09
City layout		LWW03, LRW*06	WMWG09, VABW09, EBP*12, VGDA*12	PM01	SYBG02, CEW*08, AVB08	GPSL03, GMB06, dVN06, KM07, GSdB09, LSWW11, VKW*12
Buildings		YK12		KE81, Cag96, Kwo03, WWSR03, YCZY04, CdSF05, LG06, MWH*06, FB08, Fin08, LWW08, Pat12, YK12	MZWG07	
Interiors	MSK10, MSL*11, YYT*11, Cho12			RCMLS96		Cha93, HBW06, Mar06, MM10, TBSd09, TBSd10

Table 1: *Classification of the discussed* PM *methods by virtual world feature and underlying technique family.*

Following the rows of Table 1, it is noticeable that a lot of attention has already been paid to terrains, vegetation, and city layout. More recently, buildings have received more attention and many new approaches are appearing every year. Except for rivers, procedural water bodies, such as oceans, lakes and their connections, stream networks, deltas and waterfalls, have received little attention to date.

10.2. Intuitive control

One of the most important unsolved problems of PM is the level of control the designer has over the input, generation of the model, and its final editing. In the worst case, the user control is limited to the definition of the procedural model and its parameters. The algorithm is executed and the results are generated. Enforcing, in this way, a particular intent is very tedious if not impossible. Several directions have been taken to improve these input challenges, including the use of sketch-based techniques, visual editors, and inverse PM.

Sketch-based techniques. Some PM methods are integrating increasingly more intuitive input techniques, bringing their use closer to the way of work for many creative professionals. Among them, notions like brushing and sketching, based on gestures or strokes, have a very strong appeal. So far, they have been deployed for interactive terrain editing with procedural brushes [dB09], road editing by means of sketch strokes [MS09], sketch-based input of mountain silhouettes [GMS09], procedural sketching of terrain features [STdB10b], and mixed-initiative multi-touch control of tree growth [IOI06a, IOI06b, LRBP12a]. These systems are among the most promising approaches to interactive PM.

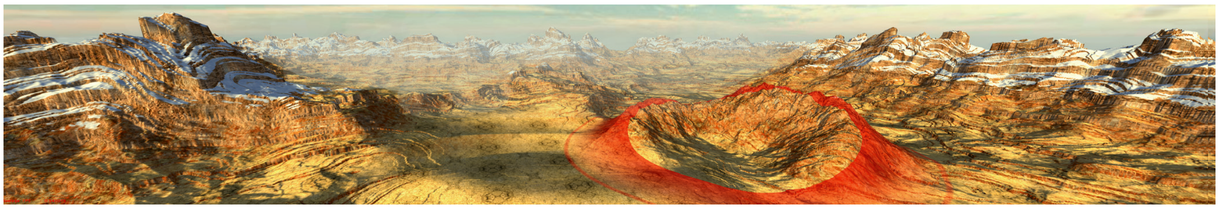

Figure 11: *Interactive terrain editing with procedural brushes [dB09]. Courtesy of G.J.P. de Carpentier and R. Bidarra © 2009 ACM, reprinted with permission.*

However, knowledge of procedural systems is still required in order to use them to their full extent.

Visual editors have been proposed to automate the generation of grammar production rules. An example is the work of Lipp et al. [LWW08]. Recently, Krecklau et al. [KK12a] introduced an editing mode to combine high-level architectural primitives, where editable parameters are manipulated through 3D handles. However, a limitation is that the creation of such primitives requires manual editing of grammar rules.

Visual node-based editors, providing a spatial insight on the flow of data being processed, have become standard for a variety of purposes and (commercial) systems, including material editors (e.g. Autodesk Maya), texture editors (e.g. Allegorithmic Substance Designer), script editors (e.g. Autodesk Softimage), and model creation and animation editors (e.g. Side Effects Houdini). The widespread use of such editors by non-technical professionals has recently prompted some researchers into exploring their suitability for the specification of grammar production rules. Examples of this are the work of Patow [Pat12], proposing a dataflow adaptation for shape grammars that bridges the dissociation among rules, and the work of Silva et al. [SMBC13], focusing on the encapsulation of such graphs into reusable, semantically-rich component nodes that can be more easily assembled through dataflow filters and constraints.

In recent years, considerable progress has been made in improving user control for specific features and methods. Often, more control can be offered by providing designers with better *interactive* modeling facilities. For height-map generation, for instance, there are several novel methods with improved user control, especially procedural brushes [dB09] (see Figure 11), terrain sketching [GMS09], and parameterized curves [HGA*10]. The work of Benes et al. makes the definition and the execution of L-systems more controllable and accessible [ŠBM*10, BvMM11]. For urban road networks, the intuitive modeling method of Chen et al. helps one to quickly define the desired road patterns [CEW*08], and the merging operations of Lipp et al. allow for fine-grained edit operations on such road networks [LSWW11]. Guided PM [BvMM11] enables designers to draw constraining outlines, called guides, for a growing L-system, and uses token passing to affect connections between these guides.

Inverse procedural modeling attempts to find a procedural representation of a given input model. Depending on the problem definition, the task can be a) to find the model rules and the parameters, or b) the procedural model is given and its parameters are learned. The most important advantage of the inverse approach is that it leaves the designer entirely out of the process: the PM is opaque, the designer does not need to know about it at all. At the same time, once found, the procedural representation has all advantages (and disadvantages) of PM.

One of the first approaches to inverse PM was the work of Aliaga et al. [ARB07] who attempt a building reconstruction by means of L-systems. More recently, full parametric L-systems have been generated for 2D vector art [ŠBM*10], and this work has been extended by Bokeloh et al. [BWS10]. Various approaches that attempt to estimate parameters of an existing procedural model have recently been presented [TLL*11, VGDA*12].

Regarding the control of PM editing operations by designers, one of the main challenges is their lack of integration with the kind of manual editing operations that they are very much used to. The dilemma is clear: if you automate too much, designers will feel too constrained and uncomfortable using the tool; if you allow plenty of manual interventions, how does one keep track of them throughout each model regeneration? Smelik et al. [STdB10a] presented a first systematic description of this tension, based on distinguishing levels of granularity of manual operations.

A different but related user control issue is the reversibility of operations. Designers very often tend to (ab)use *do-undo* combinations, in order to assess the net effect of a particular operation: if it does not yield the desired result, they just backtrack. However, many PM methods are not always reversible, and it is therefore not always possible to exactly restore the previous model situation.

Finally, the performance of PM methods is sometimes also a challenging obstacle for designers. Despite significant improvements in recent years, procedural methods are often not fast enough to provide feedback at interactive rates. This, in turn, may strongly hinder the above mentioned use of trial and error operations, thus making the modeling process more cumbersome.

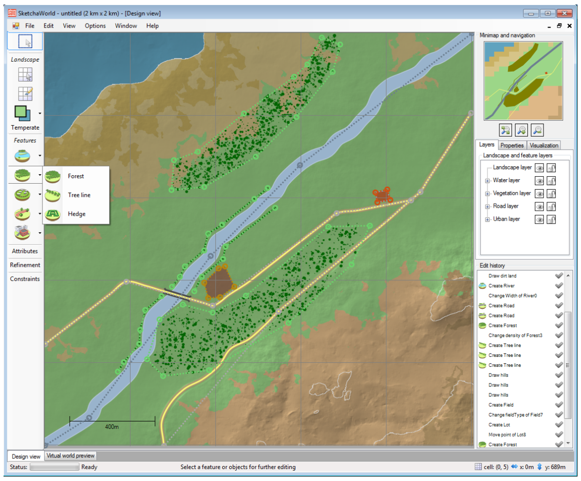

Figure 12: *SketchaWorld's interface, conceived for declarative modeling of virtual worlds [STdB11]. Courtesy of R.M. Smelik, T. Tutenel, K.J. de Kraker, and R. Bidarra.*

10.3. Method Integration

The vast majority of PM methods are specialized in generating one specific type of content or feature. To be really useful in the design of complete virtual worlds, all the heterogeneous content generated will have to be *fit*, *assembled* and *maintained* together within some framework or engine. Unfortunately, there is no present generic automation mechanism for any of these crucial tasks. Very little research attention has been given to the cooperation between distinct procedural methods nor, for that matter, to the integration of their output within the same modeling framework. Solving these challenges will likely be instrumental for the integration of PM methods in mainstream content production pipelines.

It is not enough to successfully 'engineer' the coexistence of heterogeneous content within a data framework. In fact, dissimilar features often influence each other; think, for example, of the interactions between river and hill, road and forest, road and river, etc. As a result, the execution of each PM method might have to take into account the location, type and properties of other features. So far, there has been no alternative proposed to handle such interactions other than describing *ad hoc* solutions to individual pairs of features. For example, specifically for roads, methods have been proposed for generating road embankments [BN08], and the construction of bridges or tunnels to cross bodies of water [GPMG10]. A more comprehensive approach to feature interaction handling proposed to have any two interacting features negotiate a solution based on the notion of feature extent [STdB11].

One step further than just keeping diverse features integrated in a virtual world model is the challenge of having different PM methods actively cooperate in the generation of a complex model. This would take advantage of using the most suited technique while preserving its individual qualities, in order to generate each model element in a coherent and constructive way. Again, a generic approach for this is still largely unexplored, and the problem has not been properly defined. One of the most challenging issues is the apparent mismatch in the way different PM methods describe the respective content, particularly regarding its local or global character. A recent proposal to approach this problem uses a central 'negotiator' module to broker among various cooperating PM methods [TSL*11]. Its application to the integral generation of consistent buildings combines shape grammars (for the facade), layout solving (for the floor plan) and semantic solving (for furniture placing).

Finally, once integrated, consistency of all generated features of the virtual world has to be maintained during subsequent modeling operations [BdST10, LSWW11]. This is an even further and higher-level objective, but one that is crucial for PM methods to become effective and useful for designers, whose intent should preferably be captured once and kept throughout their creative work. In fact, this challenge directly flows from the iterative nature of design: if it takes many refinement steps to incrementally generate all desired features in a virtual world, how can one guarantee that the result of a modeling step is not subsequently overruled? In other words, how easy will it be to get meaningless or corrupted output?

Integrating various procedural models and maintaining the consistency of all generated content while a virtual world is being designed is not only a very challenging and complex task, but it also goes far beyond the internals of individual procedural methods. Among other things, this requires that the modeling vocabulary provided to virtual world designers is able to map their explicit intent onto the particular procedures needed to achieve it. So far, there has been no truly generic proposal to solve this. Incremental but promising approaches in this direction include the work of Krecklau et al. [KK11, KK12b], as well as *declarative modeling of virtual worlds* [STdB11], a modeling approach that allows designers of virtual worlds to concentrate on stating what they want to create instead of on describing how they should model it (see Figure 12).

11. Conclusions

We have surveyed PM methods aimed at generating a large variety of features for 3D virtual worlds. We have observed that PM is an increasingly active research field that has delivered many high-quality results, including various successful tools for specific areas and purposes, which are available in both the public and the commercial domains. However, the acceptance and use of PM methods for virtual worlds has been persistently hindered by their considerable lack of intuitive control and by the disparity of isolated techniques that generate only specialized features.

Many research efforts have been put into increasing the ease of use of PM, particularly in improving the quality and degree of user control, due to its crucial enabling role for non-technical, creative professionals. Significant results have been achieved so far in this direction, leading to explore paths that combine proven concepts from such areas as human-computer interaction with novel and challenging techniques like inverse PM.

We believe that, among the requirements for a widespread acceptance of PM methods, the following will play a key role:

1. a procedural regeneration scheme that allows for local and global manual editing operations on procedurally generated models,
2. unification of procedural methods,
3. the ability to flexibly combine manually created content with procedurally generated models, and
4. the seamless integration of PM tools in current content development pipelines.

Driven by increasing consumer demands and expectations, and supported by rapid advances in computer hardware and display devices, the push for better, more detailed, extensive and visually convincing virtual worlds will likely continue in the coming years, putting additional pressure on their designers and artists. In this process, the role of PM methods is clear, but their potential is far from being exploited to its fullest. We can, therefore, expect that considerable research effort and novel results will continue to advance its frontiers, facilitating its deployment in increasingly more applications and domains.

References

[AD05] ALSWEIS M., DEUSSEN O.: Modeling and visualization of symmetric and asymmetric plant competition. In *Eurographics Workshop on Natural Phenomena* (Dublin, Ireland, 2005), Eurographics Association, pp. 83–88. 5

[AGW86] AMBURN P., GRANT E., WHITTED T.: Managing geometric complexity with enhanced procedural models. In *SIGGRAPH '86: Proceedings of the 13th Annual Conference on Computer Graphics and Interactive Techniques* (New York, NY, USA, 1986), ACM, pp. 189–195. 6, 13

[AK88] ARVO J., KIRK D.: Modeling plants with environment-sensitive automata. In *Proceedings of Ausgraph'88* (Melbourne, Australia, 1988), pp. 27–33. 4, 13

[ARB07] ALIAGA D. G., ROSEN P. A., BEKINS D. R.: Style grammars for interactive visualization of architecture. *IEEE Transactions on Visualization and Computer Graphics 13*, 4 (July 2007), 786–797. 14

[ASA07] ANH N. H., SOURIN A., ASWANI P.: 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* (New York, NY, USA, 2007), ACM, pp. 257–264. 3, 13

[AVB08] ALIAGA D. G., VANEGAS C. A., BENES B.: Interactive example-based urban layout synthesis. *ACM Transactions on Graphics: Proceedings of ACM SIGGRAPH Asia 2008 27* (December 2008), 1–10. 7

[BA05] BELHADJ F., AUDIBERT P.: 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* (New York, NY, USA, 2005), ACM, pp. 447–450. 5, 13

[BAv09] BENES B., ANDRYSCO N., ŠŤAVA O.: Interactive modeling of virtual ecosystems. In *Eurographics Workshop on Natural Phenomena* (Munich, Germany, 2009), Eurographics Association, pp. 9–16. 4, 13

[BdST10] BIDARRA R., DE KRAKER K., SMELIK R. M., TUTENEL T.: Integrating Semantics and Procedural Generation: Key Enabling Factors for Declarative Modeling of Virtual Worlds. In *Proceedings of the FOCUS K3D Conference on Semantic 3D Media and Content* (Sophia Antipolis - Méditerranée, France, February 2010). 15

[Bel07] BELHADJ F.: Terrain modeling: a constrained fractal model. In *AFRIGRAPH '07: Proceedings of the 5th International Conference on Computer Graphics, Virtual Reality, Visualisation and Interaction in Africa* (New York, NY, USA, 2007), ACM, pp. 197–204. 4, 13

[BF01] BENES B., FORSBACH R.: Layered data representation for visual simulation of terrain erosion. In *SCCG '01: Proceedings of the 17th Spring Conference on Computer Graphics* (Washington, DC, USA, April 2001), IEEE Computer Society, pp. 80–86. 2

[BM02] BENES B., MILLÁN E.: Virtual climbing plants competing for space. In *IEEE Proceedings of the Computer Animation 2002* (2002), Magnenat-Thalmann N., (Ed.), IEEE Computer Society, pp. 33–42. 4, 13

[BMJ*11] BENES B., MASSIH M. A., JARVIS P., ALIAGA D. G., VANEGAS C. A.: Urban ecosystem design. In *Symposium on Interactive 3D Graphics and Games* (New York, NY, USA, 2011), ACM, pp. 167–174. 5

[BMV*11] BERNHARDT A., MAXIMO A., VELHO L., HNAIDI H., CANI M.-P.: Real-time terrain modeling using cpu-gpu coupled computation. In *SIBGRAPI '11: Proceedings of the 24th Conference on Graphics, Patterns and Images* (Alagoas, Brazil, 2011), pp. 64–71. 4

[BN08] BRUNETON E., NEYRET F.: Real-time rendering and editing of vector-based terrains. In *Computer Graphics Forum: Proceedings of Eurographics 2008* (Crete, Greece, 2008), Eurographics Association, pp. 311–320. 3, 6, 15

[BSW13] BAO F., SCHWARZ M., WONKA P.: Procedural facade variations from a single layout. *ACM Transactions on Graphics 32* (2013), 8:1–8:13. 8

[BTHB06] BENES B., TĚŠÍNSKÝ V., HORNYŠ J., BHATIA S. K.: Hydraulic erosion. *Computer Animation and Virtual Worlds 17*, 2 (2006), 99–108. 3, 13

[Bun13] BUNDYSOFT: L3DT. Available from http://www.bundysoft.com/L3DT/, 2013. 11

[BvMM11] BENES B., ŠŤAVA O., MĚCH R., MILLER G.: Guided procedural modeling. In *Computer Graphics Forum: Proceedings of Eurographics 2011* (Llandudno, UK, 2011), Eurographics Association, pp. 325–334. 14

[BWS10] BOKELOH M., WAND M., SEIDEL H.-P.: A connection between partial symmetry and inverse procedural modeling. *ACM Transactions on Graphics (Proceedings of SIGGRAPH 2010) 29*, 4 (2010), 104:1–104:10. 14

[Cag96] CAGDAS G.: A shape grammar model for designing row-houses. *Design Studies 17*, 1 (1996), 35–51. 9, 13

[CdSF05] COELHO A. F., DE SOUSA A. A., FERREIRA F. N.: Modelling urban scenes for lbms. In *Web3D '05: Proceedings of* *the 10th International Conference on 3D Web Technology* (New York, NY, USA, 2005), ACM, pp. 37–46. 9, 13

[CEW*08] CHEN G., ESCH G., WONKA P., MÜLLER P., ZHANG E.: Interactive procedural street modeling. In *SIGGRAPH '08: Proceedings of the 35th Annual Conference on Computer Graphics and Interactive Techniques* (New York, NY, USA, 2008), vol. 27, ACM, pp. 1–10. 7, 8, 13, 14

[Cha93] CHARMAN P.: Solving space planning problems using constraint technology. In *NATO ASI Constraint Programming: Students' Presentations, TR CS 57/93, Institute of Cybernetics, Estonian Academy of Sciences* (Tallinn, Estonia, 1993), pp. 80–96. 10, 13

[Cho12] CHOJNACKI S.: Scoring functions for automatic arrangement of business interiors. In *SIGGRAPH Asia 2012 Technical Briefs* (2012), ACM, p. 27. 13

[dB09] DE CARPENTIER G. J., BIDARRA R.: Interactive gpu-based procedural heightfield brushes. In *FDG '09: Proceedings of the 4th International Conference on the Foundations of Digital Games* (Florida, USA, April 2009). 4, 13, 14

[DHL*98] DEUSSEN O., HANRAHAN P., LINTERMANN B., MĚCH R., PHARR M., PRUSINKIEWICZ P.: Realistic modeling and rendering of plant ecosystems. In *SIGGRAPH '98: Proceedings of the 25th Annual Conference on Computer Graphics and Interactive Techniques* (New York, NY, USA, 1998), ACM, pp. 275–286. 5, 13

[DP10] DORAN J., PARBERRY I.: Controlled procedural terrain generation using software agents. *IEEE Transactions on Computational Intelligence and AI in Games 2*, 2 (June 2010), 111–119. 3

[dVN06] DE VILLIERS M., NAICKER N.: *A Sketching Interface for Procedural City Generation*. Tech. rep., Department of Computer Science, University of Cape Town, November 2006. 7, 13

[EBP*12] EMILIEN A., BERNHARDT A., PEYTAVIE A., CANI M.-P., GALIN E.: Procedural generation of villages on arbitrary terrains. *The Visual Computer 28* (2012), 809–818. 8, 13

[eos13a] E-ON SOFTWARE: Carbon scatter 2. Available from http://www.carbonscatter.com/, 2013. 11

[eos13b] E-ON SOFTWARE: Plant factory. Available from http://www.e-onsoftware.com/products/plant_factory/, 2013. 11

[eos13c] E-ON SOFTWARE: Vue 11. Available from http://www.e-onsoftware.com/products/vue/, 2013. 11

[EWM*03] EBERT D. S., WORLEY S., MUSGRAVE F. K., PEACHEY D., PERLIN K.: *Texturing & Modeling, a Procedural Approach*, 3rd ed. Elsevier, 2003. 1, 2, 3, 13

[FB08] FINKENZELLER D., BENDER J.: Semantic representation of complex building structures. In *CGV '08: Computer Graphics and Visualization* (Amsterdam, The Netherlands, July 2008), pp. 259–264. 9, 13

[FFC82] FOURNIER A., FUSSELL D., CARPENTER L.: Computer rendering of stochastic models. *Communications of the ACM 25*, 6 (1982), 371–384. 2, 13

[Fin08] FINKENZELLER D.: Detailed building façades. *IEEE Computer Graphics and Applications 28*, 3 (2008), 58–66. 9, 13

[GGG*13] GÉNEVAUX J.-D., GALIN E., GUÉRIN E., PEYTAVIE A., BENES B.: Terrain generation using procedural models based on hydrology. *ACM Transactions on Graphics (Proceedings of SIGGRAPH 2013) 4* (2013). 5, 6

[GLLD12] GALERNE B., LAGAE A., LEFEBVRE S., DRETTAKIS G.: Gabor noise by example. *ACM Trans. Graph. 31*, 4 (July 2012), 73:1–73:9. 2

[GM01] GAMITO M. N., MUSGRAVE F. K.: Procedural landscapes with overhangs. In 10th *Portuguese Computer Graphics Meeting* (2001), pp. 33–42. 2

[GMB06] GLASS K. R., MORKEL C., BANGAY S. D.: Duplicating road patterns in south african informal settlements using procedural techniques. In *AFRIGRAPH '06: Proceedings of the 4th International Conference on Computer Graphics, Virtual Reality, Visualisation and Interaction in Africa* (New York, NY, USA, 2006), ACM, pp. 161–169. 8, 13

[GMS09] GAIN J., MARAIS P., STRASSER W.: Terrain sketching. In *I3D '09: Proceedings of the Symposium on Interactive 3D Graphics and Games* (New York, NY, USA, 2009), ACM, pp. 31–38. 4, 13, 14

[GPGB11] GALIN E., PEYTAVIE A., GUÉRIN E., BENES B.: Authoring hierarchical road networks. *Computer Graphics Forum (Proceedings of Pacific Graphics) 30* (2011), 2021–2030. 6, 13

[GPMG10] GALIN E., PEYTAVIE A., MARCHAL N., GUÉRIN E.: Procedural generation of roads. In *Computer Graphics Forum: Proceedings of Eurographics 2010* (Norrköping, Sweden, May 2010), vol. 29, Eurographics Association, pp. 429–438. 6, 13, 15

[GPSL03] GREUTER S., PARKER J., STEWART N., LEACH G.: 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 Southeast Asia* (New York, NY, USA, 2003), ACM, pp. 87–94. 6, 7, 8, 13

[Gre89] GREENE N.: Voxel space automata: modeling with stochastic growth processes in voxel space. In *SIGGRAPH '89: Proceedings of the 16th annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 1989), vol. 23, ACM, pp. 175–184. 4, 13

[Gre13] GREENWORKS: XFrog. Available from http://www.xfrog.com, 2013. 11

[GSdB09] GROENEWEGEN S. A., SMELIK R. M., DE KRAKER K. J., BIDARRA R.: Procedural city layout generation based on urban land use models. In *Eurographics 2009: Short Papers* (Munich, Germany, 2009), Eurographics Association, pp. 45–48. 8, 13

[Ham01] HAMMES J.: 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* (London, UK, 2001), Springer-Verlag, pp. 98–111. 5, 13

[HBW06] HAHN E., BOSE P., WHITEHEAD A.: Persistent real-time building interior generation. In *Sandbox 2006: Proceedings of the ACM SIGGRAPH Symposium on Videogames* (New York, NY, USA, 2006), ACM, pp. 179–186. 10, 13

[HDBB10] HUIJSER R., DOBBE J., BRONSVOORT W. F., BIDARRA R.: Procedural natural systems for game level design. In *Proceedings of SBGames 2010* (Florianopolis, SC, Brazil, 2010), pp. 177–186. 5

[HGA*10] HNAIDI H., GUÉRIN E., AKKOUCHE S., PEYTAVIE A., GALIN E.: Feature based terrain generation using diffusion equation. In *Computer Graphics Forum: Proceedings of Pacific Graphics 2010* (2010), vol. 29, pp. 2179–2186. 4, 13, 14

[Hon71] HONDA H.: Description of the form of trees by the parameters of the tree-like body: Effects of the branching angle and the branch length on the shape of the tree-like body. *Journal of Theoretical Biology 31* (1971), 331–338. 4, 13

[Int13] INTERACTIVE DATA VISUALIZATION, INC.: SpeedTree. http://www.speedtree.com/, Visited on July 2013. 4, 11

[IOI06a] Ijiri T., Owada S., Igarashi T.: Seamless integration of initial sketching and subsequent detail editing in flower modeling. *Computer Graphics Forum 25*, 3 (2006), 617–624. 5, 13

[IOI06b] Ijiri T., Owada S., Igarashi T.: The sketch L-System: Global control of tree modeling using free-form strokes. In *Smart Graphics* (2006), pp. 138–146. 5, 13

[KBKv09] Krištof P., Benes B., Křivanek J., Št'ava O.: Hydraulic erosion using smoothed particle hydrodynamics. *Computer Graphics Forum (Proceedings of Eurographics 2009) 28*, 2 (mar 2009). 3, 13

[KE81] Koning H., Eizenberg J.: The language of the prairie: Frank lloyd wright's prairie houses. *Environment and Planning B: Planning and Design 8*, 3 (1981), 295–323. 9, 13

[KK11] Krecklau L., Kobbelt L.: Procedural modeling of interconnected structures. *Comput. Graph. Forum 30*, 2 (2011), 335–344. 15

[KK12a] Krecklau L., Kobbelt L.: Interactive Modeling by Procedural High-Level Primitives. *Computers and Graphics 36*, 5 (2012), 376–386. 14

[KK12b] Krecklau L., Kobbelt L.: Smi 2012: Full interactive modeling by procedural high-level primitives. *Comput. Graph. 36*, 5 (Aug. 2012), 376–386. 15

[KM06] Kelly G., McCabe H.: A survey of procedural techniques for city generation. *Institute of Technology Blanchardstown Journal 14* (2006), 87–130. 6

[KM07] Kelly G., McCabe H.: Citygen: An interactive system for procedural city generation. In *Proceedings of GDTW 2007: The 5th Annual International Conference in Computer Game Design and Technology* (Liverpool, UK, November 2007), pp. 8–16. 6, 7, 8, 13

[KMN88] Kelley A. D., Malin M. C., Nielson G. M.: Terrain simulation using a model of stream erosion. In *SIGGRAPH '88: Proceedings of the 15th Annual Conference on Computer Graphics and Interactive Techniques* (New York, NY, USA, 1988), ACM, pp. 263–268. 5, 6, 13

[KU07] Kamal K. R., Uddin Y. S.: Parametrically controlled terrain generation. In *GRAPHITE '07: Proceedings of the 5th International Conference on Computer Graphics and Interactive Techniques in Australia and Southeast Asia* (New York, NY, USA, 2007), ACM, pp. 17–23. 4, 13

[Kwo03] Kwon D. Y.: *ArchiDNA: A Generative System for Shape Configuratons*. Master's thesis, University of Washington, 2003. 9, 13

[LD99] Lintermann B., Deussen O.: Interactive Modeling of Plants. *IEEE Computer Graphics and Applications 19*, 1 (1999), 56–65. 4, 11, 13

[LG06] Larive M., Gaildrat V.: 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* (New York, NY, USA, 2006), ACM, pp. 429–437. 13

[Lin68] Lindenmayer A.: Mathematical models for cellular interaction in development. *Journal of Theoretical Biology Parts I and II*, 18 (1968), 280–315. 4, 13

[LLC*10] Lagae A., Lefebvre S., Cook R., DeRose T., Drettakis G., Ebert D., Lewis J., Perlin K., Zwicker M.: State of the art in procedural noise functions. In *EG 2010 - State of the Art Reports* (May 2010), Hauser H., Reinhard E., (Eds.), Eurographics, Eurographics Association. 2

[LRBP12a] Longay S., Runions A., Boudon F., Prusinkiewicz P.: Treesketch: Interactive modeling of trees on a tablet. In *Proceedings of the Eurographics Symposium on Sketch-Based Interfaces and Modeling* (2012), Eurographics, pp. 107–120. 13

[LRBP12b] Longay S., Runions A., Boudon F., Prusinkiewicz P.: Treesketch: Interactive procedural modeling of trees on a tablet. In *Proceedings of the International Symposium on Sketch-Based Interfaces and Modeling* (Aire-la-Ville, Switzerland, 2012), Eurographics Association, pp. 107–120. 5

[LRW*06] Lechner T., Ren P., Watson B., Brozefski C., Wilenski U.: Procedural modeling of urban land use. In *SIGGRAPH '06: ACM SIGGRAPH 2006 Research posters* (New York, NY, USA, 2006), ACM, p. 135. 6, 7, 13

[LSWW11] Lipp M., Scherzer D., Wonka P., Wimmer M.: Interactive modeling of city layouts using layers of procedural content. In *Computer Graphics Forum: Eurographics 2011* (Llandudno, UK, April 2011), vol. 30, Eurographics Association, pp. 345 – 354. 7, 13, 14, 15

[LWW03] Lechner T., Watson B., Wilensky U.: Procedural city modeling. In 1st *Midwestern Graphics Conference* (St. Louis, MO, USA, 2003). 6, 7, 13

[LWW08] Lipp M., Wonka P., Wimmer M.: Interactive visual editing of grammars for procedural architecture. In *SIGGRAPH '08: Proceedings of the 35th Annual Conference on Computer Graphics and Interactive Techniques* (New York, NY, USA, 2008), ACM, pp. 1–10. 9, 13, 14

[LYO*10] Livny Y., Yan F., Olson M., Chen B., Zhang H., El-Sana J.: Automatic reconstruction of tree skeletal structures from point clouds. *ACM Transactions on Graphics (Proceedings of SIGGRAPH ASIA 2010) 29*, 6 (December 2010), 151:1–151:8. 4, 13

[LZG10] Li Z., Zhu Q., Gold C.: *Digital terrain modeling: principles and methodology*. CRC press, 2010. 3

[Man82] Mandelbrot B. B.: *The Fractal Geometry of Nature*. W. H. Freeman, 1982. 2

[Mar06] Martin J.: Procedural house generation: a method for dynamically generating floor plans. I3D '06: Poster Proceedings of the 2006 SIGGRAPH Symposium on Interactive 3D Graphics and Games, 2006. 10, 13

[Mil86] Miller G. S. P.: The definition and rendering of terrain maps. In *SIGGRAPH '86: Proceedings of the 13th Annual Conference on Computer Graphics and Interactive Techniques* (New York, NY, USA, 1986), vol. 20, ACM, pp. 39–48. 2, 13

[MKM89] Musgrave F. K., Kolb C. E., Mace R. S.: The synthesis and rendering of eroded fractal terrains. In *SIGGRAPH '89: Proceedings of the 16th Annual Conference on Computer Graphics and Interactive Techniques* (New York, NY, USA, 1989), ACM, pp. 41–50. 3, 12, 13

[MM10] Marson F., Musse S. R.: Automatic generation of floor plans based on squarified treemaps algorithm. *International Journal of Computer Games Technology 2010* (January 2010), 1–10. 10, 13

[MP96] Měch R., Prusinkiewicz P.: Visual models of plants interacting with their environment. In *SIGGRAPH '96: Proceedings of the 23rd annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 1996), ACM, pp. 397–410. 4, 13

[MS09] McCrae J., Singh K.: Sketch-based path design. In *GI '09: Proceedings of Graphics Interface 2009* (Toronto, Ontario, Canada, 2009), Canadian Information Processing Society, pp. 95–102. 6, 13

[MSK10] MERRELL P., SCHKUFZA E., KOLTUN V.: Computer-generated residential building layouts. *ACM Transactions on Graphics 29*, 5 (2010), 181:1–181:12. 10, 13

[MSL*11] MERRELL P., SCHKUFZA E., LI Z., AGRAWALA M., KOLTUN V.: Interactive furniture layout using interior design guidelines. In *SIGGRAPH '11: Proceedings of the 38th Annual Conference on Computer Graphics and Interactive Techniques* (New York, NY, USA, 2011), ACM, pp. 87:1–87:10. 10, 11, 13

[Mus93] MUSGRAVE F. K.: *Methods for Realistic Landscape Imaging*. PhD thesis, Yale University, New Haven, CT, USA, 1993. 13

[MWH*06] MÜLLER P., WONKA P., HAEGLER S., ULMER A., GOOL L. V.: Procedural modeling of buildings. In *SIGGRAPH '06: Proceedings of the 33rd Annual Conference on Computer Graphics and Interactive Techniques* (New York, NY, USA, 2006), ACM, pp. 614–623. 9, 12, 13

[MZWG07] MÜLLER P., ZENG G., WONKA P., GOOL L. V.: Image-based procedural modeling of facades. In *SIGGRAPH '07: Proceedings of the 34th Annual Conference on Computer Graphics and Interactive Techniques* (New York, NY, USA, 2007), vol. 26, ACM, pp. 85:1–85:10. 8, 9

[Pat12] PATOW G.: User-friendly graph editing for procedural modeling of buildings. *IEEE Computer Graphics and Applications 32* (2012), 66–75. 9, 13, 14

[Per85a] PERLIN K.: An Image Synthesizer. In *SIGGRAPH '85: Proceedings of the 12th Annual Conference on Computer Graphics and Interactive Techniques* (New York, NY, USA, 1985), vol. 19, ACM, pp. 287–296. 3, 13

[Per85b] PERLIN K.: An image synthesizer. *SIGGRAPH Comput. Graph. 19*, 3 (July 1985), 287–296. 2

[PGGM09] PEYTAVIE A., GALIN E., GROSJEAN J., MERILLOU S.: Arches: a framework for modeling complex terrains. In *Computer Graphics Forum: Proceedings of Eurographics 2009* (2009), Eurographics Association, pp. 457–467. 2, 3, 13

[PH93] PRUSINKIEWICZ P., HAMMEL M.: A fractal model of mountains with rivers. In *Proceedings of Graphics Interface '93* (May 1993), pp. 174–180. 5, 13

[PHL*09] PALUBICKI W., HOREL K., LONGAY S., RUNIONS A., LANE B., MĚCH R., PRUSINKIEWICZ P.: Self-organizing tree models for image synthesis. *ACM Trans. Graph. 28*, 3 (2009), 1–10. 4, 5, 13

[PHM93] PRUSINKIEWICZ P., HAMMEL M. S., MJOLSNESS E.: Animation of plant development. In *SIGGRAPH '93: Proceedings of the 20th annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 1993), ACM Press, pp. 351–360. 4, 13

[PJS92] PEITGEN H., JÜRGENS H., SAUPE D.: *Chaos and fractals: new frontiers of science*. Springer-Verlag, 1992. 2

[Pla13] PLANETSIDE: Terragen 2. Available from http://www.planetside.co.uk, 2013. 11

[PLH88] PRUSINKIEWICZ P., LINDENMAYER A., HANAN J.: Development models of herbaceous plants for computer imagery purposes. In *SIGGRAPH '88: Proceedings of the 15th annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 1988), ACM Press, pp. 141–150. 4, 13

[PM01] PARISH Y. I. H., MÜLLER P.: Procedural Modeling of Cities. In *SIGGRAPH '01: Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques* (New York, NY, USA, 2001), ACM, pp. 301–308. 6, 7, 8, 13

[Pro13] PROCEDURAL: CityEngine. Available from http://www.esri.com/software/cityengine/, 2013. 12

[Pru86] PRUSINKIEWICZ P.: Graphical applications of l-systems. In *Proceedings on Graphics Interface '86/Vision Interface '86* (1986), pp. 247–253. 4, 13

[Pru97] PRUSINKIEWICZ P.: A look to visual modeling of plants. In *German Conference on Bioinformatics* (1997), Springer Computer Science, Springer–Verlag Wien New York. 4, 13

[Pru00] PRUSINKIEWICZ P.: Simulation modeling of plants and plant ecosystems. *Commun. ACM 43*, 7 (2000), 84–93. 4, 13

[PSK*12] PIRK S., STAVA O., KRATT J., SAID M. A. M., NEUBERT B., MĚCH R., BENES B., DEUSSEN O.: Plastic trees: interactive self-adapting botanical tree models. *ACM Transactions on Graphics (Proceedings of SIGGRAPH 2012) 31*, 4 (July 2012), 50:1–50:10. 5, 13

[RCMLS96] RAU-CHAPLIN A., MACKAY-LYONS B., SPIERENBURG P. F.: The lahave house project: Towards an automated architectural design service. In *CADEX '96: Proceedings of the International Conference on Computer-Aided Design* (Hagenberg, Austria, September 1996). 9, 13

[RFL*05] RUNIONS A., FUHRER M., LANE B., FEDERL P., ROLLAND-LAGAN A.-G., PRUSINKIEWICZ P.: Modeling and visualization of leaf venation patterns. *ACM Transactions on Graphics (Proceedings of SIGGRAPH 2005) 24*, 3 (2005), 702–711. 4, 13

[RLP07] RUNIONS A., LANE B., PRUSINKIEWICZ P.: Modeling trees with a space colonization algorithm. *Eurographics Workshop on Natural Phenomena* (2007), 63–70. 4, 13

[Ros13] ROSENBERG J.: Geocontrol 2. Available from http://www.geocontrol2.com, 2013. 11

[Sau06] SAUNDERS R. L.: *Terrainosaurus: Realistic Terrain Synthesis Using Genetic Algorithms*. Master's thesis, Texas A&M University, December 2006. 3, 13

[ŠBBK08] ŠŤAVA O., BENES B., BRISBIN M., KŘIVÁNEK J.: Interactive terrain modeling using hydraulic erosion. In *Eurographics / SIGGRAPH Symposium on Computer Animation* (Dublin, Ireland, 2008), Eurographics Association, pp. 201–210. 13

[ŠBM*10] ŠŤAVA O., BENES B., MĚCH R., ALIAGA D. G., KRIŠTOF P.: Inverse Procedural Modeling by Automatic Generation of L-systems. In *Computer Graphics Forum: Proceedings of Eurographics 2010* (2010), vol. 29, Eurographics Association, pp. 665–674. 14

[SG71] STINY G., GIPS J.: Shape grammars and the generative specification of painting and sculpture. In *Proceedings of the Workshop on Generalisation and Multiple Representation* (1971). 9

[Sid13] SIDE EFFECTS SOFTWARE: Houdini. Available from http://www.sidefx.com/, 2013. 12

[SMBC13] SILVA P., MUELLER P., BIDARRA R., COELHO A.: Node-based shape grammar representation and editing. In *PCG '13: Proceedings of the 2013 Workshop on Procedural Content Generation in Games* (Chania, Crete, Greece, 2013), ACM. 9, 14

[Smi84] SMITH A. R.: Plants, fractals, and formal languages. In *SIGGRAPH '84: Proceedings of the 11th annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 1984), ACM Press, pp. 1–10. 1

[SS05] STACHNIAK S., STÜRZLINGER W.: An algorithm for automated fractal terrain deformation. *Computer Graphics and Artificial Intelligence 1* (May 2005), 64–76. 3, 13

[STdB10a] SMELIK R. M., TUTENEL T., DE KRAKER K. J., BIDARRA R.: Integrating procedural generation and manual editing of virtual worlds. In *PCG '10: Proceedings of the 2010 Workshop on Procedural Content Generation in Games* (New York, NY, USA, 2010), ACM, pp. 1–8. 14

[STdB10b] Smelik R. M., Tutenel T., de Kraker K. J., Bidarra R.: Interactive creation of virtual worlds using procedural sketching. In *Proceedings of Eurographics 2010: Short Papers* (May 2010), Eurographics Association. 4, 13

[STdB11] Smelik R. M., Tutenel T., de Kraker K. J., Bidarra R.: A declarative approach to procedural modeling of virtual worlds. *Computers & Graphics 35*, 2 (April 2011), 352–363. 5, 15

[SYBG02] Sun J., Yu X., Baciu G., Green M.: Template-based generation of road networks for virtual city modeling. In *VRST '02: Proceedings of the ACM Symposium on Virtual Reality Software and Technology* (New York, NY, USA, 2002), ACM, pp. 33–40. 6, 7, 13

[TBSd09] Tutenel T., Bidarra R., Smelik R. M., de Kraker K. J.: Rule-based layout solving and its application to procedural interior generation. In *3AMIGAS: Proceedings of the CASA 2009 Workshop on 3D Advanced Media in Gaming and Simulation* (Amsterdam, The Netherlands, June 2009), pp. 15–24. 10, 13

[TBSd10] Tutenel T., Bidarra R., Smelik R. M., de Kraker K. J.: A semantic scene description language for procedural layout solving problems. In *AIIDE '10: Proceedings of the 6th Conference on Artificial Intelligence and Interactive Digital Entertainment* (Stanford, CA, USA, October 2010). 10

[TLL*11] Talton J. O., Lou Y., Lesser S., Duke J., Měch R., Koltun V.: Metropolis procedural modeling. *ACM Trans. Graph. 30* (April 2011), 11:1–11:14. 14

[TSL*11] Tutenel T., Smelik R. M., Lopes R., de Kraker K. J., Bidarra R.: Generating consistent buildings: A semantic approach for integrating procedural techniques. *IEEE Transactions on Computational Intelligence and AI in Games 3*, 3 (2011), 274–288. 15

[VABW09] Vanegas C. A., Aliaga D. G., Benes B., Waddell P. A.: Interactive design of urban spaces using geometrical and behavioral modeling. *ACM Transactions on Graphics: Proceedings of ACM SIGGRAPH Asia 2009 28*, 5 (December 2009), 1–10. 7, 13

[VBHv11] Vanek J., Benes B., Herout A., Šťava O.: Large-scale physics-based terrain editing using adaptive tiles on the gpu. *IEEE Computer Graphics and Applications* (2011). 3, 13

[VGDA*12] Vanegas C. A., Garcia-Dorado I., Aliaga D. G., Benes B., Waddell P.: Inverse design of urban procedural models. *ACM Transactions on Graphics 31* (2012), 168:1–168:11. 13, 14

[VKW*12] Vanegas C. A., Kelly T., Weber B., Halatsch J., Aliaga D., Müller P.: Procedural generation of parcels in urban modeling. In *Computer Graphics Forum: Proceedings of Eurographics 2012* (2012), Eurographics Association. 8

[Vor08] Voronoi G. F.: Nouvelles applications des paramètres continus à la théorie des formes quadratiques. *Journal für die Reine und Angewandte Mathematik 134* (1908), 198–287. 7

[WCMT07] Wojtan C., Carlson M., Mucha P. J., Turk G.: Animating corrosion and erosion. In *Eurographics Workshop on Natural Phenomena* (2007). 3, 13

[WMV*08] Watson B., Müller P., Veryovka O., Fuller A., Wonka P., Sexton C.: Procedural urban modeling in practice. *IEEE Computer Graphics and Applications 28* (2008), 18–26. 6

[WMWG09] Weber B., Müller P., Wonka P., Gross M.: Interactive Geometric Simulation of 4D Cities. *Computer Graphics Forum: Proceedings of Eurographics 2009 28* (April 2009), 481–492. 7, 13

[WP95] Weber J., Penn J.: Creation and rendering of realistic trees. In *Proceedings of SIGGRAPH '95* (1995), pp. 119–128. 4

[WWSR03] Wonka P., Wimmer M., Sillion F., Ribarsky W.: Instant architecture. In *SIGGRAPH '03: Proceedings of the 30th Annual Conference on Computer Graphics and Interactive Techniques* (New York, NY, USA, 2003), ACM, pp. 669–677. 9, 13

[XFT*08] Xiao J., Fang T., Tan P., Zhao P., Ofek E., Quan L.: Image-based facade modeling. *ACM Trans. Graph. 27*, 5 (Dec. 2008), 161:1–161:10. 8

[YCZY04] Yong L., Congfu X., Zhigeng P., Yunhe P.: 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* (New York, NY, USA, 2004), ACM, pp. 412–418. 9, 13

[YK12] Yoon D., Kim K.-J.: 3D game model and texture generation using interactive genetic algorithm. In *Proceedings of the Workshop at SIGGRAPH Asia* (2012), ACM, pp. 53–58. 13

[YYT*11] Yu L.-F., Yeung S. K., Tang C.-K., Terzopoulos D., Chan T. F., Osher S.: Make it home: Automatic optimization of furniture arrangement. In *SIGGRAPH '11: Proceedings of the 38th Annual Conference on Computer Graphics and Interactive Techniques* (New York, NY, USA, 2011), ACM, pp. 86:1–86:12. 10, 13

[ZSTR07] Zhou H., Sun J., Turk G., Rehg J. M.: Terrain synthesis from digital elevation models. *IEEE Transactions on Visualization and Computer Graphics 13*, 4 (July-Aug. 2007), 834–848. 3

[ZXJ*13] Zhang H., Xu K., Jiang W., Lin J., Cohen-Or D., Chen B.: Layered analysis of irregular facades via symmetry maximization. *ACM Trans. Graph. 32*, 4 (July 2013), 121:1–121:13. 8