Procedural Attack! Procedural Generation for Populated Virtual Cities: A Survey

Werner Gaisbauer[1], Helmut Hlavacs[2]
[1] University of Vienna, werner.gaisbauer@gmail.com
[2] University of Vienna, helmut.hlavacs@univie.ac.at

Abstract

On the one hand, creating rich virtual worlds "by hand" like in the game Grand Theft Auto V is hugely expensive and limited to large studios. On the other hand, procedural content generation (PCG) allows tiny teams to create huge worlds like Hello Games did with only four people (in the beginning) for the recently released game No Man's Sky. Following in the footsteps of Hello Games, this paper tries to equip the reader with an overview about the state-of-the-art of how to build such a virtual world, i.e., a populated virtual city with buildings, streets, parks, vegetation, humans, and vehicles, using just PCG assets. Each PCG asset that is envisioned to bring the city to life is grouped and discussed in detail and the latest research trends in PCG are presented together with open questions. Using the above-mentioned PCG assets, instead of months, a city can be built in a mere couple of minutes by a user without much experience in designing 3D assets. The city can then be used for many applications like games, virtual reality (VR), or film.

Keywords: *Virtual Worlds, PCG, 3D Assets, Games, VR;*

1. Introduction

Vast universes can be built with the help of procedural content generation (PCG) [1]. The recently released game "No Man's Sky" [2] proves that a small team of people can build such a procedurally generated universe with astonishing detailed planets, plants, and creatures. Following in their footsteps, the goal of this paper is to harness the power of PCG and equip the reader with an overview about the state-of-the-art of how to make a virtual world, i.e., a populated city with buildings, streets, parks, vegetation, and vehicles, inhabited by virtual people, with as little effort as possible. It should be possible to customize the city, by spawning different kinds of assets from a palette (e.g., buildings, virtual people, streets, parks etc.). The idea is, to apply PCG algorithms, to create all the distinct assets and bring the city to life. The different PCG assets that are presented in this paper can be thought of as a foundation for a "city construction toolkit". To spark the imagination of the reader, an example for such a procedurally generated city is shown in Figure 1 from the indie game "The Hit" [3].

Figure 1. Generated buildings, pedestrians and vehicles in the indie game "The Hit".

In the vision of this paper, to build the virtual city, the user just should invoke different kinds of PCG algorithms via an interface (a graphical user interface (GUI), where the mouse plays a major role, i.e., the user "clicks together" content like puzzle pieces) and the computer builds parts of the city in a fully automated way (without the need for an artist to provide user content) as described in this research [4]. Usually, PCG algorithms have parameters (e.g., a template of a house has a given building lot and a height, or an office building has a basic configuration of how the walls are placed) that can be modified by the user to create content to her desire [5]. To make building quicker and easier for the user, a basic configuration for each PCG algorithm template is already provided by default, and the user can then iteratively change the suggested design via moving the walls, etc. (similarly, as done in this research [4]). All assets that are generated via PCG algorithms already have physical properties. The virtual people act per a standard behavior and walk on random waypoints by default, all of which can be customized by the user [6].

There are a lot of possibilities for using a populated virtual city (e.g., the generated cities can be used for VR games/experiences) and the procedural assets of the city construction toolkit can also be used "per se" in other (non-urban) settings as well. Game designers, film makers, and even researchers might be interested in the city construction toolkit for various purposes, as many games[1] and films have urban settings and there is a lot of interesting future research possible. The user of the toolkit does not have to be an artist or graphic designer (but can be of course), which is a big advantage, and cities can be rapidly built in just a few minutes, so many alternatives might be tried in a reasonable time frame.

2. Method

This section tries to explain how the literature review (survey) in this paper was done, so it is possible to evaluate how the sources were selected in principle. We did not follow a strict protocol in the sense of a running a replicable experiment but rather used a relaxed procedure for performing the literature review. However, using this relaxed procedure, it should be possible to follow in principle how the sources in this paper were selected and with what kind of bias. The relaxed procedure is described in as much detail as possible in this section.

2.1 Research Questions

The goal of this paper is to equip the reader with an overview about the state-of-the-art of how to make a procedural populated virtual city. The following research questions were derived from that general goal:

RQ1: What is the state-of-the-art of how to make procedural assets for use in populated virtual cities, i.e., buildings, humans, vegetation, and vehicles?
RQ2: What is the state-of-the-art of how to make a procedurally generated populated virtual city themeable?
RQ3: What is the state-of-the-art of how to handle crowds of people in a procedural way?
RQ4: What research trends can help create procedurally generated populated virtual cities?
RQ5: What gaps of knowledge (leading to open questions) are there in the current state-of-the-art of how to make a procedurally generated populated virtual city?

2.2 Review Procedure

The review procedure below shows how we performed the literature review in a quasi-systematic way:

1. Specifying the research questions: We specified the research questions based on our general goal for the paper.
2. Developing a review procedure: We developed a relaxed procedure for performing our literature review.
3. Searching for sources: We searched for sources related to each research question using our predefined search terms.
4. Screening found sources: Screening was done via matching the found sources (in the search step) against the predefined inclusion criteria.

[1] Procedurally generated narrative puzzles like the one used in the prototype game "Stranded in Singapore" [7] are a possibility for the player to explore the city in a fun way.

5. Synthesizing included sources: The sources included into the review (in the screening step) were synthesized into an overview about the state-of-the-art of how to make a procedural populated virtual city (i.e., the general goal of this paper).

2.3 Searching and Screening Sources

We initially used the following search terms for finding sources that answer the research questions stated above in the best possible way:

RQ1:

- *"procedural" AND "cities" OR "buildings" OR "humans" OR "vegetation" OR "vehicles"*
- *"virtual humans"*
- *"procedural vegetation l-systems"*
- *"vehicle animation" AND "maxscript" OR "autonomous"*

RQ2:

- *"neural artistic style"*
- *"patch based"*
- *"deep forger"*

RQ3:

- *"populating virtual environments"*

RQ4[2]:

- *"procedural content generation games"*
- *"interactive evolutionary computation"*
- *"WaveFunctionCollapse[3]"*

RQ5:

- *"procedural content generation" AND "mit" OR "mixed initiative"*

The search terms listed above were just the initial input for starting our search, and we had to iteratively refine those terms in some cases using common sense to find the references used in this study (some references related to RQ4 were inspired by an online video[2]). Basically, we searched for sources using mostly Google Scholar and sometimes Google search engine using the previously defined search terms (also screening related articles). We would like to note that, while it was easy to find sources (related to RQ1) for procedurally generated buildings and vegetation, it was much harder for procedural humans and vehicles, as there is currently not much material in the scientific literature available, so we had to fall back on searching for sources more generally also using Google search engine. The sources were screened and included according to our predefined inclusion criteria (see Table 1). Strict inclusion criteria were used as hard logical conditions while loose inclusion criteria acted more like a bias towards what we were looking for when screening sources. We were mostly using common sense when deciding which sources to include based on our criteria trying to make the most of our relaxed review procedure regarding added value for the reader.

[2] Part of the references for this research question was inspired by this Game Developers Conference talk in the AI Summit track [8].

[3] This reference was suggested via personal communication by a peer student.

ISSN: 2384-8766 http://dx.doi.org/10.17083/ijsg.v1i4.47

Table 1. Inclusion criteria for the review.

Criterion type	Inclusion criteria
Topic (strict)	Source topic must be related to the procedural generation of populated virtual cities and address at least one of the research questions.
Citation impact (loose)	The source should be "famous" (i.e., the source has a high number of citations).
Publication date (loose)	The source should be recent (we tried to prefer recently published sources, but a source being "famous" was more important).
Type (loose)	The source should help make a "*photorealistic*" procedural populated virtual city (like the one, for example, in the AAA game Grand Theft Auto V).

2.4 Synthesizing Sources

In total, we included 40 sources into our literature review using our review procedure for searching and screening sources. We synthesized all the included sources into an overview about the state-of-the-art of how to make a procedural populated virtual city (to reach the general goal of this paper) divided into state-of-the-art procedural assets (see Section 3) and research trends and open questions (see Section 4). Section 3 is related to RQ1, RQ2, and RQ3 and Section 4 is related to RQ4 and RQ5.

3. State-of-the-Art Procedural Assets

3.1 Buildings

Shape grammars (e.g., *CGA shape*[4]) have been used to procedurally generate very realistic and detailed buildings for different kinds of applications, at low cost: a modern city model with office buildings, a wealthy suburbia inspired by Beverly Hills with single family homes, and even a Pompeii model [5]. Other ways that have been successfully used to procedurally generate buildings are L-systems[5] [11] (see
Figure **2**), markup languages [12] (e.g., for Introversion's suspended computer game Subversion[6]), and split grammars [14]. L-systems also have been utilized to generate whole complex cities (e.g., a virtual city with approximately 26000 buildings) with a complete street map and buildings at the push of a button [11]. The commercial software Esri CityEngine is an offspring of this technology [15]. A survey of procedural techniques for city generation is given in [16].

Figure 2. Example procedural buildings somewhere in a virtual Manhattan.

[4] CGA shape is a kind of shape grammar, i.e., a unique "programming language" specified to generate architectural 3D content. The term CGA stands for Computer Generated Architecture. The idea of grammar-based modeling is to define rules that iteratively refine a design by creating more and more detail [9].

[5] In 1968, Hungarian botanist Aristid Lindenmayer developed a grammar-based system to model the growth patterns of plants. L-systems (short for Lindenmayer systems) can be used to generate all kinds of recursive fractal patterns (usually vegetation but also buildings, road patterns, and even whole virtual cities). They are incredibly useful because they provide a mechanism for keeping track of fractal structures that require complex and multi-faceted production rules [10].

[6] Nevertheless, the city generator technology demonstration looks very impressive, despite being from 2007 [13].

3.1 Themeable Cities

An interesting idea to follow when creating procedural city assets is that each asset can be "themeable" (e.g., the user can apply dark themes that produce cities like the cyberpunk city universe described in the classic book Neuromancer [17], or even ancient kind of themes could be possible [18]). Texture generation for applying city themes is an important future research direction and interesting results might be achieved by using deep learning algorithms [19-22] and WaveFunctionCollapse [23]. Project Sprawl (see
Figure **3**) is an experimental open world role-playing simulation set in a procedurally generated cyberpunk city [24]. This project idea might serve as an inspiration for creating unique procedural buildings with a theme.

Figure 3. Fragment from the development of Project Sprawl, a cyberpunk role-playing simulation.

3.2 Humans

The handbook of virtual humans, although somewhat dated, covers a lot of ground on how to make virtual humans [25], whereas this research [26] presents the basic algorithms used by MakeHuman, an open source tool for making 3D characters. The authors present a new interfacing idea that is much easier to use than conventional systems, which usually have thousands of parameters. Accessing the numerous parameters required in modeling the human form is made possible via a new widget which uses fuzzy logic.

Of interest for character generation is also the following paper [27], which discusses a method for animating characters automatically, instead of manual rigging of characters (where the human animator has to specify its internal skeletal structure and to define how the input motion deforms its surface).

The UMA (Unity Multipurpose Avatar) 2 package [28] is an asset for the Unity game development platform [29] that allows the user to create customizable characters. UMA is a vibrant open source community (the software is available on GitHub). The UMA package lets the user change 46 modifiable values to adjust everything from height to ear rotation, resulting in an almost infinite variety of custom characters. Also, the user can define her own extensions to extend the existing humanoid or fully replace it with her own idea of how it should be shaped.

Figure **4** shows the UMA framework in action (the latest version can easily handle a hundred of unique avatars).

Figure 4. 100 avatars in real time generated by UMA framework.

ISSN: 2384-8766 http://dx.doi.org/10.17083/ijsg.v1i4.47

3.3 Vegetation

The usual way to create realistic procedural vegetation is via L-systems. A good and well known overview of this topic (although a bit dated) with a lot of visual examples is given in the book [30] and also in the more recent PCG book [31]. Very realistic looking animated trees that interact with wind in real time were achieved in [32] (see
Figure **5**), using a combination of statistical observations with physical properties for tree animation that can be animated using modern GPUs at low cost. In [33], self-organizing tree models were used to generate a wide range of realistic trees and bushes. It is interesting to note that in this research, the generated trees can be modified by a human user via procedural brushes, sketching, and editing operations such as pruning and bending of branches. Evolutionary methods for generating trees are discussed in [34] and in [35] for plants. A survey about the most used tools for procedural generation of plants is presented in [36].

Figure 5. Freeze frame of animated tree.

3.4 Vehicles

The approach presented in this research [37] (which uses PCG but also non-procedural techniques) is interesting "per se" for creating animation-ready vehicle models. Using the automated tools that are described in the upper-mentioned research, animating an automobile, or multiple automobiles of varying form and dimensions is possible with minimal input and setup. The presented program is targeted towards creating believable vehicles in a time efficient way. The program was capable of automatically creating articulation for 5 unique automobile models including a 2010 Dodge Challenger, a 2009 Dodge Charger, a 1957 Fiat 500, a 2009 Toyota FJ Cruiser and a city bus (see
Figure **6**).

Complementing the research by Griffin described above is this research [38], which presents a method for synthesizing animations of autonomous space, water, and land-based vehicles in games or other interactive simulations. The research builds an animation framework suitable for interactive vehicle simulations, which can be used for synthesized animations of vehicles performing a variety of autonomous behaviors (e.g., Seek, Pursue, Avoid, Avoid Collision, and Flee).

Figure 6. Animation-ready vehicle models.

3.4 Crowd Patches

This research [6] uses blocks containing a pre-computed local crowd simulation (called crowd patches) to make it possible to simulate densely populated large environments. A potentially infinite city with realistic and varied crowds can be simulated using this approach, where the virtual humans in the crowd also interact with each other and the environment in an intelligent way.
Figure **7** shows the approach in action, i.e., a procedurally computed pedestrian street is displayed, where crowd patches are generated at run-time.

Figure 7. Environment successfully populated with crowd patches.

4. Research Trends and Open Questions

4.1 Mixed-Initiative Mode

To help the creative process of the user, the computer can assist the user in a mixed-initiative fashion (a good overview of the topic related to PCG is given in [1] and PCG for game designers in [39]). This assistance can come in various shapes, for example, the tool can suggest various new designs based on the current design of the asset (e.g., maps [40] or buildings [12]) that is being viewed and/or edited by the user. In these kind of systems, the user can then usually select one of these suggestions and the computer re-iterates and displays the next number of suggestions and so on. This way, new maps and buildings can be quickly generated that go into a certain direction which the designer envisions (e.g., maps with a lot of parks, or buildings that look like skyscrapers).

4.2 Interactive Evolutionary Computation

In the spirit of this idea, state-of-the-art PCG methods like generative grammars (e.g., L-systems [11][31], or markup languages [12] can be augmented with techniques from artificial intelligence (e.g., Interactive Evolutionary Computation (IEC) [41]) to create mixed-initiative tools that assist the human user (designer) to shape new PCG content. A closer examination of the interplay between PCG methods and IEC is presented in Figure 8.

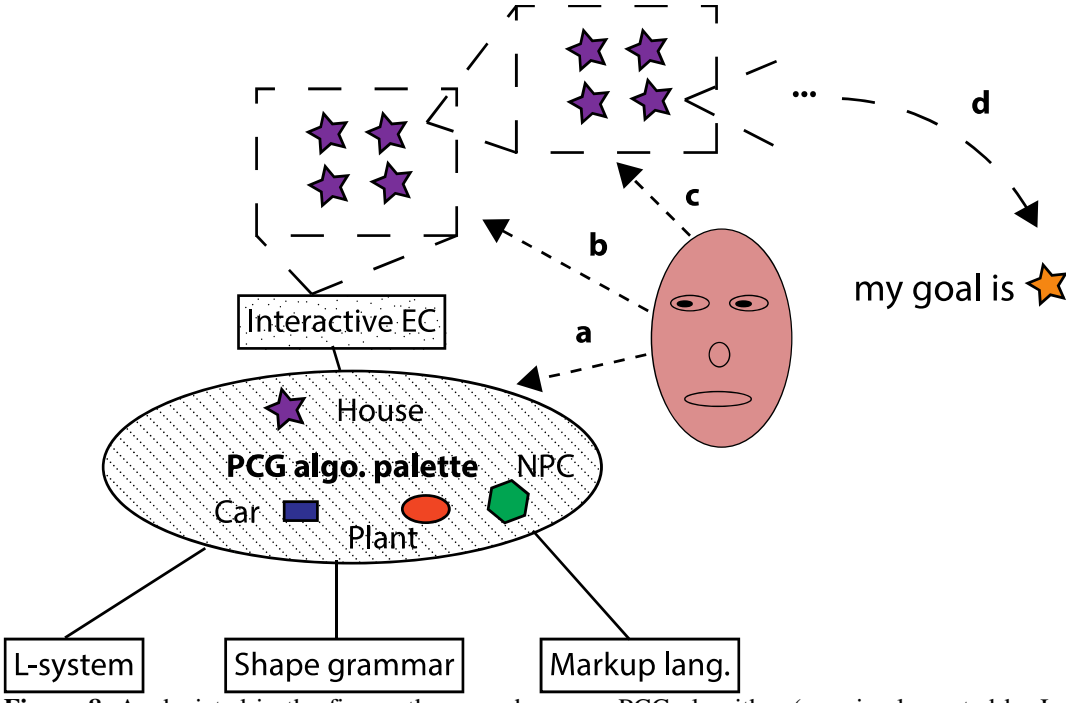

Figure 8. As depicted in the figure, the user chooses a PCG algorithm (e.g., implemented by L-system, shape grammar, or markup language) from a palette, for example, a procedurally generated house (depicted by the violet star shape) (a), and observes the different suggested houses by the IEC system (depicted by the four star shapes) (b). With already a goal in mind, e.g., different houses that possess a common impression, e.g., expensive looking family houses (depicted by the orange star) (d), the user selects a specific instance of a house (b), and the computer generates the next generation of houses (c) that depend on the user's previous selection (b). After a few generations, the goal of the user is reached (d) and the "customized" house can be put into the virtual city.

4.3 WaveFunctionCollapse

Another starting point for creating new mixed-initiative city construction tools, is WaveFunctionCollapse (WFC), an internet famous computer program that was recently published on GitHub [23], which does bitmap & tilemap generation from a single example with the help of ideas from quantum mechanics based on the following main inspirations [42][43]. Those bitmaps and tilemaps can be used to generate interesting 3D cities that can then be used for various purposes. The WFC algorithm supports constraints. Therefore, it can be easily combined with other generative algorithms or with manual creation (e.g., autocompleting a city tilemap started by a human). This is ideal for making a mixed-initiative style tool because the computer can suggest new city designs based on the input of the human user and support what ideas are already in the user's head. This kind of mixed-initiative approach saves time and effort for the user and is likely to lead to more interesting results because the user has more time thinking about certain city designs (the computer can suggest a small number of different designs to the user) instead of wasting time with "micromanaging" like completing tilemaps.

Tilemaps are easily compatible with crowd patches [6]. Each possible tile can be thought of as a kind of crowd patch (e.g., there can be predefined tiles for busy intersections like the "Shibuya Crossing" in Tokyo, casual shopping streets, or a Beverly Hills like area of houses with a lot of palms etc.) and the city can then be assembled from these tiles using the before mentioned mixed-initiative mode. This combination hasn't been researched before (to the best knowledge of the author) and is especially well suited for creating living cities with ease.

4.4 Open Questions

The following quote is an interesting open question: "All the designers I can think of working on PCG come from computer science. How can we make procedural content generation accessible to non-programmers, or at least people who don't have a strong background on CS?" [44]. This points towards mixed-initiative tools because they are usually aimed at humans without much

programming skills. In stark contrast to needing programming skills, these tools are often very supportive to non-programmers, everyday people, artists, or designers. Future work could try closing the gap (making PCG accessible to non-programmers) and search for these kinds of mixed-initiative tools that support the upper mentioned groups of people in the context of procedural generation of living cities.

Another possible open question ready to be explored in future work, which hasn't been addressed yet, might be how a discussion between human and computer at the "meta-level" could look like in the scope of this topic (mixed-initiative systems). "Being able to communicate at a meta-level about the design tasks and outcomes has not been well-explored in mixed-initiative PCG work thus far" [45]. This communication at the meta-level could be a better way for the computer to explain itself (and its actions) to the human user, e.g., why did the computer select certain objects over others etc.

5. Conclusions

To summarize, this paper is based on the first author's presentation during the doctorial consortium at the ICEC Conference 2016 in Vienna on Sept. 27. This paper provides an overview of current research for how to create populated virtual cities using just PCG methods, including research trends and open questions for inspiring future work. Emphasis is laid upon the state-of-the-art of what procedural assets are available for city building (i.e., buildings including themeable cities, humans, vegetation, vehicles, and crowd patches are discussed) and a mixed-initiative mode that should help users to easily create a virtual city shaped to their desire. Future work could build upon this survey and use and implement one or more of the procedural assets and/or a mixed-initiative mode as described here for building a virtual city or try to tackle one or more of the open questions.

References

[1] Shaker N., Togelius J., Nelson M.J., Procedural Content Generation in Games: A Textbook and an Overview of Current Research. 2015.
[2] Hello Games, No Man's Sky, 09-Aug-2016. [Online]. Available: http://www.no-mans-sky.com/. [Accessed: 02-May-2017].
[3] Stubbs D., The Hit, 04-Mar-2017. [Online]. Available: https://adfp.itch.io/the-hit. [Accessed: 02-May-2017].
[4] Thompson M.W., Evaluating the Hybridisation of Procedural Content Generation With a Design-Centric Editor. 2015.
[5] Müller P., Wonka P., Haegler S., Ulmer A., Van Gool L., Procedural Modeling of Buildings, presented at the ACM SIGGRAPH 2006 Papers, New York, NY, USA, 2006, pages 614–623. https://doi.org/10.1145/1179352.1141931
[6] Yersin B., Maïm J., Pettré J., Thalmann D., Crowd Patches: Populating Large-scale Virtual Environments for Real-time Applications, presented at the Proceedings of the 2009 Symposium on Interactive 3D Graphics and Games, New York, NY, USA, 2009, pages 207–214. https://doi.org/10.1145/2538528.2538538
[7] Fernández-Vara C., Thomson A., Procedural Generation of Narrative Puzzles in Adventure Games: The Puzzle-Dice System, presented at the Proceedings of the The Third Workshop on Procedural Content Generation in Games, New York, NY, USA, 2012, pages 12:1–12:6.
[8] Sturtevant N., Smith G., Togelius J., Making Things Up: The Power and Peril of PCG, Mar-2015. [Online]. Available: http://www.gdcvault.com/play/1022134/Making-Things-Up-The-Power. [Accessed: 02-May-2017].
[9] Esri R&D Center Zurich, CityEngine Help. [Online]. Available: http://cehelp.esri.com/help/index.jsp?topic=/com.procedural.cityengine.help/html/manual/cga/basics/toc.html. [Accessed: 02-May-2017].
[10] Shiffman D., The Nature of Code. 2012.
[11] Parish Y.I.H., Müller P., Procedural Modeling of Cities, presented at the Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques, New York, NY, USA, 2001, pages 301–308. https://doi.org/10.1145/383259.383292
[12] Martin A., Lim A., Colton S., Browne C., Evolving 3D Buildings for the Prototype Video Game Subversion, in Applications of Evolutionary Computation, Volume 6024, Number 12,

C. Di Chio, S. Cagnoni, C. Cotta, M. Ebner, A. Ekárt, A. I. Esparcia-Alcazar, C.-K. Goh, J. J. Merelo, F. Neri, M. Preuß, J. Togelius, and G. N. Yannakakis, Eds. Berlin, Heidelberg, 2010, pages 111–120.
[13] Introversion Software, Subversion City Generator Introversion Software, YouTube, 22-Nov-2011. [Online]. Available: https://www.youtube.com/watch?v=FR9xI0GgrBY. [Accessed: 02-May-2017].
[14] Wonka P., Wimmer M., Sillion F., Ribarsky W., Instant Architecture, presented at the ACM SIGGRAPH 2003 Papers, New York, NY, USA, 2003, Volume 22, Number 3, pages 669–677. https://doi.org/10.1145/1201775.882324
[15] Esri, Esri CityEngine. [Online]. Available: http://www.esri.com/software/cityengine. [Accessed: 02-May-2017].
[16] Kelly G., McCabe H., A survey of procedural techniques for city generation, ITB Journal, Volume 14, 2006.
[17] Gibson W., Neuromancer. 1984.
[18] Trescak T., Bogdanovych A., Simoff S., City of Uruk 3000 BC: Using genetic algorithms, dynamic planning and crowd simulation to re-enact everyday life of ancient Sumerians, presented at the Proceedings of the Simulation of the Past to Understand Human History Conference, 2014.
[19] Liang X., Zhuo B., Li P., He L., CNN based texture synthesize with Semantic segment, arXiv preprint arXiv:1605.04731, 2016.
[20] Gatys L.A., Ecker A.S., Bethge M., A neural algorithm of artistic style, arXiv preprint arXiv:1508.06576, 2015.
[21] Champandard A.J., Semantic Style Transfer and Turning Two-Bit Doodles into Fine Artworks, arXiv preprint arXiv:1603.01768, 2016.
[22] Champandard A.J., Deep Forger, 2015. [Online]. Available: https://deepforger.com/. [Accessed: 02-May-2017].
[23] ExUtumno, WaveFunctionCollapse, GitHub, 11-Oct-2016. [Online]. Available: https://github.com/mxgmn/WaveFunctionCollapse. [Accessed: 02-May-2017].
[24] Delacian, Delacian's Scrapbook - Project Sprawl - Cyberpunk Procedural City Generation, Tumblr. [Online]. Available: http://delacian.tumblr.com/. [Accessed: 02-May-2017].
[25] Magnenat-Thalmann N., Thalmann D., Handbook of virtual humans. 2005.
[26] Bastioni M., Re S., Misra S., Ideas and Methods for Modeling 3D Human Figures: The Principal Algorithms Used by MakeHuman and Their Implementation in a New Approach to Parametric Modeling, presented at the Proceedings of the 1st Bangalore Annual Compute Conference, New York, NY, USA, 2008, pages 10:1–10:6. https://doi.org/10.1145/1341771.1341782
[27] Baran I., Popović J., Automatic Rigging and Animation of 3D Characters, presented at the ACM SIGGRAPH 2007 Papers, New York, NY, USA, 2007. https://doi.org/10.1145/1275808.1276467
[28] UMA Steering Group, UMA 2 - Unity Multipurpose Avatar, Unity Asset Store, 27-Apr-2015. [Online]. Available: https://www.assetstore.unity3d.com/en/#!/content/35611. [Accessed: 02-May-2017].
[29] Unity Technologies, Unity - Game Engine. [Online]. Available: https://unity3d.com/. [Accessed: 02-May-2017].
[30] Prusinkiewicz P., Lindenmayer A., The Algorithmic Beauty of Plants. New York, NY, USA, 1996.
[31] Togelius J., Shaker N., Nelson M.J., Grammars and L-systems with applications to vegetation and levels, in Procedural Content Generation in Games: A Textbook and an Overview of Current Research, N. Shaker, J. Togelius, and M. J. Nelson, Eds. 2015.
[32] Habel R., Kusternig A., Wimmer M., Physically Guided Animation of Trees, Computer Graphics Forum (Proceedings EUROGRAPHICS 2009), Volume 28, Number 2, 2009.
[33] Palubicki W., Horel K., Longay S., Runions A., Lane B., Mech R., Prusinkiewicz P., Self-organizing Tree Models for Image Synthesis, presented at the ACM SIGGRAPH 2009 Papers, New York, NY, USA, 2009, pages 58:1–58:10. https://doi.org/10.1145/1576246.1531364
[34] Pedrosa D.S., GALSYS - Procedural Creation of Trees: Through Combination of Genetic Algorithms and Lindenmayer Systems. 2014.
[35] Sims K., Artificial Evolution for Computer Graphics, presented at the Proceedings of the 18th Annual Conference on Computer Graphics and Interactive Techniques, New York, NY, USA, 1991, pages 319–328. https://doi.org/10.1145/122718.122752

[36] la Re de A., Abad F., Camahort E., Juan M.C., Tools for Procedural Generation of Plants in Virtual Scenes, in Computational Science – ICCS 2009, Volume 5545, Number 89, G. Allen, J. Nabrzyski, E. Seidel, G. D. van Albada, J. Dongarra, and P. M. A. Sloot, Eds. Berlin, Heidelberg, 2009, pages 801–810.
[37] Griffin C.C., Automated Vehicle Articulation and Animation: A Maxscript Approach, 2010.
[38] Go J., Vu T., Kuffner J.J., Autonomous Behaviors for Interactive Vehicle Animations, presented at the Proceedings of the 2004 ACM SIGGRAPH/Eurographics Symposium on Computer Animation, Aire-la-Ville, Switzerland, Switzerland, 2004, pages 9–18. https://doi.org/10.1145/1028523.1028525
[39] Smith G.M., Expressive Design Tools: Procedural Content Generation for Game Designers, 2012.
[40] Liapis A., Togelius J., Sentient Sketchbook: Computer-Aided Game Level Authoring, presented at the Proceedings of the 8th Conference on the Foundations of Digital Games, 2013, pages 213–220.
[41] Takagi H., Interactive evolutionary computation: fusion of the capabilities of EC optimization and human evaluation, Proceedings of the IEEE, Volume 89, Number 9, 2001. https://doi.org/10.1109/5.949485
[42] Merrell P.C., Model Synthesis, 2009.
[43] Harrison P.F., Image Texture Tools: Texture Synthesis, Texture Transfer, and Plausible Restoration, 2006.
[44] Fernández-Vara C., Thoughts on Procedural Content Generation, 29-Jun-2012. [Online]. Available: http://gambit.mit.edu/updates/2012/06/thoughts_on_procedural_content.php. [Accessed: 02-May-2017].
[45] Togelius J., Shaker N., Nelson M.J., Mixed-initiative Content Creation, in Procedural Content Generation in Games: A Textbook and an Overview of Current Research, N. Shaker, J. Togelius, and M. J. Nelson, Eds. 2015.