# A survey of procedural content generation techniques suitable to game development

Daniel Michelon De Carli*, Fernando Bevilacqua†, Cesar Tadeu Pozzer‡ and Marcos Cordeiro d'Ornellas‡
*NTIC-Universidade Federal do Pampa
Alegrete, Brazil
Email: danielcarli@unipampa.edu.br
†Universidade Federal da Fronteira Sul
Chapecó, Brazil
Email:fernando.bevilacqua@uffs.edu.br
‡Universidade Federal de Santa Maria
Santa Maria, Brazil
Email: {ornellas, pozzer}@inf.ufsm.br

***Abstract*—The development of a complex game is a time consuming task that requires a significant amount of content generation, including terrains, objects, characters, etc that requires a lot of effort from the a designing team. The quality of such content impacts the project costs and budget. One of the biggest challenges concerning the content is how to improve its details and at the same time lower the creation costs. In this context procedural content generation techniques can help to reduce the costs associated with content creation. This paper presents a survey of classical and modern techniques focused on procedural content generation suitable for game development. They can be used to produce terrains, coastlines, rivers, roads and cities. All techniques are classified as assisted (require human intervention/guidance in order to produce results) or non-assisted (require few or no human intervention/guidance to produce the desired results).**

***Keywords*-Survey; Procedural Generation; Game Development; 3D**

## I. Introduction

One of the main components of modern games production is the content design. The outcome of that effort are the objects and the environments the player will interact with, for instance. The quality of such content generation directly impacts the cost of the project and the final content often requires a team composed of several artists and 3D modelers in order to be produced. The environment generation, for instance, demands the creation of the main area (where player will spend most of them time) and the surroundings arenas (places that may not be visited by the player). If the surrounding areas are not so important the creation process of such content can be tedious and a waste of time for the team. The time invested on such peripheral areas could be better used if applied to the main area instead.

In this context procedural techniques have been emerging as a potential solution for content creation. Even though the procedural content generation can produce a complete and polished environment that can be used with no modifications, it can also be used as a starting point for the designing team. The artist or the game designer can enhance the procedurally generated content in order to make it suitable for the game context, avoiding the tedious task of creating surrounding areas from scratch, for instance. The evolution of procedural generation techniques may allow the creation of more complex content such as whole scenes, saving working hours and decreasing the project cost.

Figure 1. Procedurally generated content [1], [2] and [3].

The study focused on procedural content generation has been evolving and resulting in several techniques that can

be applied to game development. The purpose of such techniques covers the generation of several types of contents such as continents, terrains, rivers, roads, cities and even complete worlds. The knowledge of such techniques and their results can help the creation of new ones and also facilitate the evaluation of what procedural content generation method suits better for different game development scenarios.

This paper presents a survey of procedural content generation techniques that can be used in game development. All techniques are grouped according to the their resulting content type (e.g. terrains, rivers, etc) and after categorized as *assisted* (heavily relies on human intervention/guidance in order to produce the desired results) or *non-assisted* (requires a few or no human intervention/guidance in order to produce the desired results). Every technique is described and followed by an image that illustrates the visual result of such technique; information regarding processing time is provided when available in the reviewed paper, so it is possible to identify techniques that are suitable for a real-time approach, for instance.

In the scope of this paper a technique is classified as non-assisted when the method produces a satisfying result with no human intervention and/or guidance. If the technique must be provided with a few adjustments/parameters such as the number of iterations, a height limit, etc and there is no need to readjust those elements during the iteration process, the technique is also classified as non-assisted. If the method requires a significant amount of time for adjustment and/or parametrization or if human decisions or interventions must be performed during most of the process in order to generate a satisfying result, then the technique is classified as assisted.

The remainder of this paper is organized as follows: section II presents techniques related to terrain and continents generation; section III presents techniques related to roads and rivers; section IV shows techniques related to cities and urban spaces; finally section V presents a conclusion and some thoughts about the applicability and evolution of procedurally content generation techniques related to game development.

## II. Terrains and Continents

The terrain is a fundamental part of the content of several games and it plays an important role in the replayability process. An interesting terrain will keep the player motivated to explore new places and spend more time playing. The procedural generation of terrains and continents has proved to be feasible and can be seen in several prior works (e.g. [4], [5], [6]).

Some of the techniques used in procedural generation of terrains and continents include noise [7], L-systems [8] and fractals [9]. Those techniques can be used alone, combined among each other [10], [11], [12], [13], [14] or combined with other techniques such as erosion simulation e.g [15], [16], [17], [18], [19], [20], [21]. The procedural generation can also be parametrically controlled in order to avoid completely random content. The parametrization can be achieved in different ways such as adjusting division limits [22] or controlling distortion using splines [23]. The terrain can also be generated interactively though high level tools using volumetric discrete data structure in order to make the representation of overhangs, caves and arches easier [24]. Figures 2 and 3 show some of the previously mentioned techniques.

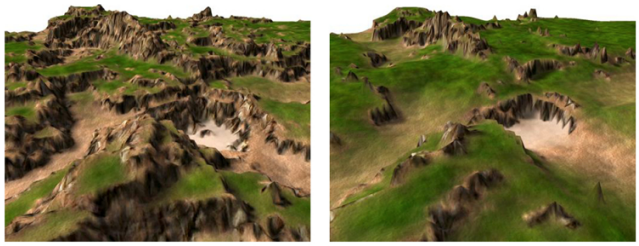

Figure 2. Rendered height maps affected by erosion simulation [16].

Figure 3. A canyon with rocks detached from the cliffs [24].

An assisted method that uses a compact vector-based model can be used to efficiently and accurately control the terrain generation process [1]. As pointed by the authors when a constraint is used the vicinity of that constraint can lack in control [25], [26] or depends on the characteristics of another terrain image [2]. In order to avoid that problem the method uses control curves with the corresponding elevation, gradient and noise constraint parameters attached to them, which generates a set of maps that are further combined in order to produce the final terrain. Those curves are based on Bezier and diffusion curves [27] and they are created and adjusted by the operator responsible for the terrain creation. The curves parameters must be tweaked by the operator according to the terrain characteristics so the method can produce the desired outcome. This method allows the creation of terrains with fine control over the content and can incrementally add/remove as many details as needed by adjusting the control curves. Even though the use of diffusion curves demands a significant amount of time

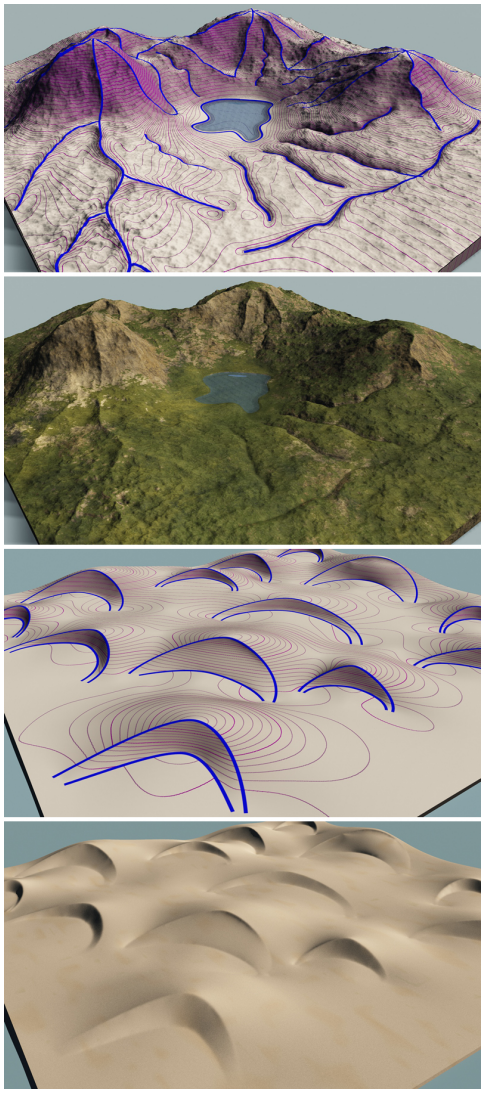

Figure 4. Different kinds of landscapes (lake and desert) created with 45 and 26 control curves respectively [1].

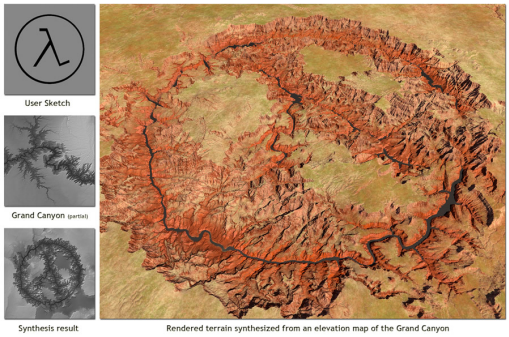


Figure 5. Grand Canyon terrain syntetization. On the left: the user sketch, the Grand Canyon elevation model and the resulting height map [2].

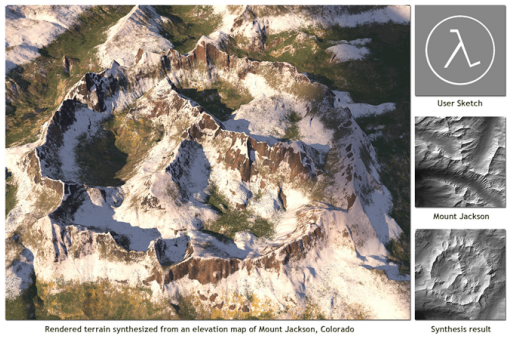


Figure 6. Mount Jackson (Colorado) terrain syntetization. On the right: the user sketch, the Mount Jackson elevation model and the resulting height map [2].

dedicated to parameters adjustment, causing the operator to spend more time to achieve any results, this approach is able to produce much more predictable and controllable results compared to a pure fractal or non-assisted method. As exemplified by the authors the operator took almost 45 minutes to carefully edit all the details needed to produce a complex terrain. Figure 4 illustrates the use of control curves and their results.

Other assisted approach to terrain generation guided by constraints is the synthesis from digital elevation models [2]. In that technique the resulting terrain is automatically generated based on the visual style of a real terrain data (e.g. a model provided by a government geological agency) and meets the feature constraints of the sketch made by the user. The algorithm breaks the sketch map into small patch regions and searches through the real terrain height field for structural feature matches. The extraction of features (valleys, ridges, hills, etc) from the real terrain height field is achieved using an adapted Profile recognition and Polygon breaking Algorithm (PPA) [28]. After those features are extracted (in form of patches), they are combined into the resulting terrain according to the matches against the user sketch. A procedure is applied to smooth the transition between the different patches. Figures 5 and 6 illustrate the technique applied to the Grand Canyon and Mount Jackson elevation models using a user sketch based on the Half-Life game logo.

Another assisted technique used to terrain and continent generation is based on interactive genetic algorithms [29]. The author states that some genetic algorithms used in terrain generation techniques overwhelm the user with parameters and adjustments that may not be easy to master and often rely on deep knowledge about the tool or are required to use real world data as input, such as Digital Terrain Elevation Data (DTED) [30], [31], [32]. The idea of the author is to make the user not aware of parameters and configurations as numbers but as images instead. The terrain generation is achieved by an interactive genetic

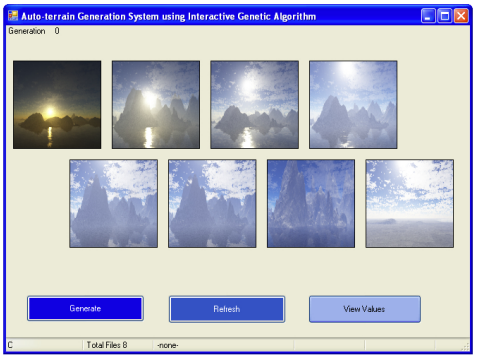


Figure 7. Initial population of terrains available for user selection [29].

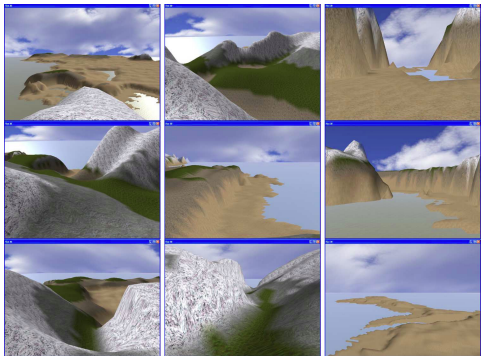

Figure 8. Several scenes produced with different agents configuration [33].

algorithm that can be defined as the process of selecting and combining members of a population in order to produce a new population (descendants). In the context of this paper, the population members are complete scenes featuring water, terrain, illumination and clouds that are presented to the user as a rendered image. Each of those elements (water, terrain, etc) is internally represented as an 8 bit chromosome, so every population member has a set of chromosomes that defines how the scene it represents is rendered. Starting with a randomly generated population of 8 members (complete scenes), the user selects 3 of them that best describe the desired resulting scene (e.g. if a scene featuring lower mountains is desired, the user will choose members of the population that feature such characteristic). After that a new population is generated based on the combination of genes (crossover) among the members the user selected and the rest of the population. Since the members the user selected have high priority to combine and transmit theirs genes to further descendants the next generation of members are more likely to have theirs characteristics, which are the features the user is looking for. In order to increase the variability of content each member may suffer a mutation operation which changes the value of a random bit of any of its features. The selection and combination process are repeated by the user until a member of the population with the desired features is found. Figure 7 shows the initial randomly generated population of 8 members available for user selection.

Evolving the concept of user input there is a non-assisted approach that makes a controlled procedural terrain generation using software agents [33]. Using a set of six different agent types (coastline, smoothing, beach, mountain, hill and river agent) parametrized according to designer-defined constraints, the agent walks in the map generating content. Depending on the number of agent and their constraints/types the user can generate different maps. The agent interacts with the environment according to its type so a coastline agent will produce land and coastlines, a mountain agent will rise the height of the points it visits, a beach agent will decrease and smooth the height of points near the coastline, etc. The agents work on the resulting map simultaneously and the action of one agent can interfere in the actions of the others. The constraints each agent carries avoid unnatural terrain patterns, such as a river climbing a mountain (river agents tend to avoid high points when they walk through the map). A terrain designer can adjust the amount of agents, their types and constraints in order to produce the desired terrain (e.g. a generation without river agents will show no river in the result, a mountain agent carrying low height attributes will produce low mountains, etc). Figure 8 shows a set of 512 x 512 heightmaps generated using different agents settings; each terrain takes about 20 seconds to be produced by the agents.

Another non-assisted technique generates pseudo-infinite terrain and continent and is mainly based on Perlin [34]. The terrain and the coastline are created based on a procedurally generated matrix that globally describes the features of every region in the terrain (continents, coastlines, etc). The matrix is generated with no human intervention or guidance. The matrix contains low detail information that servers as a seed to the rest of the algorithm that generates high detail information. Every point in the terrain is mapped to an entry in that matrix and then adjusted to meet the description found. The point receives its height value based on the matrix description and on a series of transformation and interpolations influenced by its neighbors. The content generation of every point is controlled by Perlin noise. The matrix size is much smaller than the terrain size so several points of the terrain are mapped to the same entry in the matrix. It forces the content generation of each point to rely mostly on its neighbors, using the matrix description as a clue to guide the content generation. Figure 9 shows a terrain generated by this technique.

## III. Roads and Rivers

A realistic landscapes is composed of different objects such as trees, roads and rivers. The two latters are important artifacts to break the artificial homogeneity that sometimes is created by procedural terrain generation techniques. A

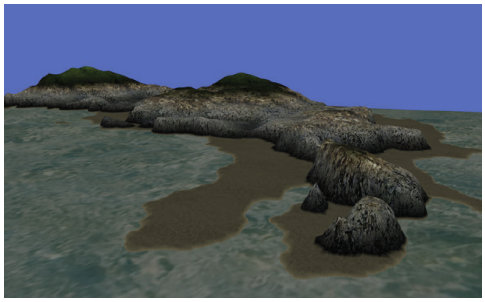

Figure 9. Terrain generated by Charack featuring coastlines and beaches [34].

Figure 10. Roads procedurally generated and influenced by the environment (river, mountains, etc) [39].

landscape featuring roads and rivers seems to be a more compelling and convincing environment to the player. The generation of such features, however, is a complex task that involves the consideration of elements such as path planning, trajectory cost calculation, constraint analysis to avoid unreal patterns, discretization of points and so on. Several procedural techniques have been proposed to generate roads such as tensor fields to guide the road graph generation [35], interactive synthesis of urban street networks [36], [37] and template patterns combined with Voronoi diagrams [38].

A non-assisted method for roads generation is based on a weighted anisotropic shortest path algorithm [39]. The method finds the path between two pre-defined points performing a computation on a continuous domain; the space existing among the origin and the end points of the road is discretized in a grid composed of several aligned points that are analyzed as a graph. The method restricts the search to paths formed by the concatenation of straight-line segments between those points, computing a path between two points that minimizes the line integral of a cost-weighting function along the road path. The line integral calculation is made by the approximation of a finite sum by discretizing the integration domain into `n` intervals. A set of parametrized cost functions is required and is used to evaluate the line integral of the cost-weighting function along the road. Those function are defined by the user and they influence the road trajectory by constraining the shortest path research. After the path is calculated the road is generated by excavating the terrain along the path and generating the road mesh as well as bridges and tunnels with the appropriate size and characteristics. Figure 10 shows two roads generated using this approach.

An assisted technique introduces the concept of procedural natural systems, an approach aimed at reducing the time needed to create natural phenomena features in game maps [40]. As pointed by the authors, a large natural phenomenon such as a river requires several work hours of a designing team in order to be realistic and visually acceptable. Using a level editing tool, for instance, the operator must excavate the river course, apply the right textures to the river/riverbed and add complementing objects to the area (trees, bushes, etc). If the course of the river must be changed, the whole process have to be repeated and often none of the already created content can be reused. The approach of a procedural natural system relies on the idea that natural phenomena consists of three parts: footprint (the appearance of the phenomenon, including height and texture definition), shape (the area in the map the phenomenon takes place) and procedure (the interpolation between the two formers in order to create the natural system). The footprint is discretized in four types of environmental features that must be adjusted to create the desired natural systems: height (how the terrain height is modified, lowering or raising the nearby area of the shape, for instance), soil (how the surface looks like), vegetation (plants and their location) and water (how much water is available). The shape is described as a set of connected control points; a shape can be a curve (first and last control points not connected) or an area (first and last points connected). The footprint and the shape are independent elements, so it is possible to design footprints and shapes separately and mix then according to the game requirements. In order to design a river, for instance, the operator must create a footprint that has the river characteristics e.g. lower heights in the center of the shape, mud between the water and the rest of the surrounding surface, small plants close to the mud and water to fill the lower height part. After the footprint creation, the operation has to define the shape (place the control points in the map). The procedure will combine the footprint and the shape, resulting in a complete river (with water, plants, mud, etc) following the defined path. If the river course must be changed, the operator just need to adjust the control points; if the river appearance must be tweaked, the operator can change the footprint and all shapes using that footprint in the map will automatically be updated to reflect the new appearance. This technique is not limited to rivers, it can be used to other natural phenomena such as canyons. Figure 11 shows the creation of a meander using natural systems.

## IV. Cities

Cities detonates the human presence in the environment and they are widely used in games such as fly simulators. The process of modeling a whole city is a time consuming task and it requires the analysis of many aspects since

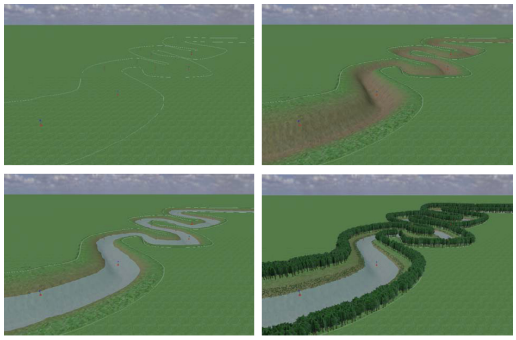

Figure 11. Creating a meander using natural systems: designing the shape in the game world then applying the height/soil, water and vegetation features [40].

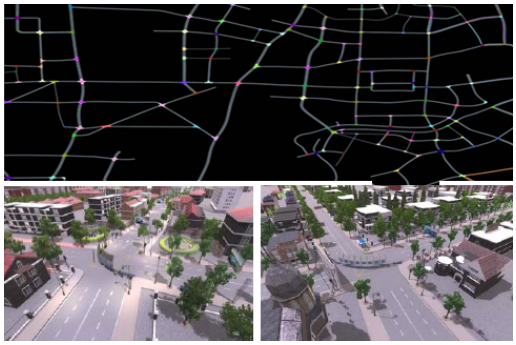

Figure 12. Road network generation on the top; below the same road network applied to a real-time application [45].

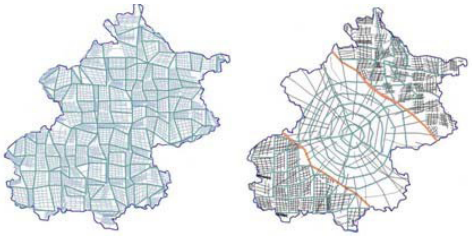

Figure 13. Urban map generation. On the left the grid style only. On the right the mixed styles (grid, radial, user defined) [46].

the city is rich in details, build over human influences and have historical background. The major aspects behind the urban areas are population, environment, transports, vegetation, streets, architecture, elevation, geology and culture. In addition to that, cities have buildings and roads as their main components. As a consequence of that in order to create a complex and a convincing urban area it is important to generate a wide diversity of buildings and roads. Those elements must be generated according to the city cultural and geographic backgrounds, so they are difficult objects to generate automatically because of their individual characteristics like style and function. The city generation subject is recurrent and it is the focus of several related works [41], [42], [43], [44].

An assisted approach for generating complex road networks for cities is based on Bezier curves and real world data [45]. Using an aerial image of the road network of a real city, the operator marks the paths using Bezier curves placed at the center of the road. After that the method interpolates the curves finding important characteristics and enhancing them such as interceptions. Those intersections and junction are created using curve approximation. During the process new elements are added based on the curves such as the sidewalks. Those elements are generated using a process of polygonal approximation. Figure 12 shows a generated road network and its usage on a real-time application.

Another assisted approach for road generation in cities is the procedural interactive method based on template techniques and parametrization algorithms [46]. The method consists of a random traveling algorithm to connect attributed points and uses templates to define the growing patterns for road networks. The template mechanism is like a simplified L-system. The work-flow is described by three main steps. In the first one the primary roads are created based on the specified points. The secondary roads are created though templates applied to the area or using parametrization mapping. In the last step is the generation of the third level of roads. The interactive part of that system allows the user to control the creation aspects and enables modification of road patterns, generation of new roads, tweaking road density and defining a new population parameter. Most of the operations are fully interactive, so the user intervention can improve the reality of the road network generated. Figure 13 illustrates a generated urban map.

An assisted approach to procedurally generate urban ecosystems is the interactive procedural system [47]. That system consists of two main processes: urban model generation and plant model generation. A socio-economical and geometrical simulation approach is used in order to generate the urban model e.g [47], [48]. The plant management algorithm determines the manageability level of contained plants for each city block. The manageability level determines the percentage of wild plants allowed in each area that later will be used by ecosystem simulation. A procedural planting algorithm seeds the plants and a competition based ecosystem simulation is used to determine the plants that will remain. In order to handle the simulation of the plants development a symmetric and asymmetric plant competition approach can be used e.g. [49], [50], [51]. Figure 14 shows the illustration of an urban ecosystem that is still evolving over the time (trees are growing).

Another assisted approach for city generation is called *CityEngine*, a system capable of modeling a complete city using a small set of statistical and geographical input data and is highly controllable by the user [3]. Instead of generating all the content procedurally it creates the city based on a

Figure 14. Urban ecosystem that evolved over the time [47].

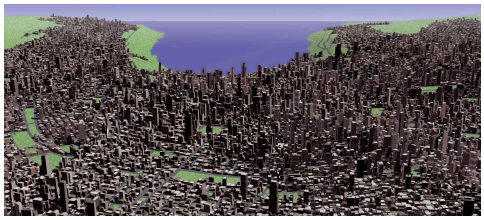

Figure 15. A virtual city with approximately 26000 buildings [3].

pipeline composed of several tools. The first step is the road-generation system that receives the user input (e.g. water, elevation and population density maps). After the network of streets is created the remaining areas (between the roads) are subdivided in allotments and filled with buildings. The first two steps are based on a L-system. Figure 15 shows a virtual city generated based on imaginary water, elevation and population density maps. Figure 16 shows a virtual representation of Manhattan.

A non-assisted approach for city generation is demonstrated in a system called PG3D [52]. The tool was designed to create realistic urban environments based in a spatial database engine. The process works through a set of stored procedures in a database and it's able to generate virtual environments with real content. In order to achieve that

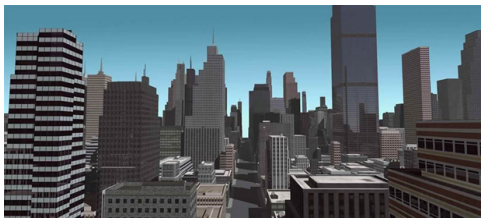

Figure 16. Virtual Manhattan (Maya render) [3].

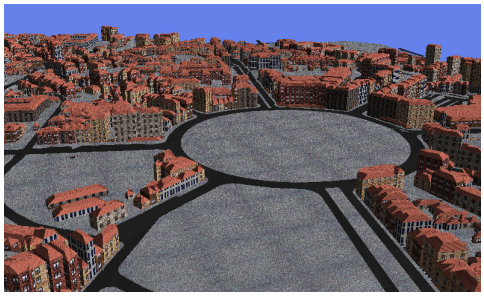

Figure 17. A procedurally generated environment [52].

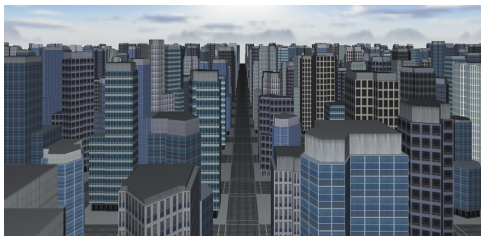

Figure 18. Real-time procedural virtual city [53].

goal PG3D uses its own grammar and shapes; the grammar is used to define the production and structures rules that are used in the process of procedural urban environment generation. The production uses a set of shapes that contains geometries that represent several types of element such as surfaces, streets, buildings and so on. Figure 17 shows a large environment procedurally generated using this technique.

Another non-assisted approach is the creation of pseudo-infinite cities generated on demand as the user walks through the terrain area [53]. The entire city is divided in equally sized square cells (blocks of the city). Each cell has a set of buildings that are determined and generated based on a seed related to the cell position. The generation process is also influenced by a global seed. Since every building is created based on a seed (cell position) if the user returns to a particular location the same buildings will be present. All geometrical components of the city are generated as they are encountered by the user. Figure 18 shows a procedurally generated city achieved with this approach.

## V. Conclusion

This paper presented an overview of several procedural content generation techniques. All techniques were grouped according to their resulting content such as terrains/continents, roads/rivers and cities. Additionally each technique was classified as assisted or no-assisted highlighting the need of human interaction or guidance in order to produce content using the referenced technique. Classical and modern methods were reviewed providing substantial information that can be used directly in the game develop-

ment industry or as a starting point for further research and creation of new techniques.

In section II five approaches were presented. Three of them were assisted methods and both the assisted and non-assisted techniques presented unique approaches to procedural content generation. Those techniques combine different methodologies such as control curves, real images, genetic algorithms, software agents and the definition of pixel values based on neighbors comparisons. Section III presented the complexity of roads and rivers generation, a process influenced and guided by elements such as trajectory calculation and realistic patterns. Two papers were reviewed demonstrating assisted and non-assisted methods. Finally section IV presented techniques related to city generation. Six papers were reviewed and more than three were assisted methods. The featured techniques used Bezier curves, template techniques, urban ecosystem and pseudo-random numbers in order to produce convincing procedurally generated cities.

Non-assisted techniques are more suitable to be used when a large amount of content must be generated and such content may not impact in the game play directly e.g. the very distant surrounding areas of an air base in a flight simulator game. A non-assisted approach can also be used as a starting point for further designing tasks such as the creation of raw data for a terrain that will be later refined by the game designer. However a non-assisted approach is mandatory in cases where part of the game content is generated on-the-fly according to the user decisions; in such cases the content is not predictable for the game designer (for any reason ) and needs to be procedurally generated according to the player context. The creation of a virtual infinite world requires the use of non-assisted techniques for instance.

Assisted techniques are more suitable to be used when a well defined and constrained content set is required. Assuming the player will spend most of the time deeply exploring and interacting with such content it must contain a high rate of details. The assisted techniques can decrease the creation time because they help the operator to design detailed content using a guided generation that fully creates or enhances the final content. Since the assisted techniques require human intervention and guidance they are more likely to produce easily predictable content, but their usage may not be adapted for on-the-fly generation of content in games. Some assisted approaches demand a significant amount of time and effort from the user in order to produce the desired result, so they tend to be better used if integrated as part of designing tools such as level editors.

Analyzing the evolution of procedurally content generation techniques over the years one can notice a tendency of combination between assisted and non-assisted methods. Even if a technique relies heavily on human guidance to produce results, it uses that guidance as a constraint in order to generate detailed content. The use of control curves based on diffusion equations, for instance, requires the operator to place the curves and to adjust its parameters (noise, height, etc) in order to produce a terrain. However the technique is the main responsible for the interpolation and application of such parameters in order to produce the result. That *modus operandi* is an evolution and a combination of completely procedural generation approaches (e.g. a height map created purely with Perlin noise) and human-only generation approaches (e.g. a map designed exclusively by an artist). The amount of effort the operator must spend in adjustments defines how independent the method is during the generation process. Despite of that there is always the concern about the content randomness, because a procedurally generated content is useful mainly if it fits the context of the game it is inserted into. A completely random content has no meaning because it is not predictable and as a consequence it has no context.

This survey can help studies focused on procedural content generation targeted for game development. It can be used to analyze techniques that are able to decrease the costs of game production as well as enriching the featured content.

## REFERENCES


[1] H. Hnaidi, E. Guerin, S. Akkouche, A. Peytavie, and E. Galin, "Feature based terrain generation using diffusion equation," *Computer Graphics Forum (Proceedings of Pacific Graphics)*, vol. vol.29, no. n.7, 2010.

[2] H. Zhou, J. Sun, G. Turk, and J. M. Rehg, "Terrain synthesis from digital elevation models," *IEEE Transactions on Visualization and Computer Graphics*, vol. vol.13, 2007.

[3] Y. I. H. Parish and P. Müller, "Procedural modeling of cities," *Proceedings of the 28th annual conference on Computer graphics and interactive techniques SIGGRAPH*, no. August, 2001. [Online]. Available: http://portal.acm.org/citation.cfm?doid=383259.383292

[4] S. Greuter, N. Stewart, and G. Leach, *Beyond the Horizon: Computer-generated, Three-dimensional, Infinite Virtual Worlds without Repetition*, 2004.

[5] M. Nitsche, C. Ashmore, W. Hankinson, R. Fitzpatrick, J. Kelly, and K. Margenau, *Designing procedural game spaces: A case study*. Citeseer, 2006.

[6] J. Togelius, R. De Nardi, and S. M. Lucas, "Towards automatic personalised content creation for racing games," *2007 IEEE Symposium on Computational Intelligence and Games*, 2007.

[7] K. Perlin, "An image synthesizer," in *Annual Conference on Computer Graphics*, vol. 19, 1985.

[8] P. Prusinkiewicz and A. Lindenmayer, *The algorithmic beauty of plants*. Springer, 1996.

[9] B. Mandelbrot, *Fractals: Form, Chance, and Dimension*. Freeman, 1977, vol. 1.

[10] S. C. Dollins, "Modeling for the plausible emulation of large worlds," Ph.D. dissertation, Brown University, United States of America, 2002.

[11] H. Haggström, "Real-time generation and rendering of realistic landscapes," Ph.D. dissertation, Citeseer, 2006. [Online]. Available: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.98.5489

[12] B. Lintermann and O. Deussen, "A modelling method and user interface for creating plants," 1998, computer Graphics Forum.

[13] O. Linda, "Generation of planetary models by means of fractal algorithms," Czech Technical University, Tech. Rep., 2007.

[14] P. Prusinkiewicz and M. Hammel, "A fractal model of mountains with rivers," in *Proceeding of Graphics Interface 93*, 1993.

[15] F. K. Musgrave, C. E. Kolb, and R. S. Mace, "The synthesis and rendering of eroded fractal terrains," *Annual Conference on Computer Graphics*, vol. vol.23, 1989.

[16] J. Olsen, "Realtime procedural terrain generation - realtime synthesis of eroded fractal terrain for use in computer games," *Department of Mathematics And Computer Science IMADA University of Southern Denmark*, 2004.

[17] A. D. Kelley, M. C. Malin, and G. M. Nielson, "Terrain simulation using a model of stream erosion," in *Annual Conference on Computer Graphics*, vol. 22, 1988.

[18] P. Roudier, B. Peroche, and M. Perrin, "Landscapes synthesis achieved through erosion and deposition process simulation," *Computer Graphics Forum*, vol. vol.12, 1993.

[19] N. Chiba, K. Muraoka, and K. Fujita, "An erosion model based on velocity fields for the visual simulation of mountain scenery," *Journal of Visualization and Computer Animation*, vol. vol.9, 1998.

[20] K. Nagashima, "Computer generation of eroded valley and mountain terrains," *The Visual Computer*, vol. vol.13, 1998.

[21] B. Benes and R. Forsbach, "Layered data representation for visual simulation of terrain erosion," in *spring conference on computer graphics*, 2001.

[22] K. R. Kamal and Y. S. Uddin, "Parametrically controlled terrain generation," in *Computer graphics and interactive techniques in Austalasia and South East Asia*, 2007.

[23] R. Szeliski and D. Terzopoulos, "From splines to fractals," in *Annual Conference on Computer Graphics*, vol. 23, 1989.

[24] A. Peytavie, E. Galin, J. Grosjean, and S. MÈrillou, "Arches: a framework for modeling complex terrains," *Computer Graphics Forum*, vol. vol.28, 2009.

[25] J. E. Gain, P. Marais, and W. Straüer, "Terrain sketching," in *ACM Symposium on interactive 3D graphics*, 2009.

[26] B. Rusnell, D. Mould, and M. G. Eramian, "Feature-rich distance-based terrain synthesis," *The Visual Computer*, vol. vol.25, 2009.

[27] A. Orzan, A. Bousseau, H. Winnemller, P. Barla, J. Thollot, and D. Salesin, "Diffusion curves: a vector representation for smooth-shaded images," *ACM Transactions on Graphics*, vol. vol.27, 2008.

[28] G.-S. Song and S.-K. Hsu, "Automatic extraction of ridge and valley axes using the profile recognition and poligon-breaking algorithm," *Computers And Geosciences*, vol. vol.24, no. n.1, 1998.

[29] P. Walsh and P. Gade, "Terrain generation using an interactive genetic algorithm," in *Evolutionary Computation (CEC), 2010 IEEE Congress*, july 2010.

[30] T. J. Ong, R. Saunders, J. Keyser, and J. J. Leggett, "Terrain generation using genetic algorithms," in *Proceedings of the 2005 conference on Genetic and evolutionary computation*, ser. GECCO '05. New York, NY, USA: ACM, 2005. [Online]. Available: http://doi.acm.org/10.1145/1068009.1068241

[31] M. Frade, F. F. de Vega, and C. Cotta, "Breeding terrains with genetic terrain programming: The evolution of terrain generators," *International Journal of Computer Games Technology*, vol. vol.2009, 2009.

[32] R. L. Saunders, "Realistic terrain synthesis using genetic algorithms," Texas A&M University, 2006.

[33] J. Doran and I. Parberry, "Controlled procedural terrain generation using software agents," *IEEE Transactions on Computational Intelligence and AI in Games*, vol. vol.2, no. n.2, 2010.

[34] F. Bevilacqua, C. T.Pozzer, and M. C. d'Ornellas, "Charack: tool for real-time generation of pseudo-infinite virtual worlds for 3d games," *Brazilian Symposium on Games and Digital Entertainment*, no. n.8, October 2009.

[35] G. Chen, G. Esch, P. Wonka, P. Müller, and E. Zhang, "Interactive procedural street modeling," *ACM Trans. Graph.*, vol. vol.27, August 2008. [Online]. Available: http://doi.acm.org/10.1145/1360612.1360702

[36] D. G. Aliaga, C. A. Vanegas, and B. Beneö, "Interactive example-based urban layout synthesis," *ACM Transactions on Graphics*, vol. vol.27, no. n.5, 2008. [Online]. Available: http://portal.acm.org/citation.cfm?doid=1409060.1409113

[37] C. A. Vanegas, D. G. Aliaga, B. Benes, and P. Waddell, "Visualization of simulated urban spaces: inferring parameterized generation of streets, parcels, and aerial imagery." *IEEE Transactions on Visualization and Computer Graphics*, vol. vol.15, no. n.3, 2008. [Online]. Available: http://www.ncbi.nlm.nih.gov/pubmed/19282549

[38] J. Sun, X. Yu, G. Baciu, and M. Green, "Template-based generation of road networks for virtual city modeling," in *Virtual Reality Software and Technology*, 2002.

[39] E. Galin, A. Peytavie, N. MarÈchal, and E. GuÈrin, "Procedural generation of roads," *Computer Graphics Forum*, vol. vol.29, 2010.

[40] R. Huijser, J. Dobbe, W. F. Bronsvoort, and R. Bidarra, "Procedural natural systems for game level design," *Games and Digital Entertainment, Brazilian Symposium*, no. November, 2010.

[41] K. Lynch, *The Image of the City*. The MIT Press, 1960, vol. 21, no. October. [Online]. Available: http://www.amazon.com/dp/0262620014

[42] C. Alexander, S. Ishikawa, and M. Silverstein, "A pattern language: Towns, buildings, construction (center for environmental structure series)," 1977, new York.

[43] G. Kelly and H. McCabe, "A survey of procedural techniques for city generation," *ITB*, no. December, 2006.

[44] B. Watson, P. Müller, O. Veryovka, A. Fuller, P. Wonka, and C. Sexton, "Procedural urban modeling in practice," *IEEE Computer Graphics and Applications*, vol. vol.28, 2008.

[45] L. Gang and S. Guangshun, "Procedural modeling of urban road network," *Information Technology and Applications (IFITA)*, vol. vol. 1, 2010.

[46] X. Dou, Y. Qi, F. Hou, and X. Shen, "Interactive urban map design with template and parameterization," *Image and Signal Processing, 2009. CISP '09. 2nd International Congress*, vol. vol. 2, 2009.

[47] B. Beneš, M. A. Massih, P. Jarvis, D. G. Aliaga, and C. A. Vanegas, "Urban ecosystem design," in *Symposium on Interactive 3D Graphics and Games*, ser. I3D '11. New York, NY, USA: ACM, 2011. [Online]. Available: http://doi.acm.org/10.1145/1944745.1944773

[48] C. A. Vanegas, D. G. Aliaga, B. Beneš, and P. A. Waddell, "Interactive design of urban spaces using geometrical and behavioral modeling," *ACM Trans. Graph.*, vol. vol.28, December 2009. [Online]. Available: http://doi.acm.org/10.1145/1618452.1618457

[49] M. Alsweis and O. Deussen, "Modeling and visualization of symmetric and asymmetric plant competition," *Natural Phenomena 2005*, 2005.

[50] P. Prusinkiewicz, "Generating spatial distributions for multilevel models of plant communities," in *Graphics Interface*, 2002.

[51] O. Deussen, P. Hanrahan, B. Lintermann, R. Měch, M. Pharr, and P. Prusinkiewicz, "Realistic modeling and rendering of plant ecosystems," in *Proceedings of the 25th annual conference on Computer graphics and interactive techniques*, ser. SIGGRAPH '98. New York, NY, USA: ACM, 1998. [Online]. Available: http://doi.acm.org/10.1145/280814.280898

[52] P. B. Silva and A. Coelho, "Procedural modeling of urban environments for digital games development," in *Proceedings of the 7th International Conference on Advances in Computer Entertainment Technology*, ser. ACE '10. New York, NY, USA: ACM, 2010. [Online]. Available: http://doi.acm.org/10.1145/1971630.1971667

[53] S. Greuter, J. Parker, N. Stewart, and G. Leach, "Real-time procedural generation of 'pseudo infinite' cities," in *Proceedings of the 1st international conference on Computer graphics and interactive techniques in Australasia and South East Asia*, ser. GRAPHITE '03. New York, NY, USA: ACM, 2003. [Online]. Available: http://doi.acm.org/10.1145/604471.604490