Hindawi
Journal of Sensors
Volume 2022, Article ID 5629645, 13 pages
https://doi.org/10.1155/2022/5629645

Research Article

A Critical Evaluation of Procedural Content Generation Approaches for Digital Twins

Abdul Latif,[1] Megat F. Zuhairi,[1] Fazal Qudus Khan,[2] Princy Randhawa,[3] and Akshet Patel[3]

[1]*Malaysian Institute of Information Technology, Universiti Kuala Lumpur, Kuala Lumpur 59100, Malaysia*
[2]*Department of Computer Science, University of Swat, Shangla Campus, Alpurai, 19100 Shangla, KPK, Pakistan*
[3]*Department of Mechatronics Engineering, Manipal University Jaipur, Jaipur 303007, India*

Correspondence should be addressed to Fazal Qudus Khan; fazal.qadus@uswat.edu.pk

Received 4 May 2022; Revised 5 June 2022; Accepted 30 June 2022; Published 29 July 2022

Academic Editor: Rabeh Abbassi

Procedural content generation (PCG) of terrains is one of the main building blocks for creating an automated and real-time virtual world. This study provides and in-depth review of the different tools, algorithms, and engines used for the PCG of terrains for the PCG of digital worlds; we focus especially on terrain generation but address also the main issues related to build structures and characters. It is important to know that the PCG of terrains can be implemented in a multidisciplinary scenario, for instance, modeling of games, simulating industrial maps, urban and rural planning of developments, government, and nongovernmental agency improvement plans. Many problems in current approaches were identified from the literature, where most of the researches studied were merely throwaway prototyping based, i.e., proof of concept-based systems, and were not tested in the real environment. Also, genetic algorithm-based PCG of terrains lack multiple features such as roads and buildings. The virtual cities generated through different engines were lacking a realistic look and feel. Terrain generation through multiobjective evolutionary algorithms (MOEAs) is investigated, and it is deemed the usage was restricted to the gaming domain and not extended to other fields. Findings suggest that the correctness of generated terrain is a big issue in the automatic generation of terrains. Thus, a focus on the automated correctness check is required. Other important content generation in video games such as structure generation and character generation has been extensively studied, and the techniques are analysed in the further sections of this research work.

1. Introduction

A digital twin is a computer software that creates simulations based on real-world data to forecast how a product or process will perform. To improve the output, these applications may use the Internet of Things (Industry 4.0), artificial intelligence, and software analytics. These virtual models have become a standard in contemporary engineering to drive innovation and enhance performance, thanks to advancements in machine learning and elements such as big data [1]. In brief, having one may help improve strategic technological trends, avoid expensive breakdowns in physical objects, and test processes and services utilizing superior analytical, monitoring, and predictive capabilities.

Human society has evolved in terms of disruptive technologies for breaking the long-held business rules using business process reengineering (BPR) [1]. One of the most significant disruptive technologies used by software developers is designing and implementing virtual worlds and their simulation and modeling content. Electronic games consume a significant part of our daily time, especially in urban areas where the Internet is available and access to game-playing hardware and software is available. The World Economic Forum estimated the gaming industry's revenue

to be approximately USD 165 billion in 2020 [2]. It has been roughly 62 years since the computer gaming industry began during the late 1950s, with the first game's creation named "Tennis for Two" by physicist William Higinbotham [3]. Since then, electronic computer games have progressed, improved, and grown. Games are influenced by social norms and values, human views, and relationships, and most importantly, it has helped the game-playing community, content creators, and developers to understand the advancements in computer technologies. The 1970s saw the introduction of arcade games and gaming consoles. Personal computers were introduced in 1975, targeting niche markets. The gaming industry was still working on introducing new games and ideas for this technology's public use. In the 1990s, the personal computers were mass-produced, and hence, the home gaming consoles rapidly increase. The acceptance of home gaming came with the development of the first-person shooter game [4]. The continuous and rapid growth of the industry and incorporation of 3D content has made the video gaming industry phenomenal in size [5]. As such, the demand for exciting content design and generation has increased. The gaming industry's content creators and developers are coping with meeting the demands of these high-end users. These users are always eager to look for undiscovered exciting and challenging content within the minimal time possible and less cost incurred.

The technology advancements and high-end computational power availability have resulted in the game-playing community becoming very choosy and cautious in adopting a game. Video game developers and content creators always struggle with content production in terms of quantity and quality. Manual efforts and labours increase production costs and the time to deliver to the market. In game production, the content creators and developers use various tools to produce exciting content like Maya [6] and Unity 3D [7]. The game developers must create a virtual world for the game and create various types within that world along with interactive stories and playable levels. The whole process needs to be scalable [8] and cost-effective for the producers of games and the whole gaming community. Content generation for video games has been a critical concern for video game developer and producers. The journey from manual content creation in video game development to content creation through algorithms has been challenging. The researchers have been trying a mix of ideas from pure manual content creation to mix and automated content generation [9].

In video and electronic games, procedural content generation (PCG) creates the gaming world's content in an automated manner, using algorithms. The content can range from playing characters, nonplaying characters, storyline, and puzzles to the ecosystem and environment, which create the ambiance and feel [10] of the storyline being played. Thus, to generate many contents, the PCG mechanism is used. There can be several reasons and purposes for using such content generation methods. The approach can reduce the duration and cost of the content generation process [11], achieving the right time-to-market and affordable content. Also, exciting and unpredictable content to the game's users is aimed at increasing the fanbase and playtime. According to Freiknecht and Effelsberg's definition [12], "procedural content generation is the automatic creation of digital assets for games, simulations or movies based on pre-defined algorithms and patterns that require a minimal user input."

According to Merriam-Webster's [13] definition, terrain refers to a geographic area or a piece of land with physical features, which tend to erode. When it comes to electronic video games, simulations, or videos, then terrain or virtual terrain is considered virtually. A virtual world resembles an actual physical terrain or product of pure imagination and fantasy or even a mix of both. Virtual terrain modeling and generation are crucial in games and simulations and training applications [14]. The goal is to create a terrain that is as per the requirements with minimal effort and keeping the element of surprise and interest intact. Virtual terrain in gaming is a stage set where the story and its characters work together in a coherent and immersive manner to create an exciting experience. The gaming environment's terrain adds value to the story and increases the interest level of the players, and an example could be tropical or urban terrain.

Auto terrain generation is one of the critical factors that can significantly reduce the cost and efforts involved in developing a new electronic game. Cost and time are not the only factors that are of concern to the game developer and designers. It is also vital that the content is exciting and unpredictable to keep the audience and gaming users engaged. Apart from the electronic gaming domain, the interesting uncharted and unpredictable content generation is a critical factor in simulation and virtual training systems.

According to Hecker [15], "the game based on random generation will build the content based on the predefined elements that the developers hard coded" and "the game based on procedural generation which create original content itself for the player of explore or use." These can be used to generate elements of a game. For example, in the game "Binding of Isaac," random generation is used, whereas in the games like Minecraft and Terraria, procedural generation is used. The most important terminology in procedural generation is the noise, also known as the Perlin noise. Perlin noise is a procedural texture primitive, a kind of gradient noise used by visual effects artists to make computer images seem more realistic. Although the function seems to be pseudorandom, all its visual features are the same size. In terrain generation, the first step is to create a horizon with hills, valleys, and mountains. To create this, an offset is introduced which will create patterns to make the terrain more interesting. To shift this offset randomly, Perlin noise is used to introduce the variation in the terrain. The offset can be shifted up and down or left or right to create smoother and more realistic terrains like mountains, hills, and valleys. The same principle can be used in 3 dimensions to generate caves by using two sets of noise generators in two axes. The intersection of each point in space will be the new point that defines the height or depth in 3D space.

The gaming industry and other niches are also keen on utilizing modern technology trends to achieve their virtual and automated content generation [16]. For instance, the

tourism industry has explored the virtual tourism concept [17]. Virtual tourism might be the only method of sightseeing and tourism in the future in case there are pandemics such as COVID-19. Such an approach may apply to various domains where social gathering and mobility are restricted [18]. Urban planning and landscape designing are time-consuming and expertise-oriented jobs are critical for humanity's future. Also, top academic universities teach architecture design and urban planning [19] involving virtual terrains for the planning phase. The industry-leading solution providers in GIS and urban planning also provide solutions [20] that help the urban planner and architects. PCG can help them to rapidly build terrain plan with minimal effort while remaining focused on the desired goals. Virtual terrains are practical in mission-critical domains where the usage of existing tools and machines is not feasible. Therefore, pilots trained by using a simulator are comparatively cheaper as compared to being trained multiple times in actual scenario [21].

The virtual terrain is a crucial component of the virtual world. Such terrain is applicable for games or any other virtual world or any other applicable niches. According to the findings, the accuracy of created terrain is a major concern in the automated production of terrains. As a result, the automatic correctness check must be prioritised. Other key content generation methods in video games, such as structure creation and character generation, have been widely investigated, and the approaches are discussed in the following sections. Nonetheless, manual terrain generation is a time-consuming and tiresome job and requires enormous effort and cost. Therefore, automated terrain generation is needed when we need a reduction in time and cost to deliver. It can also help in desired terrain generation with more minor or no input requirements. This work will study the previous work done in the field of procedural terrain generation. The last section concludes the paper and future directions to the research in the same domain.

2. Terrain Generation

According to Yin and Zheng [22], the users can naturally place terrain features through the Parametrically Controlled Terrain Generation (PCTG) method. It can be achieved by making sketches and adjustable parameters that can easily be understood while controlling the terrains' generation process. It is pertinent to state about the importance of procedural generation of terrains that the terrain's synthesis is widely applied in simulation training, moviemaking, and computer games. Experimental results have shown that the use of terrains is very efficient [23]. The gaming industry is increasingly focusing on procedural content generation methods, according to Frade et al. [24]. Considering the high degree of unpredictability of the procedural algorithms employed in video games, the gamers are expected to perform several tests and simulations to obtain an effective procedural system.

2.1. Procedural Terrain Generation. Players in computer gaming can easily be modelled by using AI agents while at the same time producing gameplay data [25]. The agent employed is capable of shooting and moving different gaming scenarios. Moreover, the terrain developed is specific to game content and is being implemented in video games. Technically, the automated generation of terrains in computer gaming makes the development of computer games cheaper and thus reduces the designer's work. It also enriches the experience of the player by continuously offering new game content. The work employs a turn-based 2D shooting game and emphasizes designing a gameplay-driven terrain generation in Scorched Earth. The features of the game also entail trajectory projectiles for gravity management. After every turn, the player can estimate the shooting angle and the projectile power for firing. Players can also avoid hitting the terrain when they estimate the correct shooting angle. Players also have the possibility of moving the tanks so that they can position themselves while shooting.

Procedural generation is a way of producing data algorithmically rather than manually in computers, usually by combining human-generated materials and methods with computer-generated randomness and computing capacity. It is frequently used in computer graphics to produce textures and 3D models. It is used in video games to automatically generate enormous volumes of material. Smaller file sizes, more content, and unpredictability for less predictable gameplay are some of the benefits of procedural generation, depending on the implementation. A subset of media synthesis is procedural generation.

PCG is an important technique for computer game development that is likely to become even more important in the future, both offline, for making the game development process more efficient (designing content such as environments and animations now consumes a large portion of the development budget for most commercial games), and online, for increasing replay value, ad revenue, and other benefits.

According to [26], Procedural Content Generation for Games (PCG-G) is tough since the generator must produce material, adhere to the artist's limitations, and provide players with intriguing examples. The paper discusses a six-layer taxonomy of game content: bits, space, systems, scenarios, design, and derived, and highlights various potential research possibilities in PCG-G.

The paper [27] describes how customized levels for platform games may be produced automatically. By developing models that predicted player experience based on level design elements and playing behaviors. These models are built using preference learning and are based on questionnaires given to participants after they complete various stages.

The paper summarizes the procedural game level generation by developing

(1) more accurate models based on a much bigger data set

(2) a system for adjusting level design parameters to specified players and playing styles

(3) assessment of this adaptation mechanism using both algorithmic and human players

In the paper [28], a search-based procedural terrain generation is defined, and the aim was to investigate what can and cannot be achieved by the approaches that go by that name and highlight some of the field's key research problems. The paper further discussed the correlation between search-based and other different types of procedural content generation methods.

DTL (Dungeon Template Library) [29] is a software library tool used to generate rougelike and terrain generation based on procedural generation.

THREE.Terrain [30] is an engine working on the procedural generation method which is used for terrain generation using Three.js 3D graphics library.

We classify the work done in procedural terrain generation into mainly four categories based on the algorithms' nature. These categories are as follows:

(1) Genetic algorithm-based approaches

(2) Engine-based classification approaches

(3) Multiple objective-based approaches

(4) Other approaches

2.1.1. Genetic Algorithm-Based Approaches. Designers may generate terrains depending on their aesthetic sentiments or desired characteristics using the Genetic Terrain Programming approach, which is based on evolutionary design using genetic programming. This method develops Terrain Programs (TPs), which can produce a family of terrains—-different terrains with the same morphological properties. The Interactive Genetic Algorithm (IGA) used for terrain generation means those algorithms where human evaluation or fitness test is needed. A nonspecialist can use the IGA in the generation of rapid terrains [31]. Additionally, graphics engines can allow terrains to be specified using more than 800 floating-point parameters. However, the increasing number of floating-point parameters can overwhelm nonspecialist users. The design of an auto terrain generation system (TAGS) is discussed in this research. TAGS is specifically based on the implementation of the IGA and addresses complicated procedural technique issues. The proposed design implements user preference, thereby exploring the multidimensional parameter space for satisfying aesthetic and user intuition. A semiautomatic technique with computer-aided modeling and physics-based modeling can be an innovative work. The combination of these techniques will reduce the aspects of the user input and, at the same time, promote high-quality output. Walsh and Gade [31] suggested that TAGS has been proven merely as a proof of concept. A more controlled study with unbiased users, a large sample of users, and many features are needed to acquire accurate results. Many features are not included, such as roads and buildings in the terrain generation, which makes the generated terrains unrealistic.

Wheat et al. [32] introduced a PCG method for generating 2D game difficulty levels tailored to a particular gamer's skills. Dynamic difficulty adjustment (DDA) technique, based on genetic algorithm, helps attract a larger gamer group by generating numerous levels. The study emphasizes game content's procedural generation, precisely dynamic game level difficulty on 2D platformer games. The study is specific to the game developed and not generalized. Only four parameters were considered in the study, along with a small number of participant sample. The results are suggestively requiring further study and research.

2.1.2. Engine-Based Classification Approaches. Rhalibi et al. [33] applied procedural methods to the 3D game environment by producing dynamically different game levels with random and user-selected constraints. The work is deemed not to have been tested by an actual company for evaluation. A web application development in Java programming language was part of the study, and future incorporation of artificial intelligence agents is intended to arrange the game world.

Dynamically generating structures such as cities with varied building structures by using PCG is among the current gaming industry approaches, as discussed by Greuter et al. [34]. In combination with algorithms, pseudorandom numbers produce a variety of buildings and streets, creating an impression of a city. The cities, however, are not as realistic because most of them are office skyscraper-type buildings without houses, factories, and schools, i.e., small buildings. During the rendering process, the building occluded by others is not needed to be drawn.

2.1.3. Multiple Objective-Based Approaches. A procedural generation of maps using a real-time strategy (RTS) game was developed by Lara-Cabrera et al. [35]. Kohonen network was used for evaluation purposes. A multiobjective evolutionary algorithm (MOEA) was utilized to evolve maps and to categorize the distance to bad or good in the training set. User involvement is entirely focused on aesthetics only and not on the playability in the games. In other domains, e.g., urban planning or city designs, other factors are emphasized, i.e., aesthetics related to the urban planning or landscaping norms.

Component-based generation of terrain can be achieved by selecting features from available ones having specific attributes. Such technique has been achieved for software product line shown by Khan et al. [36]. Thus, a multiple-level user's defined objective can be used as a selection criterion for the terrain's components or features to generate it automatically for various purposes. The approach is not specifically for terrain generation instead for software product line generation. However, based on the feature's selection algorithms, terrains can be generated as per the user's set objectives.

2.1.4. Other Approaches. This section summarizes different terrain generation approaches which do not fall in the categories of previous sections. It is essential to generate the procedural contents dynamically [37]. Role-play game (RPG) is a popular genre for many gamers. The procedural content

generation (PCG) reduces the design efforts and the time for game development. The researchers in [37] employed a machine learning approach to predict the type of player using the Bayesian network and produce suitable game content but with low accuracy. One possible reason was that the generation occurred in a semiautomated manner. Also, there was no access to the source code of the study's target game, limiting achieving the desired accuracy. The study worked on predefined role-playing game (RPG) elements rather than randomly generated elements and components.

Game development is cumbersome and complex, where the development of terrain, environment, objects, and characters needs to happen [38]. De Carli et al. [38] stated different methods and techniques, i.e., classical and current about PCG, suitable for game development for producing terrains, coastlines, rivers, roads, and cities. Such a method may assist in reducing the cost of game development. This study also clarifies assistive and nonassistive methods for content generation and mainly focuses on cost reduction using nonassisted methods for content generation.

Roberts and Chen [39] introduced a novel machine learning PCG approach called "learning-based procedure content generation (LBPCG)," which can produce vigorous content for the gamers with a lesser interruption while the game is being played. However, user testing results were not rigorous to predict the effectiveness and satisfiability of the proposed framework. The research work predominantly focused on the computational efficiency in online content rather than the generation of quality content. Content categorization is based on the developer and their prior knowledge of the user's preferences. Also, the approach is not economical.

Luo et al. [40] proposed a data-driven scenario generation framework for game-based training. For scenario evaluation, the time of the event is necessary to be considered. Integration of simulation, training model for content evaluation of the game, and AI player modeling was the paradigm's primary goal. The generated model of scenarios does not include persistent objects within the generated virtual environment. As per this study, the framework only works on the set of commands and does not accept open inputs in the form of open text, for instance.

Quality assurance and effective control for the designers are the two significant challenges in PCG [41]. Super Mario Bros (SMB) was used in the case study to demonstrate the effectiveness of interaction between rule-based and learning-based methods for generating game levels. The merger was proved to be effective in producing quality and controllable game levels. Undesirable content is removed using rules known as constructive primitives (CP). The learning-based approach is prone to potential errors and hence needs to be further researched.

Since it is hard to match the accuracy and quality of a handcrafted design with the PCG, Shaker et al. [42] designed a fitness function for SMB levels via a grammatical evolution algorithm. Player experience has not been incorporated in the fitness evaluation mechanism. To ensure playability, the grammatical evolution generator produces levels with less density.

Smith et al. [43] worked on a launchpad to create an autonomous level generator based on rhythm and pace for 2D platformers. Two parameters, i.e., quality assurance and designer control, were introduced by the launchpad generator. The core focus was on performance; hence, the addition of complex components resulting in complex levels will slow down the performance.

Another game level PCG, based on generating character-sequence levels, was developed for "Angry Birds" by Jiang et al. [44]. Two methods, i.e., pattern-struct and a preset model, were used for constructing the "funny quote PCG" for different levels. The structures are evaluated based on a fitness function, which considers several important structural aspects, i.e., the number of pigs and blocks. There is a need for a mechanism to test the generated levels for playability and involve player experience and emotions in the generation process. Common problems in the procedural content generation of terrains are shown in Figure 1.

Training on short datasets, a lack of adequate data, parameter tweaking, and other issues plague the PG machine learning technique [9]. Most procedural content generation techniques (PCG) do not include assessment, and designers establish the goals. Automated generation, artificial intelligence-assisted design, maintenance, analysis, and data compression are some of the applications for PCGML. The suggested study focuses on autonomous generation, which combines the algorithms and the fitness function to develop game material without human input. Video games are a popular kind of multimedia that requires a wide range of machine learning techniques. In general, game design necessitates that the game's level be both playable and attractive [10]. Simultaneously, there is no universally accepted method for standardizing datasets and evaluating performance in game design difficulties.

The 3D nonassisted PCG method was developed for 3D-canyon scenes by De Carli et al. [45] based on noise-generated heightmaps. K-means is a supervised assistive method [46]. A nonassisted mean shift algorithm and image operations were used to portray a canyon design. However, improvements are needed to ensure that the generated design is realistic by incorporating credible textures, vegetation, and other objects.

Table 1 summarizes and tabulates the different algorithms and software used for generating terrains for game levels.

2.1.5. Procedural Terrain Generation Software's and Its Applications. Terrain generation and procedural terrain generation are in the market and industry for quite some time. There are various software and applications that exist to generate terrains in various formats. The generated terrains later can be exported into several compatible formats for the target. This study also investigates different generation software available in the market. However, there is insufficient scholarly data and information about most of the tools and applications. The information gathered is categorized according to the criteria of open source/closed source, 2D and 3D support, export or output, input controls as having graphical user interface or command line interface, and scripting. This

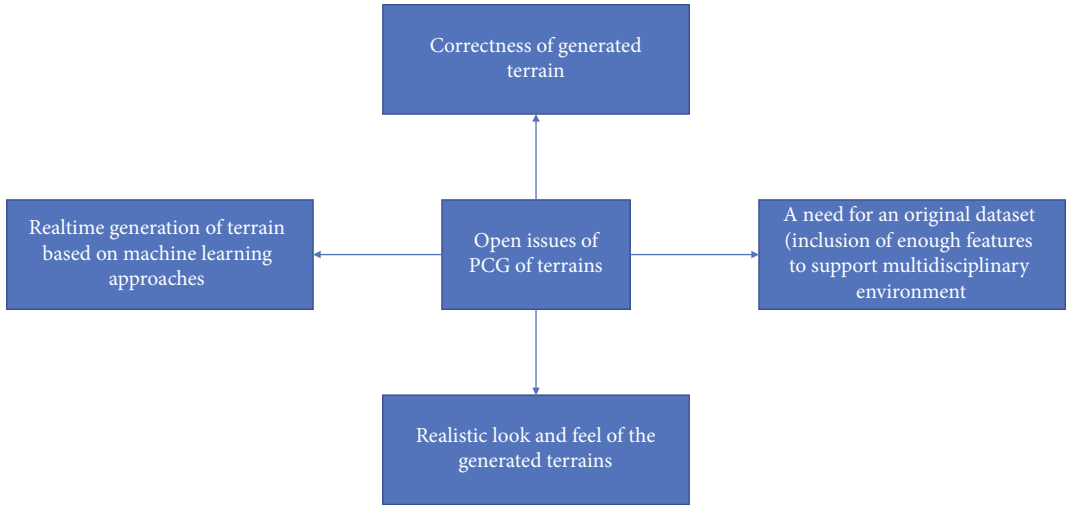

Figure 1: Common problems found in the PCG of terrains.

Table 1: Tabulated summary of the terrain generation algorithms.

Reference	Algorithm	Characteristics
[47]	Fractal noise, e.g., Perlin noise	Interpolation discontinuity of the second-order and nonoptimal gradient calculation
[48]	Midpoint displacement algorithm	The midpoint-displacement model for mountains now includes a squig-curve model of a river's path
[49]	Diamond-square algorithm	Terrain height is generated at random from four seed values organized in a grid of points, covering the entire plane in squares
[50]	Simple collision algorithm	Texturing and creating lifelike textures
[51]	Squarified treemap algorithm	In a two-dimensional region, show tree structures
[52]	Intent-driven partial-order causal link (IPOCL) planning algorithm	It solves the constraints of traditional causal dependency planners
[53]	Long short-term memory (LSTM)	Generation of simulated and original game levels and human level trajectories
[54]	Multidimensional Markov chains	The state space of a Markov chain that simulates the behavior of a system is frequently infinite in several dimensions
[53]	Autoencoder CNN algorithm	Heuristic approach for generation of terrains of all kinds

study also attempts to gather information about the underlying algorithms used in terrain generation within the applications.

Based on the study, the applications show that most procedural terrain generation tools are either standalone terrain generators and editors or part of a complex application, i.e., features as a plugin. In either case, the user requires prior knowledge of terrain generation and tweaking. Also, if the generated terrain is not as per requirements, the user may edit the content. Terrain generation in addition to manual verification for correctness is the key to such tools. This approach may be feasible for the domain experts but not for a user with limited knowledge about terrain generations. For instance, an urban planner that is not technically profound using terrain generation tools may expect less manual intervention with the tool concerning terrain editing. The urban planner's interest will be that a city environment is generated with correct measurements and necessary utilities and modalities with minimal or no corrections.

There exists a tool for terrain generation supported by different engines. There is information about open-source and closed-source software for terrain generation, 2D and 3D capabilities, export formats, the algorithms used, and the user control for the generation of the terrain, i.e., graphical user interface based or command-line interface. There is few open-source software that exists which can offer full support for 2D and 3D. Also, there is barely any information about the algorithms used (only a few share that info), while almost all have support for the graphical user interface to generate terrains.

Figure 2 shows the interface for the World Machine terrain generator. It provides manual controls and editing features for terrain generation and modification.

The interface for "Artifex Terra 3D" is shown in Figure 3, where it allows terrain editing in real time based on the heightmap. It also has texturing capabilities.

The EarthSculptor tool has an easy-to-use user interface, as shown in Figure 4. It is also a real-time heightmap editor

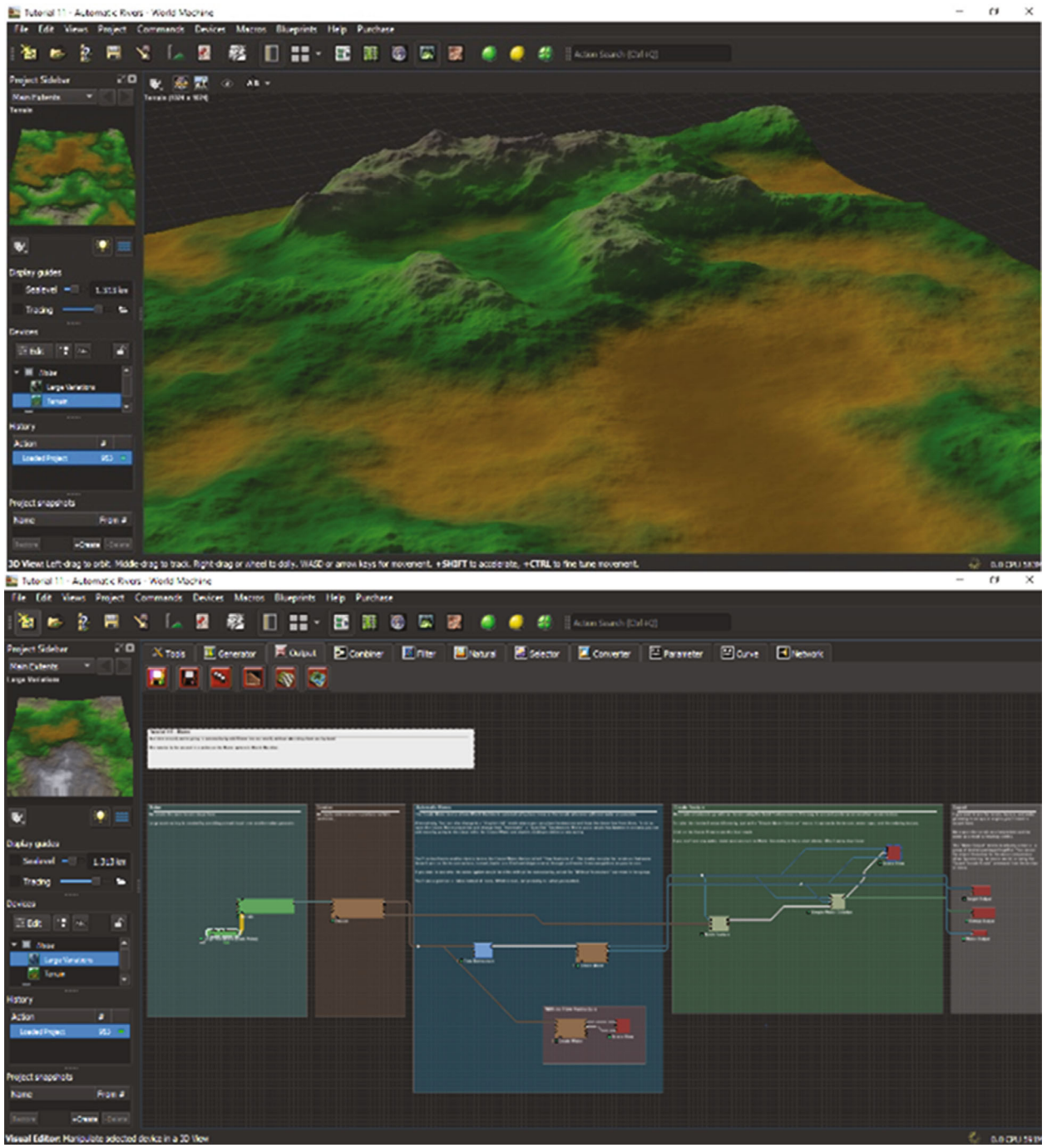

Figure 2: World Machine: terrain generator/editor.

that can generate terrains for various purposes in other target applications.

2.2. Random Terrain Generation. Random number generators generate a predictable, periodic series of pseudorandom numbers [55]. There are two kinds of random number generators:

(1) Computer software pseudorandom number generators (pseudorandom number generators) are software versions of random number generators. They do not create random numbers. Instead, they replicate actual randomness by simulating the selection of a value using algorithms

(2) Hardware random number generators (HRNG) are a kind of random number generator that uses a physical mechanism rather than an algorithm to generate random numbers. Thermal noise, photoelectric noise, diffraction grating noise, and other quantum

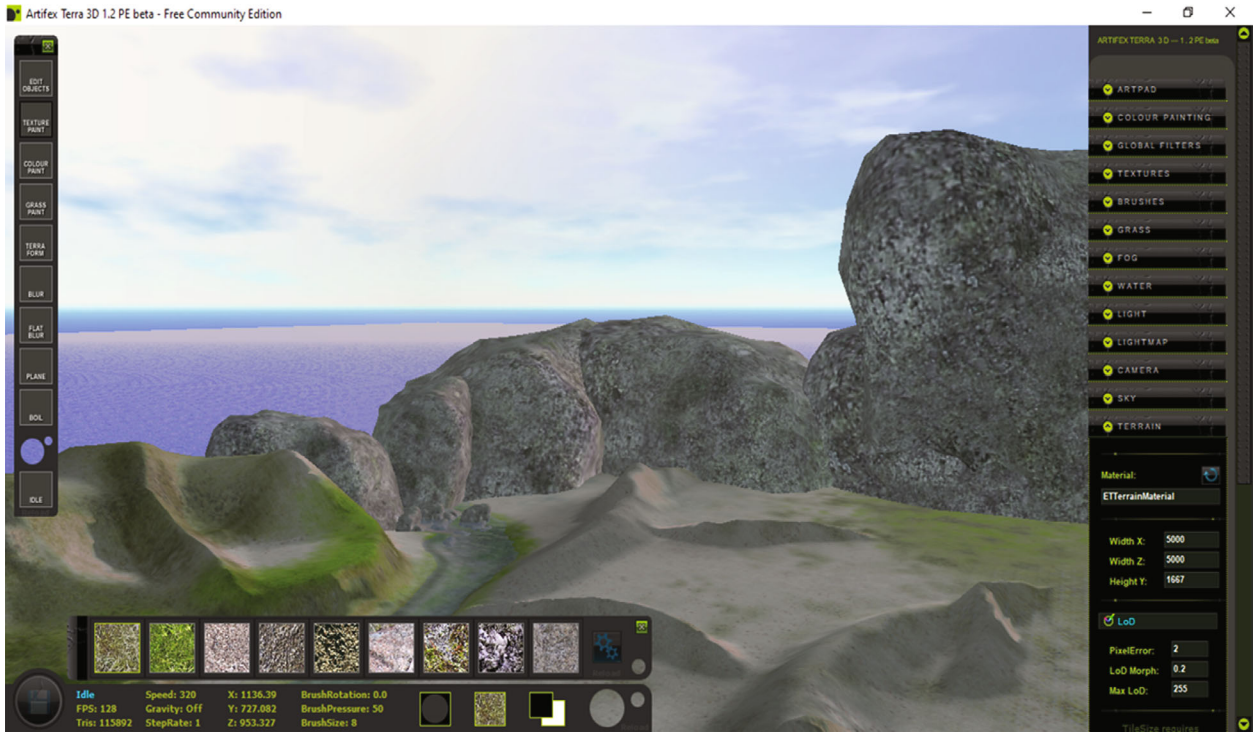

Figure 3: Artifex Terra 3D.

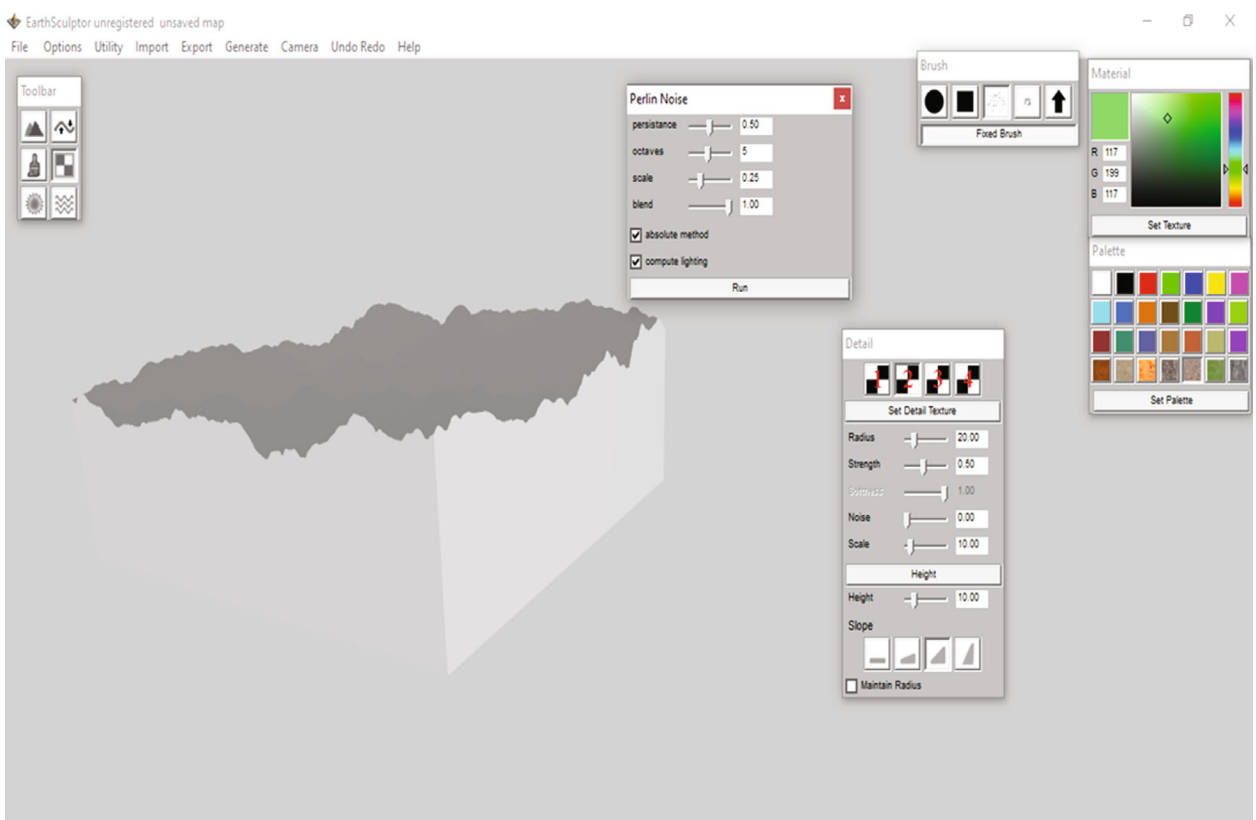

Figure 4: EarthSculptor.

processes are examples of tiny occurrences that produce low-level, significantly unpredictable "noise" signals

According to the paper [56], the uplift model, a novel and simple-to-implement terrain generator, is described in this study. It is based on the geology hypothesis of crustal deformations caused by uplifts. The eventual net impact of a series of uplifts on the Earth's surface is an average of the effects of each uplift at each location on the terrain. The application of this natural model results in a highly realistic-looking effect in the produced terrains. The model has six parameters that enable for a wide range of landscape types to be generated. Comparisons are performed with various terrain creation algorithms that are already in use. Averaging causes the surface to erode, whereas fractal surfaces are spiky and more suited to extraterrestrial planets.

In the paper [57], the suggested terrain model relies on a height-field representation that has been enhanced with materials and hardness data. The Poisson Faulting method, which was originally employed to create fractional Brownian surfaces, is the technique's starting point. The fault line generator's step function was changed with a circular one in the update. The resulting geometry was used to classify materials and calculate the hardness of the terrain model. The finished model was created by eroding geometry in accordance with the materials and their hardness data.

As per the paper [58], the goal of the authors is to apply the fractal dimension idea to assess the quality of raster digital elevation models (DEM). Most DEM evaluation approaches, which consist of calculating a global accuracy with respect to a reference, ignore artefacts generated by digitizing and sampling. They offered a method for computing fractal dimension based on the fractional Brownian motion model and utilized this model to highlight certain artefacts in digital elevation models.

Terrain generator [6] is a software based on random level terrain generation as a game that combines art and programming to create a natural-looking landscape. Slopes, soil, trees, cliffs, water, and other natural features may all be found in terrain. Stamping and pen techniques are frequently used to make them, and they are classified as games, simulations, and art. Simple 2D textured blocks to sophisticated 3D terrains are all possible.

3. Structure Generations

With the advancement in the gaming industry and the need for large quantity of content, the two most crucial parts of organic generations are as follows:

(1) Building generators

(2) Structure generators

The most popular procedural content generation (PCG) application is the generation of structures, cities, and buildings. "Citygen" [59] is a city generator system which generates the modern city's urban geometry. The system can develop the entire structure and system for roads and footprints of buildings with an input of just the terrain model. This system can then be used to place the buildings automatically or manually. There are several city generation techniques, they are as follows:

(1) Geometric rule-based systems

(2) L-systems

(3) Agent-based systems

(4) Template-based systems

(5) Split grammars

The methods described above show how to create collections of structures or even entire towns, but there have also been attempts to construct building interior "floor plans." The project, LaHave House, uses shape grammars to produce floor layouts. Based on player movement and positioning, a system that produces office building rooms in real time was created. Martin [60] investigated a graph-based technique in which rooms are treated as nodes and connections between rooms are treated as edges, with user-defined limitations such as room count and room function.

PCG approach has been effectively applied to generate numerous game contents, i.e., textures, geometric models, animations, and even sound clips [61]. PCG techniques for terrain elevation, water elements, vegetation, road networks, and urban environments are discussed in detail by Smelik et al. [61].

Tutenel and Bidarra [62] created a rule-based system that uses a semantic database to identify room kinds, and entities can form associations with their near neighbours. The concepts in this research were greatly influenced by Tutenel et al. [63], who tested a restricted rectangular L-growth algorithm that generated fully connected rooms for buildings. In a rectangular-growth-based technique, Camozzato et al. [64] procedurally create floor layouts using a hand-drawn building façade as input. Guo and Li [65] developed a system that creates multilevel structure floorplans using a combined method of agent-based search and optimization.

The methods to generate buildings can be bifurcated into two parts:

(1) Floor plan generation

A basic restricted growth algorithm is used to produce the floor layout. The original inspiration for utilizing this as a floor plan generator came from Tutenel et al.'s [63] L-shaped restricted growth technique. Unlike their technique, this one does not expand in a rectangle pattern, instead using a single block as the natural granular unit in Minecraft.

(2) External wall generation

The system then begins the process of constructing the external walls after the floor design has been produced. This is accomplished by employing a technique known as Cellular Automata (CA), which is derived from a family of

algorithms known as Cellular Automata (CA). CA is employed in this system to self-organize the placement of solid blocks and glass panes, resulting in unique structural external walls. This system employs neighbour summation, which means it is less concerned with neighbour states and more concerned with the total of those states. If the block is a window, it is a 1, and if it is a solid block, it is a 0. Because each block has four neighbours in addition to itself, the sum of the states can be anything between 0 and 5. At wall start-up, a matrix representing the building's height at width/depth (depending on which wall is being created) is generated at random, with solid blocks accounting for 75% of the wall and glass blocks accounting for 25%.

4. Character Generation

Both the characters and the setting in video games are vital in establishing a feeling of immersion for the player. Although each figure may be created by hand to make them seem more lifelike, filling a gaming world with more than 100000 individuals makes the process intrinsically complicated. In such circumstances, developers must consider adding new technologies to automate and speed up their development cycle. The procedural element of PCG is where the knowledge of how to make something (the generative part) is put into action to generate the desired sort of content [55]. That implies there is a need to build up a system that can be given a certain set of instructions, ideally parametrizable, and will return a specific outcome. Using a modified style-based generative adversarial network (StyleGAN), a form of neural network, a novel approach for procedurally generating Nonplayable Characters (NPCs) in video games has been offered. The most cost-effective way for gaming content creation is PCGML, which stands for Procedural Content Generation with Machine Learning. It is used to cut production effort and conserve storage space. Their method combines PCGML with StyleGAN to create NPCs that were distinct in both look and behavior. Character creation is influenced by the attributes or features, resulting in a diversified and engaging gaming environment for the players.

Developers utilize the DCG to overcome a major issue in the video game development cycle: time. These algorithms are used by game developers to shorten the time it takes to produce a game or aspects of it. Real-time strategy (RTS) is a video game genre in which players must take control of safe regions of a map and remove their opponents' assets (e.g., towers, buildings, and shelters). The Multiplayer Online Battle Arena (MOBA) is a subgenre of this genre that enables two teams to battle against each other online. In general, maps are static, and the amount of material accessible to players is continually increasing as the game grows. Due to design challenges caused by the newly additional content, game designers must spend an increasing amount of effort while creating these games. In addition, the topography of MOBA game levels is built on heightmaps, which are well-suited to the use of generative algorithms [28].

The role-playing game (RPG) format is used to create characters in MOBA games. This approach determines a character's strengths and weaknesses based on a set of values. They have a level value that is a direct representation of how the learning process works, in which a character may acquire experience points by doing certain activities. Then, when they achieve a particular number of points, their character is promoted to a higher level, thereby making it stronger, resulting in a more experienced person.

There are also a set of qualities that specify things like hit points (the amount of damage a character can take before dying), damage resistance, and magical damage. These characteristics determine how effectively a character succeeds in various scenarios. Characters are divided into classes based on their level and qualities. Classes specify the collection of abilities that a character may have [29]. These are unique powers that a character may utilize to help them in battle and accomplish various special acts.

5. Role of Procedural Generation in Digital Twins

We may want to populate the digital twin with organic or inorganic matter. This is where procedural generation would help. The information from the sensors is unaltered whereas the surroundings, buildings, avatars, and extra objects can be generated using the procedural generation techniques as mentioned in this research [28]. Procedural generation is a means of producing data algorithmically rather than manually in computers, usually by combining human-generated materials and methods with computer-generated randomness and processing capacity. It is extensively used in computer graphics to produce textures and 3D models [29].

6. Conclusion

Based on the investigative literature findings, it is deemed that plenty of work has been done in the terrain generation domain. However, many research types remain open and offer opportunities for improvement in PCG of terrains for modeling and simulations. The PCG of terrains can be implemented in a multidisciplinary scenario for modeling games, industrials maps, urban and rural planning of developments, government, and company improvement plans. The critical finding identified through the literature is that many research studies are merely proof of concept and were not tested naturally. It was found out that genetic algorithm-based PCG of terrains lack multiple features such as roads and buildings. The cities generated through different engines were lacking a realistic look and feel. Terrain generation using MOEA is also analysed, where the application is restricted to the gaming domain and not extended to other fields. The correctness of generated terrain is also a significant challenge in the automatic generation of terrains; thus, a focus on the automated correctness check is essential. The modern user's current technological and aesthetic requirements demand fast, exciting, and quality content generation. The content, which can be playing on nonplaying content, must be usable. Hence, procedural generation methods are the best candidates for terrain and game content generation, quick, exciting, and usable. This research study has

highlighted the different genres in the domain ranging from terrain generation, level generation to complete city, and virtual world generation. Therefore, it is recommended that further research work is necessary to evaluate the autogenerated terrains on aesthetics of users, e.g., of different ages, cultures, and preferences. At the same time, the validation of the generated terrain must also be verified for correctness. Indoor localization sensor combinations that include the software necessary for sensor data fusion in addition to the hardware should be employed. Processing systems, scenario-live-simulations, and digital shop floor management result in a procedural mix that must be followed. The capacity to constantly give all subsystems with the most up-to-date status of all essential information, procedures, and algorithms is critical to the digital twin.

Data Availability

All the data are included within the article, and no external data were used to support this study.

Conflicts of Interest

The authors declare that there is no conflict of interest regarding the publication of this paper.

References

[1] J. A. Hoffer, J. F. George, and J. S. Valacich, *Modern Systems Analysis and Design*, Pearson, Boston, Mass, 2014.

[2] The history of the gaming industry in one chart, *World Economic Forum*February 2021, https://www.weforum.org/agenda/2020/11/gaming-games-consels-xbox-play-station-fun/.

[3] "BNL | History: the first video game?," September 2020, https://www.bnl.gov/about/history/firstvideo.php.

[4] "10.2 The evolution of electronic games | Media and Culture," September 2020, https://courses.lumenlearning.com/suny-massmedia/chapter/10-2-the-evolution-of-electronic-games/.

[5] "The video games industry is bigger than Hollywood," Sepember 2020, https://www.myboosting.gg/blog/esports-news/the-video-games-industry-is-bigger-than-hollywood.

[6] "Maya Software | Computer Animation & Modeling Software | Autodesk," September 2020, https://www.autodesk.ae/products/maya/overview.

[7] Unity Technologies, "Unity real-time development platform | 3D, 2D VR & AR Engine," September 2020, https://unity.com/.

[8] A. Iosup, "POGGI: generating puzzle instances for online games on grid infrastructures," *Concurrency and Computation: Practice and Experience*, vol. 23, no. 2, pp. 158–171, 2011.

[9] J. Togelius, E. Kastbjerg, D. Schedl, and G. N. Yannakakis, "What is procedural content generation? Mario on the borderline," in *Proceedings of the 2nd International Workshop on Procedural Content Generation in Games*, pp. 1–6, New York, NY, USA, 2011.

[10] G. Tsaramirsis, M. Papoutsidakis, M. Derbali, F. Q. Khan, and F. Michailidis, "Towards smart gaming olfactory displays," *Sensors*, vol. 20, no. 4, 2020.

[11] M. Andreas and B. Esben, *Procedural character generation: implementing reference fitting and principal components analysis*, Aalborg University, 2007.

[12] J. Freiknecht and W. Effelsberg, "A survey on the procedural generation of virtual worlds," *Multimodal Technologies and Interaction*, vol. 1, no. 4, p. 4, 2017.

[13] "Dictionary by Merriam-Webster: America's most-trusted online dictionary," September 2020, https://www.merriam-webster.com/.

[14] "Declarative terrain modeling for military training games," September 2020, https://www.hindawi.com/journals/ijcgt/2010/360458/.

[15] C. Hecker, "Let's get to the floating point," *Game Developer Magazine*, pp. 19–23, 1996.

[16] G. Tsaramirsis, S. M. Buhari, K. O. Al-Shammari, S. Ghazi, M. S. Nazmudeen, and K. Tsaramirsis, "Towards simulation of the classroom learning experience: virtual reality approach," in *2016 3rd International Conference on Computing for Sustainable Global Development (INDIACom)*, pp. 1343–1346, New Delhi, India, 2016.

[17] "Virtual garden walkthrough," September 2020, https://www.jfp.co.jp/walk/index_e.htm.

[18] F. Q. Khan, S. Rasheed, and M. Ashraf, "Investigating the use of 3D mobile games for teaching ethics & basics to children," *Journal of Information Communication Technologies and Robotic Applications*, vol. 14, no. 4, pp. 65–75, 2019.

[19] Projects, "Harvard Graduate School of Design," September 2020, https://www.gsd.harvard.edu/projects/.

[20] "Advanced 3D City Design Software | ArcGIS CityEngine," September 2020, https://www.esri.com/en-us/arcgis/products/arcgis-cityengine/overview/.

[21] K. Valentino, K. Christian, and E. Joelianto, "Virtual reality flight simulator," *Internetworking Indonesia Journal*, vol. 9, no. 1, 2017.

[22] H. Yin and C. Zheng, "A parametrically controlled terrain generation method," in *2012 7th International Conference on Computer Science Education (ICCSE)*, pp. 775–778, Melbourne, VIC, Australia, 2012.

[23] R. Lara-Cabrera, C. Cotta, and A. Fernández-Leiva, "Procedural map generation for a RTS game," in *13th International GAME-ON Conference on Intelligent Games and Simulation*, Malaga (Spain), 2012.

[24] M. Frade, F. F. de Vega, and C. Cotta, "Aesthetic Terrain Programs database for creativity assessment," in *2012 IEEE Conference on Computational Intelligence and Games (CIG)*, pp. 350–354, Granada, Spain, 2012.

[25] Q. Yu and R. Crawfis, "Gameplay-driven terrain generation in Scorched Earth," in *2015 Computer Games: AI, Animation, Mobile, Multimedia, Educational and Serious Games (CGAMES)*, pp. 126–130, Louisville, KY, USA, 2015.

[26] M. Hendrikx, S. Meijer, J. Van Der Velden, and A. Iosup, "Procedural content generation for games," *ACM Transactions on Multimedia Computing, Communications, and Applications*, vol. 9, no. 1, pp. 1–22, 2013.

[27] N. Shaker, G. Yannakakis, and J. Togelius, "Towards automatic personalized content generation for platform games," in *Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, pp. 63–68, Washington, USA, 2010.

[28] J. Togelius, G. N. Yannakakis, K. O. Stanley, and C. Browne, "Search-based procedural content generation," in *European*

Conference on the Applications of Evolutionary Computation, vol. 6024, pp. 141–150, Berlin, Heidelberg, 2010.

[29] "Dungeontemplatelibrary -? Dungeon free resources (terrain & roguelike generation) - (DungeonTemplateLibrary)," September 2021, https://opensourcelibs.com/lib/dungeontemplatelibrary.

[30] "THREE.Terrain - a procedural terrain generation engine for use with the Three.js 3D graphics library for the web. - (THREE.Terrain)," September 2021, https://opensourcelibs.com/lib/three.terrain.

[31] P. Walsh and P. Gade, "Terrain generation using an Interactive Genetic Algorithm," in *IEEE Congress on Evolutionary Computation*, pp. 1–7, Barcelona, Spain, 2010.

[32] D. Wheat, M. Masek, C. P. Lam, and P. Hingston, "Dynamic difficulty adjustment in 2D platformers through agent-based procedural level generation," in *2015 IEEE International Conference on Systems, Man, and Cybernetics*, pp. 2778–2785, Hong Kong, China, 2015.

[33] A. E. Rhalibi, S. Cooper, C. Carter, M. Merabti, and J. Wetherall, "Web-based hardware accelerated procedural content generation," in *2011 IEEE Consumer Communications and Networking Conference (CCNC)*, pp. 368–372, Las Vegas, NV, USA, 2011.

[34] 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 Southeast Asia*, New York, NY, USA, 2003.

[35] R. Lara-Cabrera, C. Cotta, and A. J. Fernández-Leiva, "A self-adaptive evolutionary approach to the evolution of aesthetic maps for a RTS game," in *2014 IEEE Congress on Evolutionary Computation (CEC)*, pp. 298–304, Beijing, China, 2014.

[36] F. Q. Khan, S. Musa, G. Tsaramirsis, and S. M. Buhari, "SPL features quantification and selection based on multiple multi-level objectives," *Applied Sciences*, vol. 9, no. 11, 2019.

[37] Y. Lee and S. Cho, "Context-aware petri net for dynamic procedural content generation in role-playing game," *IEEE Computational Intelligence Magazine*, vol. 6, no. 2, pp. 16–25, 2011.

[38] D. M. De Carli, F. Bevilacqua, C. T. Pozzer, and M. C. d'Ornellas, "A survey of procedural content generation techniques suitable to game development," in *2011 Brazilian symposium on games and digital entertainment*, pp. 26–35, Salvador, Brazil, 2011.

[39] J. Roberts and K. Chen, "Learning-based procedural content generation," *IEEE Transactions on Computational Intelligence in AI and Games*, vol. 7, no. 1, pp. 88–101, 2015.

[40] L. Luo, H. Yin, W. Cai, J. Zhong, and M. Lees, "Design and evaluation of a data-driven scenario generation framework for game-based training," *IEEE Transactions on Computational Intelligence in AI and Games*, vol. 9, no. 3, pp. 213–226, 2017.

[41] P. Shi and K. Chen, "Online level generation in Super Mario Bros via learning constructive primitives," in *2016 IEEE Conference on Computational Intelligence and Games (CIG)*, pp. 1–8, Santorini, Greece, 2016.

[42] N. Shaker, M. Nicolau, G. N. Yannakakis, J. Togelius, and M. O'Neill, "Evolving levels for Super Mario Bros using grammatical evolution," in *2012 IEEE Conference on Computational Intelligence and Games (CIG)*, pp. 304–311, Granada, Spain, 2012.

[43] G. Smith, J. Whitehead, M. Mateas, M. Treanor, J. March, and M. Cha, "Launchpad: a rhythm-based level generator for 2-D platformers," *IEEE Transactions on Computational Intelligence in AI and Games*, vol. 3, no. 1, pp. 1–16, 2011.

[44] Y. Jiang, T. Harada, and R. Thawonmas, "Procedural generation of angry birds' fun levels using pattern-struct and preset-model," in *2017 IEEE Conference on Computational Intelligence and Games (CIG)*, pp. 154–161, New York, NY, USA, 2017.

[45] D. M. De Carli, C. T. Pozzer, F. Bevilacqua, and V. Schetinger, "Procedural generation of 3D canyons," in *2014 27th SIBGRAPI conference on graphics, patterns and images*, pp. 103–110, Rio de Janeiro, Brazil, 2014.

[46] "MacQueen: some methods for classification and analysis of multivariate observations," May 2019, https://projecteuclid.org/euclid.bsmsp/1200512992.

[47] K. Perlin, "Improving noise," *ACM Transactions on Graphics*, vol. 21, no. 3, pp. 681-682, 2002.

[48] P. Prusinkiewicz and M. Hammel, "Fractal model of mountains with rivers," *Graphics Interface*, vol. 93, pp. 174–180, 1993.

[49] G. S. P. Miller, "The definition and rendering of terrain maps," in *Proceedings of the 13th annual conference on Computer graphics and interactive techniques*, vol. 20no. 4, pp. 39–48, Dallas, USA, 1986.

[50] "Loopix-Project," September 2021, https://www.loopix-project.com/.

[51] M. Bruls, K. Huizing, and J. J. van Wijk, "Squarified treemaps," in *Data visualization 2000*, pp. 33–42, Springer, Vienna, 2000.

[52] M. O. Riedl and R. M. Young, "Narrative planning: balancing plot and character," *Journal of Artificial Intelligence Research*, vol. 39, pp. 217–268, 2010.

[53] N. A. Barriga, "A short introduction to procedural content generation algorithms for videogames," *International Journal on Artificial Intelligence Tools*, vol. 28, no. 2, p. 1930001, 2019.

[54] W. Ching, S. Zhang, and M. Ng, "On multi-dimensional Markov chain models," *Pacific Journal of Optimization*, vol. 3, pp. 235–243, 2007.

[55] T. Xu, I. D. Moore, and J. C. Gallant, "Fractals, fractal dimensions and landscapes—a review," *Geomorphology*, vol. 8, no. 4, pp. 245–262, 1993.

[56] J. R. Rankin, "The uplift model terrain generator," *International Journal of Computer Graphics & Animation*, vol. 5, no. 2, pp. 1–8, 2015.

[57] K. K. Warszawski, S. S. Nikiel, and M. Mrugalski, "Procedural method for fast table mountains modelling in virtual environments," *Applied Sciences*, vol. 9, no. 11, p. 2352, 2019.

[58] L. Polidori, J. Chorowicz, and R. Guillande, "Description of terrain as a fractal surface, and application to digital elevation model quality assessment," *Photogrammetric Engineering and Remote Sensing*, vol. 57, no. 10, pp. 1329–1332, 1991.

[59] 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, London, UK, 2007.

[60] J. Martin, "Procedural house generation: a method for dynamically generating floor plans," *Symposium on Interactive 3D Graphics and Games*, 2005.

[61] R. M. Smelik, K. J. de Kraker, S. A. Groenewegen, T. Tutenel, and R. Bidarra, "A survey of procedural methods for terrain modelling," in *Proceedings of the CASA Workshop on 3D*

Advanced Media In Gaming And Simulation (3AMIGAS), p. 10, Amsterdam, Netherlands, 2009.

[62] T. Tutenel and R. Bidarra, "Rule-based layout solving and its application to procedural interior generation," *CASA workshop on 3D advanced media in gaming and simulation*, 2009.

[63] T. Tutenel, K. J. De Kraker, and R. Bidarra, *A constrained growth method for procedural floor plan generation*, 2010.

[64] D. Camozzato, L. Dihl, I. Silveira, F. Marson, and S. R. Musse, "Procedural floor plan generation from building sketches," *The Visual Computer*, vol. 31, no. 6-8, pp. 753–763, 2015.

[65] Z. Guo and B. Li, "Evolutionary approach for spatial architecture layout design enhanced by an agent-based topology finding system," *Frontiers of Architectural Research*, vol. 6, no. 1, pp. 53–62, 2017.