Using Cellular Automata as a Basis for Procedural Generation of Organic Cities

Melek B. Temuçin, İlker Kocabaş, and Kaya Oğuz

Abstract **— Procedural content generation (PCG) methods are commonly employed in computer games, simulations, and other related industries. While these methods are used for levels, terrains, stories and missions, their usage for procedural city generation is relatively rare because cities are heterogeneous structures with different components such as roads, layouts and buildings that depend on and affect each other. Additionally, ancient cities grew organically to areas that are safe and to those that provide food and water. This resulted in cities that do not have apparent regular patterns, such as rectangular building blocks. We propose an approach that uses cellular automata (CA) that generates clusters of areas. The CA is repeated for each cluster to hierarchically create different levels of the city. This procedure creates an organic city layout with fractal properties. The layout specifies the building blocks, main roads, and foliage. We also present a set of methods that can transform this layout into a three-dimensional model of the city. The results are promising; cities can be created in under a minute with minimal required input, and the resulting virtual city looks organic, rather than an algorithmic layout that has repeating patterns.**

Index Terms **— cellular automata, computer games, procedural city generation.**

I. Introduction

Procedural Content Generation (PCG) methods are chiefly used in computer games for algorithmic creation of content that ranges from maps and terrains to stories and missions. The field has always been actively studied, but has gained more attraction thanks to the abundance of games on several platforms such as gaming consoles and mobile devices [1]. It is also a popular technique in the movie industry where content such as crowds, foliage, and scenery are generated for computer generated imagery [2].

PCG is commonly applied as an offline tool where the content is generated before or as the game level is being loaded. Most of the time, the algorithms are used to generate the indoor or outdoor levels of a game level. Procedural city generation is relatively rare because they are one of the most complex structures the humanity has ever built, being the result of human behaviour and interaction as well as environmental factors. As a result, no two cities are the same. They are made up of heterogeneous structures with different components such as roads, layouts and buildings that depend on and affect each other, often requiring different algorithms to generate them. These make procedural city generation a challenging and interesting task.

The structure of cities has been studied for a long time for their planning and modelling urban land use [3,4], and later for modelling urban dynamics and development [5,6]. Historically newer cities are usually planned in advance, but ancient cities grew organically to areas that are safe and to those that provide food and water [7]. To achieve this organic look in a procedurally generated city, the algorithm should refrain from regular patterns. Among several studies that propose methods on generation of cities, only a few of them focus on the organic city generation. To provide an organic look, there should be randomness and structure at the same time. Cellular automata (CA) are known for their simple rule-based setup that can generate city patterns [8]. By incorporating randomness in the rule set, the patterns turn into real-life clusters.

We contribute to the literature by proposing a method that relies on the random initialization of the matrix and the rule sets of cellular automata that are run recursively on each of these clusters to generate organic layouts for cities. The recursive application of the same CA also results in fractal properties that provide a more natural and organic outcome for a city that is randomly generated. The resulting organic looking cities can be used in video games, however we believe that they would be more useful in applications such as training simulations. Our method uses the resulting organic layout to generate other city components, such as road networks, buildings and foliage.

The rest of the paper is organized as follows. The following section lists related work on procedural city generation and the application of CA to the cities. Section 3 discusses our proposed method. We conclude by the discussion based on the results of our method in Section 4.

II. Related Work

When the existing works for procedural city generators are studied, agent-based approaches [9]-[11] and L-systems [12], [13] are frequently used, although there are a high number of custom-made algorithms as well [14]-[19]. Most of the city generators aim to create the final state of the city, however some studies focus on the development of city through time [20], [21], how it is affected by human behaviour [11] or how practical the generated layout is [22].

Published on December 28, 2020.
M. B. Temuçin, International Computer Institute, Ege University, İzmir, Turkey.
(e-mail: busra.temucin@gmail.com)
İ. Kocabaş, International Computer Institute, Ege University, İzmir, Turkey.
(e-mail: ilker.kocabas@ege.edu.tr)
K. Oğuz, Faculty of Engineering, İzmir University of Economics, İzmir, Turkey.
(e-mail: kaya.oguz@ieu.edu.tr).

Since the proposed method relies on CA, we would like to focus on other approaches that make use of them. CA are mathematical models for complex natural systems containing large numbers of simple identical components with local interactions [23]. CA has been used for city planning for a long time [3], [4]. White and Engelen suggested a more detailed model that featured transition rules for each type of land which created urban layouts with fractal patterns [24]. Batty gave a few simple CA models to produce fractal patterns and claimed that planned cities can be generated by them [8]. However, he also emphasised that real cities, which are based on organic growth, are not geometrically ordered, and probability can be introduced for more variety.

Only a handful of works that approach urban dynamics through CA are procedural city generators. Kato et al. combined CA and genetic algorithms to generate a virtual city [25]. The city was modelled as a grid of cells, where each cell represented one of the seven states that included building and road types. They presented a set of building-layout and road-generation rules that governed the CA. The chromosomes in the genetic algorithm represented the time-series pattern of the changes in the city.

The approach of Honda et al. generated dynamically changing virtual cities [26]. The system consisted of three modules: road module, time module and layout module. Road module was based on L-systems; it created and updated road network based on the current state of the city and extracted the blocks between the roads. The time module consisted of two CA; area CA and block CA. Area CA took the areas as the cell structure. It implemented the global influences over an area, namely the land use type and overall building attributes such as high-rise and low-rise. Block CA developed the blocks under the areas according to the area constraints and helped the blocks blend with each other by interacting a block with its neighbours. This way they achieved smooth transition between areas.

Our approach introduces the application of CA directly for the layout of the city. The building foundations, main roads, and greenery areas are determined by this process. Our method then uses this layout for further processing to create a three dimensional view of the city.

III. Method

We propose a method that applies cellular automata recursively to generate organic layouts that will be the basis for a random city. City blocks, the bases for buildings and other areas, and the main roads are all defined within this layout, generated simultaneously via CA. Although this layout is the main focus of our method, we also provide methods to convert it to a 3D city model using several techniques.

Our method can be detailed in two major steps. The initial step is the layout generation; this layout is then passed to a post-processing step before the second step generates the 3D representation of the random city.

A. Layout generation using CA

The cellular automata in our method run on a matrix of size $M \times N$, which is randomly generated with integer values 0 or 1 that denotes empty and full cells, respectively. The probability of a cell being empty or full is equal. This results in a chaotic, noise-like pattern of empty and full cells. They are grouped together by the Grouping Cellular Automaton (GCA) that uses the 8-cell Moore neighbourhood. GCA rule set is given below.

IF an empty cell has more than three neighbours
THEN mark the cell as full
IF a full cell has less than five neighbours
THEN mark the cell as empty

The result of GCA is roughly shaped clusters that are far away from each other. The Smoothing Cellular Automaton (SCA) is used to smooth the rough and fractured edges of these clusters. The rule set for SCA is given below.

IF an empty cell has more than four neighbours
THEN mark the cell as full
IF a full cell has less than four neighbours
THEN mark the cell as empty

After being smoothed, clusters should be expanded without colliding with each other. To do so, the clusters are marked with a different identification number using a flood fill algorithm. When the flood fill algorithm returns, cells in the same cluster share the same identification number. The clusters are then expanded until a constant distance d is left between them. This distance is set to $d=2$ by default, which provides an adequate distance between clusters since these spaces will be set as main roads.

Just as the randomization, GCA, and SCA is run on the initial $M \times N$ matrix, we run them on each of the resulting clusters, creating sub-clusters with similar patterns. These steps are repeated one final time on the sub-clusters. The resulting clusters have fractal properties because of the repeating patterns on different levels, and have an organic look because of the randomisation and the cellular automata. The number of these recursive calls are controlled by the variable s, short for stage, and is set to $s=3$ in our trials (Fig. 1a, 1b, 1c).

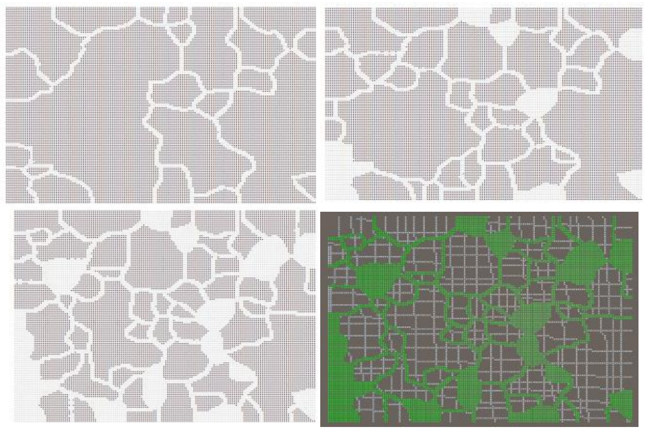
Fig. 1. Applying CA for three consecutive stages from left to right, and then finalizing by adding the streets.

Due to their nature, cellular automata run continually unless they reach a homogeneous final state. Our method requires CA that has a stopping condition. A halting threshold l variable is defined to stop both GCA and SCA. This variable depends on the size of the initial matrix, *M* and *N*, and *s*, the number of stages variable. Its definition is given in Equation (1).

$$l = \frac{M \times N}{10^{[log_{10}(M \times N)-1]}} \times 10^{s-1} \quad (1)$$

GCA should stop when the full cell clusters are formed. Forming of these clusters depends on the initial size of the matrix. It is observed that as the GCA runs, clusters emerge from the initial noise-like pattern, the noise decreases over time and is pushed to the edges of clusters. When the clusters are stabilized, the amount of noise falls under the threshold value *l* and GCA is stopped.

SCA should stop when there is no noise left on cluster edges. The number of changes made on the edge of the clusters is recorded for each step of SCA. When they are under the *l* value, SCA is stopped.

At the end of *s=3* runs of cellular automata, the spaces between clusters are set as the main roads. Some of these spaces are very large, and they are set as foliage, or recreational areas.

The streets within the clusters are handled by a post-processing step. They are one cell wide and mostly straight with some random skewness to add visual variety. This is necessary for creating realistic streets where buildings are usually aligned, making the streets usually straight.

A sample map is given in Figure 1d, where the green shows the main roads and foliage, darker shades represent the full cell clusters where buildings will be placed, and the whites shows the streets.

Real cities are not based on perfect grids. The resulting matrix is a low-resolution pixelated version of the map, and if it is used directly in representation of a 3D map, it will not be realistic. Our CA algorithms have created a layout that contains clusters of building blocks, streets, main roads and foliage areas, and different methods can be used to transfer this to a 3D model that can be used in different scenarios. For example, for a 2D game, the resulting matrix would be enough, and may require some smoothing on the streets. For a 3D game, the matrix can be converted to a graph, as will be done in the following subsections.

B. *Conversion to 3D Model*

The layout matrix consists of three types of cells; main roads, streets and buildings. Initially, graphs are generated for main roads and streets, and they are smoothed to remove the gridded look. During this process only road and street cells are taken into consideration to prevent any changes to building cells. Then, the building graph is created by dividing the building cell clusters with the most optimal way possible and assigning a node to each resulting building foundation. Finally, 3D models of roads and buildings are placed to generate a 3D view of the city.

The first step in road graph generation is to create the nodes and connect them with their neighbours. The conversion is one-to-one for street cells, since they are only one cell wide, but not for the main roads. This procedure follows the study by Sexton and Watson [27] but we had to include a series of adjustments, such as cycle elimination, removing isolated nodes, and connecting dead ends to other roads within some predefined range. The final procedure is the smoothing of the graphs. This is done by relocating each node to the average position of their neighbours. A sample is shown in Fig. 2.

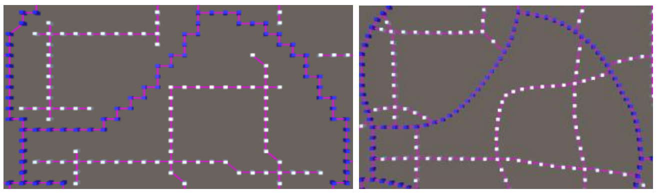
Fig. 2. The image on the left shows a jagged street, and the image on the right shows the street after the smoothing process.

To place 3D buildings, their positions and sizes have to be specified. We already know the approximate locations of our buildings with the help of the layout matrix. Similar to road graph generation, the first step is to convert them to graphs. However, the buildings require road access. To eliminate the uniform grid structure and properly align the buildings next to roads, the building cells are grouped before their conversion to building nodes. These cell groups, which we call building foundations, are generated by dividing the cell clusters via binary space partitioning (BSP) algorithm until all foundations are equal or smaller than cluster threshold, without forming a BSP tree. Cluster threshold depends on cluster size; larger clusters put out larger foundations.

Resulting foundations may contain non-rectangle cell arrangements which are not suitable for model placement. These are processed further until the largest rectangle is found. Fig. 3 shows the process before and after the foundations are laid.

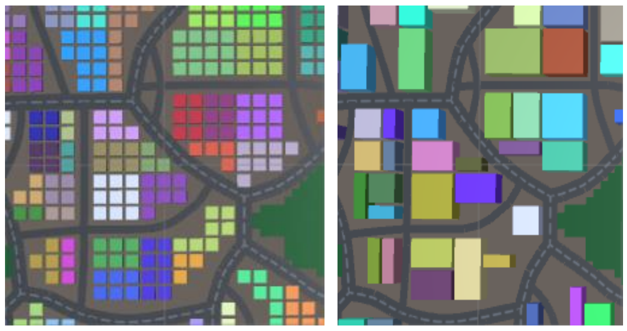
Fig. 3. The image on the left shows the foundations formed by the binary space partition. The image on the right shows the foundations after they are finalized.

The final step is to generate the 3D view of the city by placing road, building and foliage models. The road models are rectangles covered with textures and stretched between connected nodes. For the recreational areas tree models are placed on a grid layout. The building models are primitive cubes covered with textures. They are scaled slightly smaller than the foundation size to avoid intersection with adjacent buildings. The foundation size is used to determine the building height and type, differentiating between commercial and residential areas, which have different textures. Building textures and tree models are found in OpenGameArt site [28,29,30], while road textures are custom made. An example city model is shown in Fig. 4.

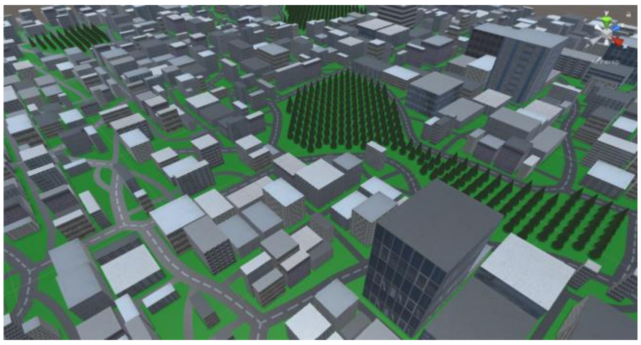

Fig. 4. The 3D model of the procedurally generated city.

IV. Results and Conclusion

The main contribution of this study is to provide a new method that can generate a basis for a city layout that does not have an algorithmically generated feel, while creating roads, buildings and foliage simultaneously as opposed to common methods in the literature.

The methods were realized on Unity3D [31] with C# as the scripting language. The complete process completes under a minute and depends on the size of the input M and N values. It could be used in games, simulations, or other media that use procedural city generation to provide more content.

As future work, there are several paths that can be followed. The most apparent direction is to place the city on a terrain that is defined by a height map. In this case, water bodies can be placed instead of parks, and streams can be placed along the roads. The notion of desirability can also be introduced to the cellular automata, as well. Another direction is to have a more realistic looking city by populating the city with elements such as pavements and traffic lights, as well as more types of buildings and land use.

References

[1] N. Shaker, J. Togelius, and M. J. Nelson, *Procedural Content Generation in Games: A Textbook and an Overview of Current Research*. Springer, 2016.

[2] M. Hendrikx, S. Meijer, J. Van Der Velden, and A. Iosup, "Procedural content generation for games: A survey," *ACM Transactions on Multimedia Computing, Communications and Applications*, vol. 9, pp. 1-24, Feb. 2013.

[3] W. R. Tobler, "Cellular Geography," in *Philosophy in Geography* (S. Gale and G. Olsson, eds.), pp. 379-386, Dordrecht: Springer Netherlands, 1979.

[4] H. Couclelis, "Cellular worlds: a framework for modeling micro—macro dynamics," *Environment and Planning A*, vol. 17, no. 5, pp. 585-596, 1985.

[5] M. Batty, Y. Xie, and Z. Sun, "Modeling urban dynamics through GIS-based cellular automata," *Computers, Environment and Urban Systems*, vol. 23, no. 3, pp. 205-233, 1999.

[6] L. Benguigui, D. Czamanski, and R. Roth, "Modeling cities in 3d: a cellular automaton approach," *Environment and Planning B: Planning and Design*, vol. 35, no. 3, pp. 413-430, 2008.

[7] A. Emilien, A. Bernhardt, A. Peytavie, M.-P. Cani, and E. Galin, "Procedural generation of villages on arbitrary terrains," *The Visual Computer*, vol. 28, p. 809–818, June 2012.

[8] M. Batty, "Cellular automata and urban form: a primer," *Journal of the American Planning Association*, vol. 63, no. 2, pp. 266-274, 1997.

[9] T. Lechner, B. Watson, and U. Wilensky, "Procedural city modeling," in *In 1st Midwestern Graphics Conference*, 2003.

[10] T. Lechner, P. Ren, B. Watson, C. Brozefski, and U. Wilenski, "Procedural Modeling of Urban Land Use," in *ACM SIGGRAPH 2006 Research Posters*, SIGGRAPH '06, (New York, NY, USA), ACM, 2006.

[11] C. A. Vanegas, D. G. Aliaga, B. Benes, and P. A. Waddell, "Interactive design of urban spaces using geometrical and behavioral modeling," in *ACM Transactions on Graphics (TOG)*, vol. 28, p. 111, ACM, 2009.

[12] N. Kato, T. Okuno, A. Okano, H. Kanoh, and S. Nishihara, "An ALife approach to modeling virtual cities," in *IEEE International Conference on Systems, Man, and Cybernetics*, vol. 2, pp. 1168-1173 vol.2, Oct. 1998.

[13] Y. I. Parish and P. Müller, "Procedural modeling of cities," in *Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques*, pp. 301-308, ACM, 2001.

[14] 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*, pp. 87-ff, ACM, 2003.

[15] S. Greuter, *Undiscovered worlds, real-time procedural generation of virtual three-dimensional spaces*. PhD thesis, RMIT University, Melbourne, Victoria, Australia, 2008.

[16] G. Kelly and H. McCabe, "Citygen: An interactive system for procedural city generation," in *Fifth International Conference on Game Design and Technology*, pp. 8-16, 2007.

[17] S. Groenewegen, R. M. Smelik, K. J. de Kraker, and R. Bidarra, "Procedural City Layout Generation Based on Urban Land Use Models," in *Eurographics (Short Papers)*, pp. 45-48, 2009.

[18] A. Emilien, A. Bernhardt, A. Peytavie, M.-P. Cani, and E. Galin, "Procedural generation of villages on arbitrary terrains," *The Visual Computer*, vol. 28, pp. 809-818, June 2012.

[19] R. Laycock, G. Ryder, and A. Day, "Automatic generation, texturing and population of a reflective real-time urban environment," *Computers & Graphics*, vol. 31, pp. 625-635, Aug. 2007.

[20] B. Weber, P. Müller, P. Wonka, and M. Gross, "Interactive Geometric Simulation of 4d Cities," *Computer Graphics Forum*, vol. 28, pp. 481-492, Apr. 2009.

[21] B. Williams and C. J. Headleand, "A Time-Line Approach for the Generation of Simulated Settlements," pp. 134-141, IEEE, Sept. 2017.

[22] O. Pueyo, A. Sabrià, X. Pueyo, G. Patow, and M. Wimmer, "Shrinking city layouts," *Computers & Graphics*, vol. 86, pp. 15-26, Feb. 2020.

[23] S. Wolfram, "Universality and Complexity in Cellular Automata," in *Theory and Applications of Cellular Automata*, vol. 1 of *Advanced Series on Complex Systems*, pp. 91-125, Singapore: World Scientific, 1986.

[24] R. White and G. Engelen, "Cellular automata and fractal urban form: a cellular modelling approach to the evolution of urban land-use patterns," *Environment and Planning A*, vol. 25, no. 8, pp. 1175-1199, 1993.

[25] N. Kato, T. Okuno, R. Suzuki, and H. Kanoh, "Modeling virtual cities based on interaction between cells," in *IEEE International Conference on Systems, Man, and Cybernetics*, vol. 1, pp. 143-148, IEEE, 2000.

[26] M. Honda, K. Mizuno, Y. Fukui, and S. Nishihara, "Generating autonomous time-varying virtual cities," in *International Conference on Cyberworlds*, pp. 45-52, Nov. 2004.

[27] C. Sexton and B. Watson, "Vectorization of Gridded Urban Land Use Data," in *Proceedings of the 2010 Workshop on Procedural Content Generation in Games*, PCGames '10, (New York, NY, USA), pp. 5:1-5:8, ACM, 2010.

[28] K. Sear, "Buildings - Batch of 21 Seamless Textures with normal maps," 2016.

[29] T. Speed, "Free Urban Textures: Buildings, Apartments, Shop Fronts," 2015.

[30] Mitylernal, "Low-poly Tree," 2016.

[31] "Unity3d," 2020.

Melek Büşra Temuçin was born in Izmir, Turkey in 1990. She has completed her BSc degree in Mathematics from Ege University, Izmir, in 2013. She has received her MSc degree in Information Technologies from Ege University, Izmir in 2020.
Her publications include "Procedural City Generation Using Cellular Automata" in Eurasiagraphics 2017 Conference Proceedings. The same paper was also published as a chapter in Contemporary Topics in Computer Graphics and Games (Bern, Switzerland: Peter Lang D, 2019). Her research interests include procedural content generation in games and game design.

Kaya Oğuz was born in Izmir, Turkey, in 1980. He has completed his BSc degree in Software Engineering from Izmir University of Economics (IUE), in 2007. He has received a scholarship from IUE to complete his MSc degree (with honors) in Computer Games Technology from University of Abertay Dundee, Scotland, UK in 2009. He has received his PhD in Information Technologies from Ege University, Izmir, in 2016. He is mostly interested in the application of machine learning algorithms to computer games.

He was a Research Assistant with the Izmir University of Economics, from 2009 to 2011. Then, he was a Lecturer with Ege University. He has taught several courses with the Software and Computer Engineering Department as a Guest Lecturer, Izmir University of Economics. In 2017, he has joined the Department of Computer Engineering, Izmir University of Economics, Izmir, Turkey, as an Assistant Professor. His research interests include procedural content generation for computer games, machine learning, and medical imaging analysis.

Dr. Oguz has been a member of IEEE and ACM since 2010.

İlker Kocabaş was born in Izmir, Turkey, in 1978. He has completed his BSc degree in Electrical & Electronics Engineering from Middle East Technical University (METU), in 2000. He has received his MSc and PhD in Information Technologies from Ege University, Izmir, in 2005 and 2011 respectively. He is mostly interested in the information retrieval area.

He was a Research Assistant with International Computer Institute, Ege University, from 2002 to 2013. He is currently assistant professor at the same institute. His research interests include information retrieval, natural language processing, machine learning, and data science.