Procedural Generation of Environments

***Abstract*—Procedural Generation is a computational technique used to create content during runtime, by combining computer algorithms with human-defined rules and objects, which are placed randomly in a virtual environment. This approach reduces the manual content creation and produces less repetitive results.**

This method has been extremely important for the evolution of visual computing and computer graphics, especially in the 3D digital games. In this field, this technique is used to automatically create large amounts of content in a game, in a way that creates immersive experiences and/or environments and also minimizes repetitive game patterns

This article will explore the creation of digital environments using procedural generation, the algorithms it uses and its impact on visual computing . In addition, it will discuss the advantages and disadvantages of this technique in relation to the manual creation of environments in virtual worlds.

***Index Terms*—Procedural Generation of Environments, Visual Computing, Noise Algorithms, L-Systems**

I. Introduction

THE creation of virtual environments, has a crucial role in Visual Computing, because these environments enable the development of iterative application such as digital game, virtual simulations and computer vision systems. These environments not only helps the user engagement but also provide realistic representations of real world scenarios, creating a safe space for testing and improving the user learning and information retention.

The manual process of placing models in order to create virtual environments can be a tedious and often results in repetitive patterns. To avoid these problems, developers have considered to use a technique that could generate these environments dynamically, simulating how natural structures are created in the real world, during runtime. This technique is called Procedural Generation.

The term procedural refers to a process that computes a particular function. Fractals are geometric shapes containing detailed structure at arbitrarily small scales, which can often be generated procedurally. So developers could use computational algorithms to procedurally generate fractals, which could have detailed meshes or models during runtime. An example of procedural generation is the creation of terrains for digital games,such as **No Man's Sky** (2016). This game is a science fiction exploration and survival game developed by the company **Hello Games** out of Guildford, UK. The player's objective is to navigate in the space in a ship, to explore every planet, the planets and their environments are created using procedural generation.

Fig. 1. No Man's Sky Environment.

II. Evolution of Procedural Generation of Environments

A. Early History

Procedural Generation of Environments started to appear in early video games. Roguelike games such as **Beneath Apple Manor** (1978) and **Rogue** (1980), a genre inspired by the classical tabletop game **Dungeon Dragons**, started to introduce procedural generation in order to randomly produce dungeons. These dungeons were created in ASCII or regular tile-based systems and define rooms, monsters to challenge the player. In the same year **Beneath Apple Mano** was released a maze game called **Maze Craze** used an algorithm to generate a random, top-down maze for each level. These methods to create random level maps, changed the way of developers thought, because they saw a new way to create game world while maintaining the complexity of the gameplay and spending less time creating them manually.

Later digital game started to use pseudo random number generates(PRNG) to create new game maps. PNRGs were often used with a predefined seed values in order to generate very large game worlds. Examples of these video games are **Sentinel** (1986), which had 10 000 different levels stored in only 64 kilobytes and a particular digital game **Elite** (1984). This game was supposed to have a total of 2^{48} (approximately 282 trillion) galaxies, can you imagine that? It's like a whole universe inside a video game. However, since the game's map was enormous, the publisher was afraid that could cause disbelief in players and the game's map was reduced to eight galaxies.

It was revolutionary that in the early eighties developers could use computers' power to generate maps. However since they were so limited by the computer or consoles hardware's restrictions at that time, this approach was not widely used and they were not utilizing all the power of procedural generation.

B. Modern Use

Procedural generation is therefore more widely used. This method has its origins in video games and has expanded to fields such as the film industry. **Terragen**, a computational tool for creating realistic environments using procedural generation, became fundamental in the film and television industry, being used, for example, in creating the environments for the film **The Martian** (2015).

Returning to digital games, many open-world and survival games procedurally generate their game worlds from either a random seed or one provided by the player, ensuring that each playthrough is different. Notable examples include **Minecraft** (2011) and **Vintage Story** (2016). With the growth of multiplayer games, procedural generation of environments began to appear in this area. This can be seen from the multiplayer mode of **Minecraft** and **No Man's Sky**, a game mentioned before. **No Man's Sky** shows the full potential of this technique, featuring the largest game map in video game history, with 18 trillion planets. The original map of **Elite** may seem simple in comparison. These planets can be explored on foot and by flight. Each planet has its own characteristics, including unique landscapes, plants, and animals.

III. Steps of Procedural Generation of Environments

Procedural generation of environments can be divided into three main steps: terrain generation, vegetation generation and building generation. This section will discuss the key algorithms used in each step, detailing their characteristics, how they work and showing real examples of applications using these steps.

A. Procedural Generation of Terrain

In visual computing, noise algorithms [15] (Figure 2), play a crucial role in creating natural terrains. You might be wondering **noise? How an annoying sound can be useful to generate terrains?**. In mathematics the word noise has different meaning, it refers to a pattern of pseudo random values or variations that simulates natural randomness. These algorithms generate random data in a consistent and smooth manner. This makes it ideal for creating realistic terrain and textures. By introducing a control pattern into these random patterns that can adjust the frequency, amplitude, and continuity of the pattern. This makes it possible to simulate natural phenomena such as ocean waves and mountains. This makes it possible to simulate natural phenomena such as ocean waves and mountains.

The most commonly used noise algorithms are:

- **Perlin noise**: A gradient-based noise that creates a smooth, continuous variations. Ideal for creating natural textures such as terrain, clouds, water surfaces, and more.
- **Simplex noise**: An improved version of Perlin noise. It provides faster calculations and reduces artifacts.
- **Voronoi noise**: Creates a cell-like pattern. Often used to simulate natural phenomena such as cracks, veins, and organic structures.
- **White noise**: A completely random signal with no discernible patterns. Used to add randomness to textures or to simulate effects such as static.
- **Worley noise**: Creates cell-based patterns that resemble natural phenomena such as terrain fractures, marble, and cellular structures. Used in texturing to simulate organic patterns or irregularities such as cracks, veins, and other geological features.

One practical example of the use of these algorithms is the famous sandbox video game **Minecraft**.Figure 3, shows how its biomes are generated [7]. It's a layer process that starts with an island layer, followed by multiple zoom operations to increase resolution (from 4096 to 256).Then temperature layers are used to determine climate zones, which are essential for determining the location of the biome. The process then splits into three stacks: biome generation, river system creation using **White Noise**, and ocean temperature mapping using **Perlin Noise**.The generation of the biome is completed by combining the previous elements and applying a **Voronoi** zoom operation.

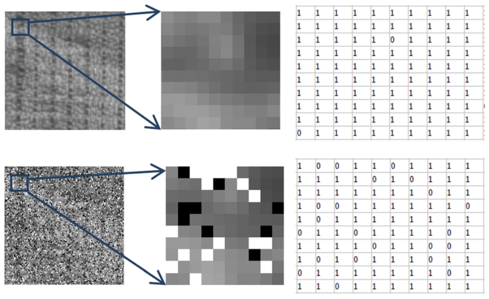
Fig. 2. Noise Algorithm.

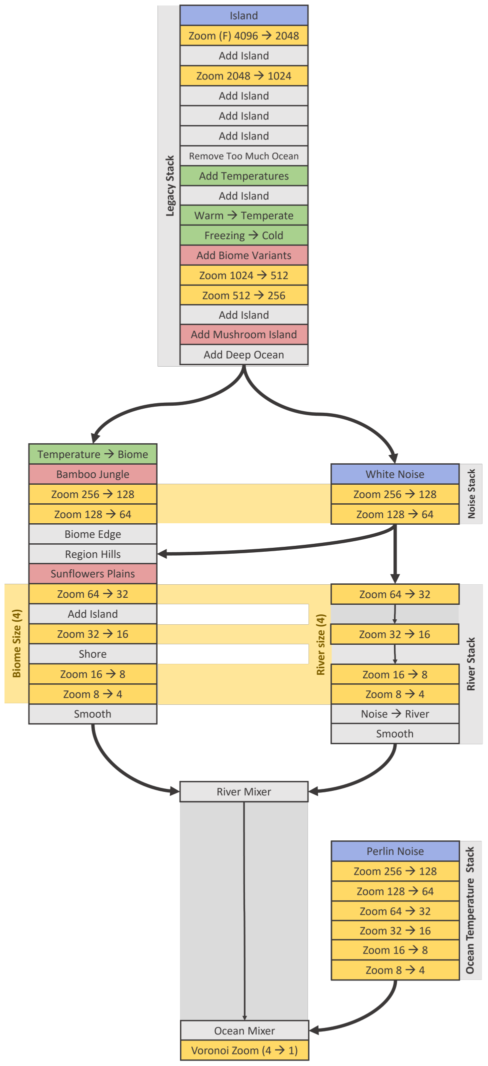

Fig. 3. Minecraft Biomes Generation.

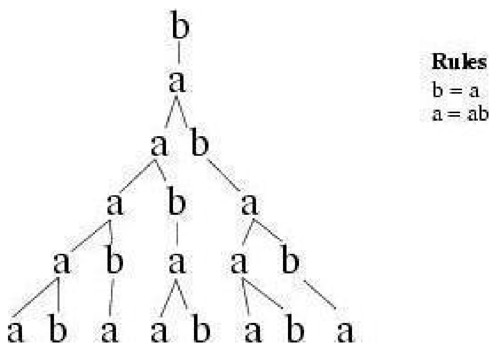

Fig. 4. L-System Grammar Example

B. Procedural Generation of Vegetation

Natural environments often include vegetation. In order to create more realistic virtual environments with vegetation, procedural generation applications have to adopt **L-Systems** that could simulate different growth stages of plants creating dynamic and more diverse ecosystems.

The **'L'** in **L-System** stands for **Lindenmayer**, the biologist who created this method [9]. The purpose of this system was to provide a formal description of the development of simple multicellular organisms and to illustrate the neighborhood relationships between plant cells. Later, this method was extended to describe higher plants and complex branching structures.

Plants in different states of their growth create repetitive structures and patterns, to explain this phenomenon this systems use a set of production rules to recursively (grammars) explain how a simple plant can increase to have complex structures. The **Lindenmayer-Systems** grammars can be defined as a tuple $G = (V, \omega, P)$, where V is the alphabet, ω is the axiom, and P is the set of production rules. For example, considering $V = (F, +, -, [,])$, a simple **L-System** that generates a structure similar to a plant could be defined as:

- $\omega = F$
- P = $F \rightarrow F[+F]F[-F]F$
- Where:
  - F represents drawing forward
  - $+$ represents turning left by 25 degrees
  - $-$ represents turning right by 25 degrees
  - $[$ represents saving the current position
  - $]$ represents returning to the saved position

When this system is interpreted in multiple iterations, it creates increasingly complex branching patterns that resemble plant structures (Figure 4), demonstrating how simple rules can create botanical forms. With the simplicity and the power of **L-Systems**,video games like **No Man's Sky** adopt this technique to procedural generate their environments vegetation.

C. Procedural Generation of Buildings

Buildings are fundamental components of the virtual environments, and their creation cannot be made in one single step. Software like the three-dimensional modeling, **City Engine** exemplifies this process. It starts by designing the building on the ground floor and then building its shape and facade generation.

1) Procedural Layout Generation: To create these layouts, **Treemap** based algorithms were used for filling rooms in a given rectangular building, but these algorithms were limited to rectangular spaces. To avoid this issue **Lopes** [10] proposed the **constrained growth method for procedural floor plan generation**. This method is able to process any arbitrary shape region, allowing the creation of more realistic and flexible room layouts compared to the rectangular ones.

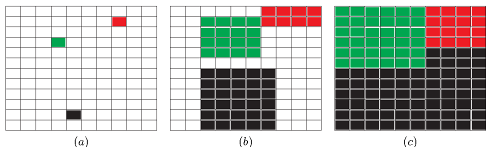

Fig. 5. Constrained growth method for procedural floor plan generation

Figure 5 illustrates how this technique works: (a) shows a colored grid with the starting cell for each room (red,green and black); (b) demonstrates the intermediate growth phase for each room; and (c) shows the final stage of this technique when each room has grown to its predefined size while maintaining spatial relationships and adjacency constraints.

2) Procedural Facade Generation: In order to procedurally generate a more diverse range of buildings, its use shape grammars(Figure 6). These grammars are basic type of split grammars which instead of using a set of production rules with symbols, like the **L-Systems** , they use shapes and allow their rotation and splitting. Examples of shape grammars:

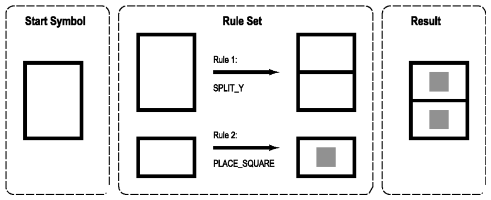

Fig. 6. Shape Grammar Example

- **CGA**: Created by **Müller** [3], can automatically create a variety of models based on user-written rules.
- **CGA++**: Introduced by **Schwarz** [3], increases the context sensitivity of **CGA** shapes, allowing for combinations of multiple buildings to be combined to produce better results.

IV. Procedural vs Manual Generation of Environments

It is important to analyze the pros and cons of procedural generation of environment in comparison to creating them manually. This comparison is important to understand when each method may be more appropriate in real-world applications. Procedural generation has several important advantages: First, it significantly reduces development time and effort. This is because the environment is created automatically when the algorithm is properly applied. This automation reduces production costs. This is because fewer artists and designers are required to create environments. Additionally, procedural generation provides much more detailed results. Reducing duplicate patterns often interrupts the manually created environment. However, manual built environments have their own unique strengths. It allows designers to creatively control every aspect of their environment. This allows for the deliberate selection of artifacts that cannot be reproduced by the algorithm. This is in contrast to procedural generation, which relies on well-designed algorithms that can create errors or inconsistent patterns if not implemented correctly. Guarantees consistent quality in the environment Perhaps the most important thing is Manual Creation is excellent at creating storytelling-driven environments. Each element serves a specific purpose in the narrative or game design. The following table summarizes the main differences between the two methods:

Procedural Generation	**Manual Generation**
Reduced development time	Greater creative control
Lower production costs	Consistent quality assurance
High content variety	Precise element placement
Scalable content creation	Better narrative integration
Risk of algorithmic artifacts	Time-intensive production

TABLE I
PROCEDURAL VS MANUAL ENVIRONMENT GENERATION

V. CONCLUSION

To sum up procedural generation of environments, is an efficient, powerful an save time technique to create virtual environments. It's safe to say in the future, more companies will adopt this method, with anticipated developments including:

1) Better optimization of procedural algorithms.
2) Expanded computer applications
3) Novel implementation strategies

For example, the most anticipated video game in history **Grand Theft Auto VI** (2025), will use procedural generation, to create the interiors of 70% of buildings making them enterable, a long request feature requested among the fans of **Grand Theft Auto** series. However, in this article you read about the advantages and disadvantages of procedural generation. Therefore when creating a computer application with virtual environments, we need to consider the type of environment we want before choosing between manual or procedural we need to think of the type of environment we want.

ACKNOWLEDGMENTS

In this article, artificial intelligence (AI) writing tools, specifically **Claude AI** and **ChatGPT**, were used. The purpose of using these AI tools was to correct grammatical errors, improve cohesion between sentences, and suggest synonyms to avoid the repetition of certain words.

REFERENCES

[1] Wikipedia Contributors, "Procedural generation," *Wikipedia, Aug. 16, 2019. [Online]. Available: https://en.wikipedia.org/wiki/Procedural_generation*

[2] Wikipedia Contributors, "Fractal," Wikipedia, Jul. 23, 2019. [Online]. Available: https://en.wikipedia.org/wiki/Fractal

[3] Y. Liu, "An Overview of Procedurally Generating Virtual Environments," International Journal of Computer Science and Information Technology, vol. 2, no. 1, pp. 141–147, Mar. 2024, doi: https://doi.org/10.62051/ijcsit.v2n1.16.

[4] M. C. Erbudak, "Terrain Rendering via Perlin Noise and OpenGL Geometry Shaders," Medium, Jun. 15, 2022. [Online]. Available: https://medium.com/@muhammedcan.erbudak/terrain-rendering-via-perlin-noise-and-opengl-geometry-shaders-55eb10aa1d3c

[5] "The Book of Shaders," The Book of Shaders. [Online]. Available: https://thebookofshaders.com/11/

[6] M. Letourneau and J. W. Sharp, AMS-StyleGuide-online.pdf, American Mathematical Society, Providence, RI, USA, [Online]. Available: http://www.ams.org/arc/styleguide/index.html

[7] A. Zucconi, "The World Generation of Minecraft," Alan Zucconi, Jun. 5, 2022. [Online]. Available: https://www.alanzucconi.com/2022/06/05/minecraft-world-generation/

[8] "Procedural Plant Generation with L-Systems," YouTube. [Online]. Available: https://www.youtube.com/watch?v=feNVBEPXAcE

[9] "L-system," Wikipedia, Jan. 4, 2021. [Online]. Available: https://en.wikipedia.org/wiki/L-system

[10] R. Lopes, T. Tutenel, R. M. Smelik, J. De, and R. Bidarra, "A constrained growth method for procedural floor plan generation," in Proceedings of the 11th International Conference on Intelligent Games and Simulation, Jan. 2010. [Online]. Available: https://www.researchgate.net/publication/265988238_A_constrained_growth_method_for_procedural_floor_plan_generation

[11] D. Vaillancourt, [Online]. Available: https://proceedings.esri.com/library/userconf/swuc14/papers/swucTW_12.pdf

[12] D. Vaillancourt, [Online]. Available: https://proceedings.esri.com/library/userconf/swuc14/papers/swucTW_12.pdf. Accessed: Apr. 25, 2024.

[13] "What is Terragen?," Planetside Software, 2024. [Online]. Available: http://planetside.co.uk/terragen-overview/. Accessed: Nov. 25, 2024.

[14] "Unveiling the Revolutionary Interior System of GTA 6," Toolify AI, [Online]. Available: https://www.toolify.ai/ai-news/unveiling-the-revolutionary-interior-system-of-gta-6-2105374. Accessed: Apr. 25, 2024.

[15] "Perlin noise," Wikipedia, [Online]. Available: https://en.wikipedia.org/wiki/Perlin_noise. Accessed: Apr. 25, 2024.