A rule based procedural content generation system

Abstract: This paper introduces new ways and experimental results of procedural content generation (PCG) for natural environments and objects (such as terrain, biomes, river, and vegetation), with special attention to the game design and aesthetic visualization. Our solution allows us to define a set of rules to control the content generation in order to meet the requirements of game design and level design teams. In this paper we focused on the content generation for natural environments and used the PCG in a 3rd person open-world adventure game implemented in Unity. A playtesting has been conducted and the results are analyzed.

Keywords: Procedural content generation, Game Design, Level Design, 3rd Person Open-world Adventure Game, Algorithm.

1. Introduction

Procedural Content Generation (PCG) refers to the practice of generating game content through an AI-processed procedural generation method. PCG has been used in the game and cinema industry since the 80's for several different purposes. Some of the popular motivations are replayability, adaptability and economics. Due to the amount and diversity of automatically generated content, it is possible to offer a more refreshing gameplay each time the player starts the game, they will have a different experience. In a more advanced application, the PCG AI can be sensitive to the player skills, and the expected level difficulty, in order to adapt the gameplay to their skill level. On the other hand, it is still possible to replace a somewhat time-consuming process in game development by automating the processes typically addressed by a Human that is easily performed by AI. This method is, for the reasons mentioned above, very often approached by independent studios that often do not contain the human, technological and sufficient funds to invest in manpower for these areas of video game design. Thus, with this method it is possible to use algorithms that are easily adaptable to various situations depending on the circumstances presented during content generation.

The term "open-world" has a specific connotation attached to it: composed of vast, free-to-explore open environments, spatial orientation and navigation are key to the interpretation of gameplay goals. As such, there's an emphasis on "a large number of different verbal and nonverbal clues, which together form the guidance systems" of the game to induce the player's awareness to use a specific path towards a goal [1]. Furthering a sense of immersion, the sense of gameplay progress and even "progress in the plot of a digital game is equated with progress in spatial environments" [2]. So "features such as high points for a better overview, spectacular landscapes underscoring dramatic storylines, prominent locations serving as obvious landmarks, regularly distributed settlements justifying the next resources

to procure or missions to execute, and labyrinth-like forests or towering high-rises, (...) perforated with cave systems, subway lines, and tunnel entrances by which the depth of the worlds is extended without having to provide spatial continuity (for example, one enters a tunnel system and emerges at another location)" [2], the landscape stages multiple calls to action.

Taking into account this relevance of exploring, cartographing and traversing an expansive world implied in the 'open-world' genre in combination with 'adventure', where the player must carry out different missions of a strategic type to evolve and succeed, the enunciation of narrative through environmental storytelling, generating emergent narratives among players, rather than just embedding plot-driven events, further evidences the importance of landscape variance that can be proposed through different biomes. A biome is a large area characterized by its vegetation, soil, climate, and wildlife. There are five major types of biomes: aquatic, grassland, forest, desert, and tundra [3]. In this paper we propose a novel system that uses biome seeds to create multiple biomes and populates them densely with specific assets.

The research questions of this paper are the following:

- Identification of types of game content that can be generated by PCG.
- Explore existing technical solutions and identify their main features, capabilities and limitations.
- Identify studies that address the contributions, difficulties and challenges that PCG can bring to game development from a game design perspective.
- Propose and describe the new PCG algorithm and its architecture.
- Test the new algorithm and gather feedback about the game and the PCG algorithm from a player's perspective.

The rest of the paper is structured as follows. In Section two we provide a literature review about different aspects of PCG generation solutions, the difficulties and challenges of its use on game design and development. In section three we describe the entire process of the design of our solution, identifying the elements to generate and the architecture of the PCG algorithm. Section four describes the application of the PCG system in a 3rd person open-world adventure game, and presents the playtesting game session, with focus on the automatically generated world. Section five gives a conclusion to our research and discusses future work.

2. Literature review

This section begins with an overview of what kind of content can be generated procedurally, follows several technical methods to generate content, and some specific solutions to help solve your problem are discussed.

In [4], the authors introduced six main classes of game content that can be generated by PCG systems: Game Bits - are basically the assets of the game, textures, sound, materials, and some simple behaviors. Game Space - is the environment in which the game takes place. Can be indoor or outdoor maps. Game System - include more complex environments as road networks, urban environments or ecosystems. Game Scenarios - are the scenes organized into levels with events of the game. Can be puzzles, levels, storyboards. Game Design - all the main features that define the design of the game such as rules, goals, mechanics. Derived Content - content created outside the main gameplay of the game. Can be news, leaderboards, videos.

Several technical approaches can be used to generate content, with different features, capabilities, and tradeoffs for design. In [5] several PCG solutions are identified:

Simulation based - the PCG starts with a template world and a set of operators that can change the world evolution in a simulation mode. Generation as optimization - It looks for the best combination of components according to a certain evaluation function that determines the desired results to respect as much as possible a value of optimization. Some examples are evolutionary algorithms as genetic algorithms. Constraint satisfaction - certain design constraints are specified, whereupon a constraints solver will create content that meets these constraints. Constructive generation - implies the generation of content through predefined building blocks which, later, are used in the generation of levels according to the implemented algorithm. It is a type of approach often used in Rogue-Likes. Generation with Grammars - This method consists of specifying rules according to a certain space of defined possibilities which, passing to an interpreter, are processed in the creation of contents. Constructionist selection - although it is debatable whether it should be considered a procedural generation algorithm, in this method the building blocks are selected from a library, and used by the PCG to create an environment for the player.

In [6] a study is carried out on how PCG is used in different game genres, for which the algorithms are grouped into three categories: traditional methods, search-based methods, and machine learning methods. Traditional methods - encompass, pseudo-random number generator and constructive methods, generative grammars and others; Search-based - methods include artificial intelligence and operational search algorithms such as heuristic search, local search and optimization. Machine learning - includes algorithms that use data to improve performance on a specific task.

In [7] a finer technical classification is made, dividing the algorithms into different types: Terrain Generation - allows the generation of virtual terrains that are the basis of the game world. City, Building and Road Generation - encompasses the algorithms that allow the generation of this type of more complex systems. Level Generation - is responsible for creating areas where the game's action takes place (e.g. platformers, arenas, dungeons). Story and Narrative Generation - generate stories based on characters and their relationships, making the games replayable with new narratives. Puzzle Generation - create content that targets cognitive problem-solving skills, rather than motor or reaction skills. Dynamic Generation - refers to methods capable of creating dynamic systems and agents such as weather, individual AI and flocking behaviors. Miscellaneous Generation - other systems that have not received much scientific attention.

However, the difficulties and challenges related to the use of these techniques is the balance of the games itself and making a good construction of levels and content where there are no repetitive or chaotic patterns that can negatively affect the player's gameplay. In [8] is presented a framework for PCG that quantifies the player's experience and the quality of content, based on a computational model, and is able to generate content for different emotional representations.

To develop a well-regarded game in terms of design, one must take into account the demands required by the type of design that is intended for that same game. It is therefore necessary to take into account how the player and the designers will influence the generation of the content and how to pass this information on to the AI, so that it manages the content considering these requirements. In [9] is presented a framework capable of analyzing and discussing the role of PCG in game design. This framework identifies three dynamics that only PCG can provide: the replayability can offer different surprising content, the possibility

to build strategies for the different content on each play, and provides different content that supports the replay of mechanisms in different scenarios.

3. Design of PCG algorithm

The proposed PCG system is designed to be used in an open-world adventure game. In this section, the elements belonging to a natural environment are identified, and then the PCG architecture capable of generating this world is presented.

3.1. Elements identification

In order to take a deeper look into our algorithm, we first need to identify all the elements it can generate, Fig. 1 shows exactly that.

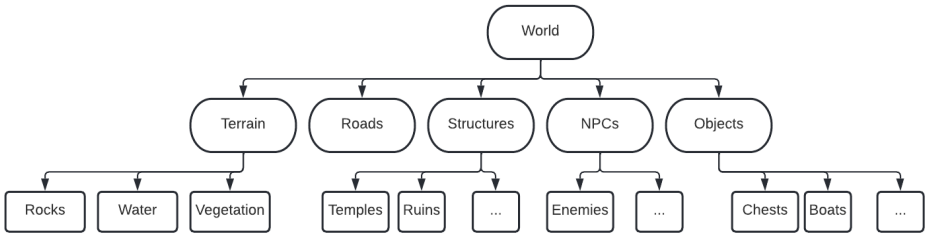

Fig. 1. Identification of the elements that can be included in the Scene.

We divided the elements into three types by granularity. The first with more granularity consists of only one element:

- World - the name World is in itself self explanatory, we are generating the world where we are placing all the Structures, Objects, NPC's, etc.

The second type of elements are:

- Terrain – is the ground in which the player stands, terrain varies with height and it's thanks to this variation that we can find lakes, mountains, forests and all the assets that make these feel like real locations;
- Roads – an offset in hue from the terrain's color allows us to generate roads which the player can follow and explore;
- Structures – our world generation allows us to dictate a set of rules that when met will spawn our structures, these are key points in driving the story forwards while proving elements of exploration to the player;
- NPC's – these spawn when their spawning conditions are met and serve to add enemies for the player to fight;
- Objects – are assets that aren't related to our terrain, these serve to amplify the gameplay experience of the player and allow for a better environmental storytelling.

The third type of elements (more granularity), consists of a deconstruction of the key components that make up our second layer, it's where we place all the content you can find inside our main identification elements:

- Under Terrain we have three groups, we have vegetation, rocks, and water, all key components in making sure our terrain generation mimics real world environments.
- Examples of Structures are: temples, ruins, houses, buildings.

- The only component inside NPC's (non-player characters) is enemies, named this way because they are all hostile mobs.
- Within Objects we find any type of objects that do not belong to the other classes (e.g. chests and boats)

3.2. Architecture/Design/Model

Taking into consideration the aforementioned elements, for a more practical explanation of the process, we can further divide all the elements in distinct layers/groups to understand its phases of content generation (Fig. 2).

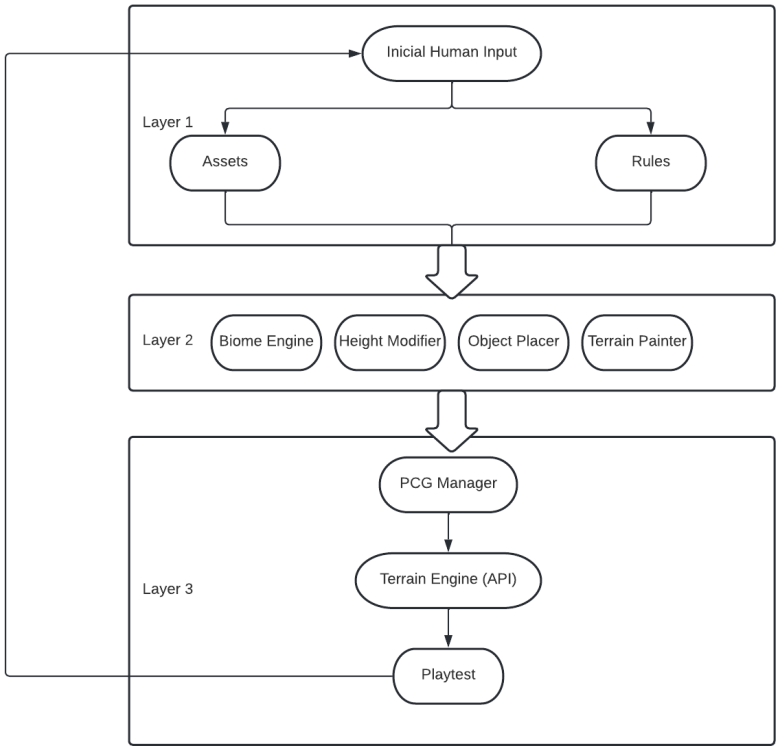

Fig. 2. Architecture of the Seed based PCG System.

The first layer, the input, is when the human designer has influence on the data being processed by the system. We first start to introduce the assets defined by the assets list provided by the designer. All these assets are inherently required to the level design in particular and given the natural complex subjective properties of these selection, for a more pragmatic approach, this process must be accomplished purely by human input.

The second layer, the configuration, is the intermediate section between both human and AI processes. Here, the designer defines the rules of terrain generation in the form of concrete values. This process allows for a better translation between the more complex designer

ideas/rules to a language that the algorithm can properly decipher. Here we evaluate important terrain generation and level design properties such as:

- Height Modifier – One of the three modifiers to be implemented in each biome. The height modifier adjusts the height of the terrain according to the biome in particular (e.g. biomes like swamps tend to have lower heights and valleys tend to be more irregular). The human designer has influence on settings such as height variation, noise map scale, number of passes the algorithm takes and much more. Here the algorithm will take all these data and work around them using seven different methods that we explain further in the next chapters. Here it's also defined rules for positioning buildings around the terrain;

- Biome generation – The biome generation is a crucial part of the system given the fact that we are trying to simulate credible real world terrain generation. Here, the designer can define settings like decay rate, minimum and maximum intensity and apply modifiers such as textures and height alterations dependent on the biome in particular. After introducing the data, the algorithm takes action by processing all the information provided and carefully spreading each biome property such as textures and other assets throughout the terrain using an ooze-style biome generation method [10] for smoother and more natural transitions between different biomes. This process is better explained in the next chapter of this paper;

- Terrain Painter (Texturization) - the "Terrain Painter", as the name implies, applies textures to the terrain according to rules defined by the designer, in order to make the game environment more immersive and natural. This component takes into account several parameters to apply different textures, such: elevation, biomes and slope of terrain. For example, in a situation where the terrain is at a higher height (such as the peak of high mountains) it makes sense for snow to be generated. In the opposite way, when the altitude is close to the water level a sand or grass texture will be applied. In the latter case, the choice of texture to apply will also be influenced by the biome and the slope of the terrain. The first rule will always be the elevation the texture generator will choose a texture from a predefined array of textures and apply them to the terrain according to the height which, normally is the y value of the coordinates. Then other parameters will be evaluated for the choice of textures, such biome or the slope of the terrain.
- Object placer – The object placer is another modifier used in the biome generation configurations. Take into consideration that we identify as "objects": simple and unitary assets with little complexity regarding its shape and number of elements. The aforementioned "buildings" are more complex assets consisting of multiple elements and complex shapes, consequently, requiring more convoluted positioning rules defined by the designer. Here we work with these simple objects by defining rules such as: height limits, the density, the number of objects to spawn, define if the said object can spawn above or in water or even both, the list of assets to spawn, etc. It is important to carefully evaluate certain aspects of object placement to build a more natural and real environment, particularly, where we place them. Rules such as the height limits and verifying if certain objects can go above or under water are crucial to build an immersive terrain. Misplacement of these objects (e.g. trees generally can't go under water and lily pads can't climb up mountains) can

immensely compromise the gameplay experience thus being one of the most important aspects of the world generation for a better player immersion;

The third layer receives the biome map, the terrain noise map and the rules defined previously. All these elements are passed to the PCG Manager which will generate the terrain, apply the biomes and place the objects. The PCG Manager will then control the Terrain Engine, to perform the various necessary transformations such as: terrain elevation, texture application, and object placement. The end result will be generation of a natural-looking 3D scenery with features like: mountains, lakes, different types of biomes, vegetation and different objects.

At the end of the generation process, the scenario is included in a prototype that will be playtested by the design team to identify errors and to obtain feedback on the players' experience in relation to the generated content and adjust the PCG system. This entire process has a cyclical and iterative nature, where it will be necessary to make the PCG system better to produce a plot according to the requirements of the design team.

4. Conclusions

Procedural content generation can significantly reduce costs and development time of games. However, the systems are often not flexible enough to the requirements defined by the level designer. This paper showed how we approached the design and implementation of a ruled biome based procedural content generation system, with attention to the requirements of the level design. The proposed system was designed to be used in a 3rd person open-world adventure game, and was developed using the Unity game engine.

During the development of the game, a playtesting session was held, from which it was possible to obtain positive feedback about the game, about the procedural generation of the content and about the gameplay that it provides to the players. This feedback allowed us to adjust certain parameters of the algorithm with the aim of improving mainly the gameplay, the game experience and its difficulty balance as the player progresses in the game.

As future work, we intend to carry out a new playtesting session to evaluate the changes made after the first playtesting. We also intend to apply L-systems [16], [17] to the modeling of plants, and a terrain painter based on a Fuzzy Logic controller that can be influenced by parameters [18], [19].

References

[1] B. Suter, M. Kocher, e R. Bauer, Eds., *Games and Rules. Game Mechanics for the "Magic Circle"*. Bielefeld: Transcript, pp. 169-189., 2018.

[2] B. Suter, R. Bauer, e M. Kocher, *Narrative Mechanics: Strategies and Meanings in Games and Real Life*. transcript Verlag. pp. 161-176, 2021. doi: 10.14361/9783839453452.

[3] National Geographic Society, «The Five Major Types of Biomes | National Geographic Society», *The Five Major Types of Biomes | National Geographic Society*, maio de 2022. https://education.nationalgeographic.org/resource/five-major-types-biomes (acedido 15 de junho de 2022).

[4] M. Hendrikx, S. Meijer, J. Van Der Velden, e A. Iosup, «Procedural content generation for games: A survey», *ACM Trans. Multimed. Comput. Commun. Appl.*, vol. 9, n. 1, p. 1:1-1:22, fevereiro 2013, doi: 10.1145/2422956.2422957.
[5] G. Smith, «Procedural Content Generation - An Overview», 2015.
[6] N. A. Barriga, *A Short Introduction to Procedural Content Generation Algorithms for Videogames*. 2018.
[7] M. Dahrén, «The Usage of PCG Techniques Within Different Game Genres», 2021. Acedido: 26 de maio de 2022. [Em linha]. Disponível em: http://urn.kb.se/resolve?urn=urn:nbn:se:mau:diva-46422
[8] G. N. Yannakakis e J. Togelius, «Experience-Driven Procedural Content Generation», *IEEE Trans. Affect. Comput.*, vol. 2, n. 3, pp. 147–161, jul. 2011, doi: 10.1109/T-AFFC.2011.6.
[9] G. Smith, «Understanding procedural content generation: A design-centric analysis of the role of PCG in games», *Conf. Hum. Factors Comput. Syst. - Proc.*, abr. 2014, doi: 10.1145/2556288.2557341.
[10] P. Scott, «Procedural Biome Generation: Ooze-Style - PROCJAM Tutorials», 2018. https://www.procjam.com/tutorials/en/ooze/ (acedido 23 de junho de 2022).