Online Appendix to: Procedural Content Generation for Games: A Survey

MARK HENDRIKX, SEBASTIAAN MEIJER, JOERI VAN DER VELDEN, and ALEXANDRU IOSUP,
Delft University of Technology, The Netherlands

A. MORE DETAILS ON THE TAXONOMY OF METHODS FOR PROCEDURAL CONTENT GENERATION

We discuss in this appendix the classes of methods introduced in Section 3 of the article.

A.1 Pseudo-Random Number Generators (PRNG)

Nature often gives the illusion of randomness, for example in the shape of a mountain, a cloud, or a flower. Pseudo-Random Number Generators (PRNGs) can, therefore, be used for mimicking the randomness found in nature.

Perlin noise [Perlin 1985; 1990] is a PRNG-based noise generator with wide use in media and entertainment. This noise generates maps of data points (random values). The map data points are generated by a seeded PRNG through interpolation. More detail can be added to the noise map by combining multiple layers of Perlin noise and by using scaling. The Perlin noise is representative for the many PRNG-based techniques used in games [Van Verth and Bishop 2008]; other popular PRNGs exist [Lecky-Thompson 2001].

A.2 Generative Grammars (GG)

Generative grammars, stemming from Noam Chomsky's study of languages in the 1960s, are sets of rules that, operating on individual words, can generate only grammatically correct sentences. They can be used to create correct objects from elements encoded as letters/words. In this section we discuss L-systems, split grammars, wall grammars, and shape grammars, which have been used for content generation in entertainment.

A.2.1 *Lindenmayer-Systems.* (L-systems) consist of a grammar consisting of symbols which describe the characteristics of an object. A string generated by the grammar describes the structure or the behavior of an object.

A.2.2 *Split Grammars.* Similarly to L-systems, split grammars [Wonka et al. 2003] work on string-encoded shapes. New shapes are generated from a basic set of shapes by applying rewriting rules governing shape-to-shape conversion, the split grammar generates a new shape. For example, using a split grammar an initial wall-shape can be divided into two smaller shapes, which in turn can be rewritten (converted) into a window frame shape and a window shape. Split grammars are context free, meaning that the rewriting process always yields the same result given a set of rewriting rules, no matter in which order the string of symbols is evaluated.

A.2.3 *Wall Grammars.* Wall grammars by Larive and Gaildrat [2006] are specifically designed for creating building exteriors. Shapes are manipulated to form a building exterior similarly to split

DOI 10.1145/2422956.2422958 http://doi.acm.org/10.1145/2422956.2422958

grammars, but wall grammars can generate more advanced shapes, for example, the wall extrusion rule can lead to complex three-dimensional shapes like balconies and fire escapes.

A.2.4 *Shape Grammars.* Shape grammars [Müller et al. 2006] are context-sensitive and sequential grammars originating from the work of Stiny in the 1970s. For each rewriting step, the symbol and its neighbors in the string determine what symbol(s) replace the original symbol. Therefore, the rewriting process of shape grammars is different from L-systems or split grammars. Similarly to wall grammars, shape grammars can generate more complex structures.

A.3 Image Filtering (IF)

Image filtering has as main goal to improve an image with regard to a (subjective) measure, or to emphasize certain characteristics of an image to display (partially) hidden information. Many techniques have been developed for image filtering; the yearly image processing tutorials at SIGGRAPH are good surveys of the state-of-the-art with application in multimedia-related fields. In this section we present two fundamental image processing techniques, binary morphology and convolution filters.

A.3.1 *Binary Morphology.* Binary morphology is a set of techniques used for binary operations on images. The binary image often required by binary operations can be obtained through thresholding, a process where pixels below a certain intensity are set to zero and the rest to one, effectively creating a binary image. Typical examples of binary morphology operations include dilation, in which pixels are added to the edge of an element in an image, and erosion, which does the opposite. By combining basic binary operators, more useful complex operations can be achieved. For example, by first dilating an image and then subtracting the original from the result, the final result depicts the edges of each element in the original image.

A.3.2 *Convolution Filters.* Convolution filters are filters which can be represented by a simple image, function, or discrete dataset. Convolution is a mathematical operator on two signals, where one signal is used to modify the other thereby creating a new signal. These type of filters can be used, for example, to remove noise, smooth, sharpen, detect edges of objects, or even detect the movement direction of objects in an image. Using convolution filters, a simple texture can be manipulated to create a whole new texture, thereby saving storage space.

A.4 Spatial Algorithms (SA)

Spatial algorithms manipulate space to generate game content. The output is created by using an input with structure, for example a grid, or self-recurrence. In this section we discuss tiling and layering, grid subdivision, fractals, and Voronoi diagrams.

A.4.1 *Tiling and Layering.* Tiling is a technique used to create a game space by decomposing a map into a grid. The grid is not limited to a rectangle-size; hexagonal shapes are also common. Grids are 2D data structures, but isometric projections can be coupled to grids to create the illusion of a 3D map.

Layering is a technique that integrates several grids, called layers, into the same map. A tile is then constructed by overlapping parts from each layer, some of which may contain transparent parts. This approach enables the creation of overlay effects, such as running water, and of 3D-looking game space by using only a limited amount of source terrain textures.

A.4.2 *Grid Subdivision.* Grid subdivision is an iterative and dynamic technique for object generation. An object is first divided into a uniform grid with the appropriate textures. A grid subdivision algorithm, for example, the Patch-LOD algorithm [Pl et al. 2006], is used to iteratively add detail to

the object. The dynamic part of this technique links the iterative generation to the point of view; only the grid cells that are closer to the current point of detail must be subdivided into smaller cells to create the required level of detail. An example using this technique is the rendering of a procedurally generated terrain: only the cells near the player are detailed (generated), while the rest of the terrain is more coarse, thereby saving computational resources.

A.4.3 *Fractals.* Fractals are recursive figures which consist of copies of themselves, for example, a Koch snowflake. With fractals, a few parameters control a wide range of possible results. An advantage of fractals is that objects with seemingly infinite detail can be stored as a simple recursive function. The generation of fractals is a resource-intensive process, due to recursiveness. Using iterated function systems, the resulting image can be generated faster, using an iterative rather than a recursive process.

A.4.4 *Voronoi Diagrams.* Voronoi diagrams [Aurenhammer 1991] are decompositions of metric spaces into parts whose size and shape is determined by the position of seed points (points of interest) in the metric space. The decomposition establishes borders of points equally distant from the closest seed points, and territory containing exactly one seed point and all the locations for which the territory's seed point is the nearest point of interest. For a small map with a few points, a diagram can easily be calculated using an iterative approach. However, when the collections of points increase in size, an improved computational approach is required. A variety of algorithms have already been proposed to make Voronoi diagrams usable in near-real time [de Berg et al. 2008, Chapter 7].

A.5 Modeling and Simulation of Complex Systems (CS)

It is impractical in some cases to describe natural phenomena with mathematical equations. Models and simulations can be used to overcome this problem. In this section we describe cellular automata, tensor fields, and agent-based simulation.

A.5.1 *Cellular Automata.* Cellular automata [Chopard and Droz 1998] are a discrete computational model based on cells aligned in a grid, where each cell has a state and is subject to a common set of rules. The computational model is applied at discrete time steps. The rules of the board determine how cell neighborhood and state influence the next state of the cell. The resulting behavior can be random, but also periodic.

Cellular automata can be combined with other systems to form new computational models. An example of a combined system is a open L-system, a combination of cellular automata and L-systems. In open L-systems, the behavior of the objects is largely determined by interaction with the environment the object is in and by interaction with other objects in that environment [Hidalgo et al. 2008]. This allows for the modeling of spatial constraints, such as the minimum distance between two objects.

A.5.2 *Tensor Fields.* Tensor fields [Chen et al. 2008, Section 4] are two-dimensional generalizations of vectors, which can be used to specify the shape of a game space. The tensors describe the direction of the elevation of the map. Because tensor lines can be visualized, they are suitable for interactive design and manipulation of road networks.

A.5.3 *Agent-Based Simulation.* (ABS) [Davidsson 2001] is based on modeling a complex situation using individuals, called agents. Emergent behavior (complex behavior that arises out of relatively simple agent interactions) is a feature of ABS that contrasts with the average behavior observable through traditional modeling techniques. Agents can be added, removed, or replaced during the simulation; agents may also learn in time.

A.5.4 *Other Complex Systems and Theories.* Many other complex systems and theories have and may still make their way in procedural content generation for games, including theories of the dramatic act (for story generation), of the cognitive process (for entity behavior), etc.

A.6 Artificial Intelligence (AI)

Artificial Intelligence is a large field in computer science that tries to mimic animal or human intelligence. Examples include speech recognition, planning, and execution of physical tasks by robots.

A.6.1 *Genetic Algorithms.* Genetic algorithms [Goldberg 1989] are used to solve optimization problems by mimicking biological evolution. Possible solutions are coded as a strings (chromosomes), and a fitness function is used to evaluate the quality of a solution. A mutation and crossover function are applied to create new solutions. The mutation function converts a solution in a new one. The crossover function specifies the exchange of chromosome parts between a set of parent chromosomes. The mutation rate and crossover rate determine how frequently these operations occur.

For optimization problems, a pool of N initial candidate solutions is first generated. Next, each solution is rated by using the fitness function, after which new N solutions are created by applying the mutation and crossover functions on randomly selected sets of parents based on fitness. The process continues until a satisfactory solution is found or until a predefined round count.

A.6.2 *Artificial Neural Networks.* Artificial neural networks [Haykin 1994] are computational models with the ability to learn the relationship between an input and output by minimizing the error between the output and expected output. ANN's can be used for finding patterns, and classifying, remembering, and structuring data. An ANN consists of computational units called neurons, which are connected by weighted edges. A single neuron can have several incoming and outgoing edges. When a neuron receives an input, it first combines the inputs of all incoming edges and tests if it is triggered by this input. If the neuron is triggered, it sends the combined signal over the output lines. The ANN functions in an environment, which provides the input signals, processes the output signals, and calculates the error which the ANN can use to adjust the weight on the edges and thereby learn.

A.6.3 *Constraint Satisfaction and Planning.* Constraint satisfaction and planning [Russell et al. 1995] entails finding a path from an initial state to an end state by applying actions. A planning problem consists of an initial state, actions, and a goal test. Planning Domain Definition Language (PDDL) is commonly used to express planning problems. An action can be executed when the initial condition is satisfied. The effect of an action can be the addition or deletion of variables resulting in a new state. Forward state-space search algorithms start planning from the initial state. In contrast, backward state-space search algorithms start in the final state. Despite both types of algorithms, planning is NP-hard in general, which explains the importance of heuristics in planning.