Brief Dreams: An Interactive Procedural Virtual World Builder for Imaginary Landscapes

by

Yiming Zheng

Master of Fine Arts, Bard College, 2020

Bachelor of Fine Arts, Beijing Film Academy, 2017

Project Submitted in Partial Fulfillment of the
Requirements for the Degree of
Master of Science

in the
School of Interactive Arts and Technology
Faculty of Communication, Art and Technology

Declaration of Committee

Name:	**Yiming Zheng**
Degree:	**Master of Science**
Title:	**Brief Dreams: An Interactive Procedural Virtual World Builder for Imaginary Landscapes**
Committee:	**Kate Hennessy** Supervisor Associate Professor, Interactive Arts and Technology
	Halil Erhan Committee Member Associate Professor, Interactive Arts and Technology

Abstract

Brief Dreams is an interactive art project that allows users to design, build and explore virtual environments using the procedural content generation (PCG) framework in Unreal Engine 5.3. In this project, users will first build 3D environments on the empty terrain with provided procedural contents to generate 3D models in predefined cluster patterns, including natural biomes, canyons, trails and rivers. As users build their virtual environments, they can explore their creations in a first-person walkthrough and modify the appearance of the world by changing the time of the day and weather. This project aims to provide an accessible way for general users, with or without a professional design background, to take advantage of the PCG framework to build virtual worlds in a short period of time.

Keywords: Procedural Content Generation; Unreal Engine; 3D World Building; Video games

Acknowledgements

I would like to express my deepest gratitude to my project supervisor, Dr. Hennessy, for her invaluable guidance and support throughout the development of this project and its documentation. Her insightful suggestions and advice motivated me to make steady progress and complete this prototype, which I am proud to include in my portfolio.

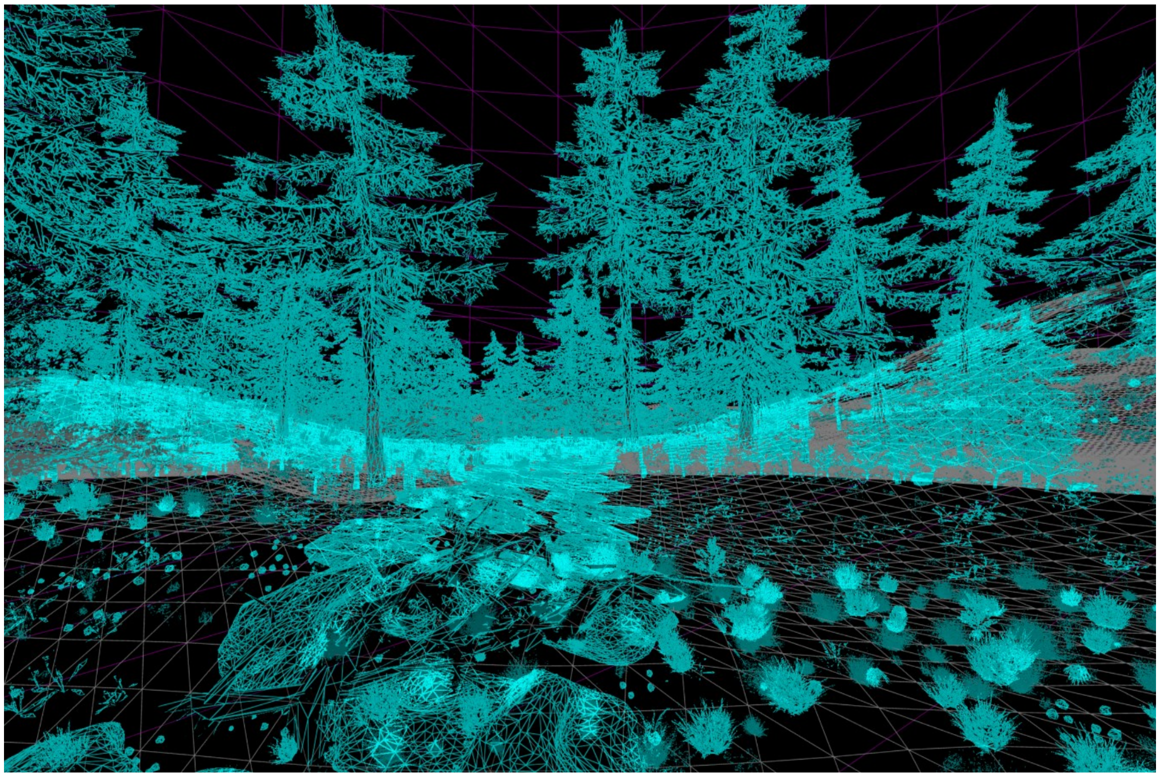
I am also grateful to Dr. Erhan for providing valuable feedback that helped shape the initial concept and refine it into the final iteration of the prototype.

I extend my sincere thanks to my friends, especially Henry Leung, who has always been willing to offer help on my project and share his thoughts with me.

Finally, I want to express my heartfelt appreciation for the support of my family. Their support has brightened my days and played a crucial role in helping me complete this program.

Table of Contents

List of Figures



Yiming Zheng, 2024

Chapter 1.

Introduction

The video game industry has emerged as an exceptionally dynamic and rapidly expanding domain, positioning itself as the leading force in the realm of entertainment industry. As the scale and complexity of video games have grown, the development process demands a significant amount of labor and resources (Adeel et al., 2021), requiring years of development and hundreds of employees to complete (Kanode & Haddad, 2009). This complexity has led to an increasing need for automated tools that allow artists and designers to create expansive, intricate virtual environments.

Procedural content generation has emerged as a solution to address the challenges of generating game content based on predetermined algorithms (Togelius et al., 2011). PCG is a term that describes computer software capable of autonomously generating game content, either independently or in collaboration with human players or designers(Shaker et al., 2016), this technology can be widely used to generate various types of content for video games, including 3D assets, game levels, and quests (Gravina et al., 2019). Recent research demonstrates that combining the potential of PCG with trained AI models can lead to the design of functional and playable 2D top-down arcade-style game levels while maximizing diversity in the newly created levels (Steward, 2023).

Although many world-building approaches in video games utilize PCG to automatically generate various elements within virtual worlds, PCG is defined as the algorithmic creation of game content with limited or indirect user input (Togelius et al., 2011), such as setting parameters to influence the generated content. This limitation may hinder the creativity and engagement of players who wish to have more control and freedom in creating a virtual environment (Volkmar et al., 2022). To address this issue, I propose a new approach to utilize PCG in virtual world-building that invites beginner designers, artists, and novices interested in world-building and level design, who may face obstacles in manual world-building workflows, to directly manipulate the procedural content provided through a user interface, allowing them to build virtual environments in a short period of time.

1.1. Motivation

My motivation for creating this project comes from my practical experience as a developer and artist in creating intermediate-scale 3D environments for video game projects, as well as my experience as a player in navigating various procedural content in large scale video games.

Working as a level designer and environmental artist on the VR game project *Harmony* (2023), I started world building by first finding reference images and photographs of real-life underwater scenes. Based on these references, I selected and imported 3D assets from the asset store and manually placed individual 3D models on the terrain to create coral reefs and the seabed (Figure 2). This process required a significant amount of time to ensure that every 3D model was placed in the designated location without clipping or floating above the terrain while also matching the visual appearance of the reference images.

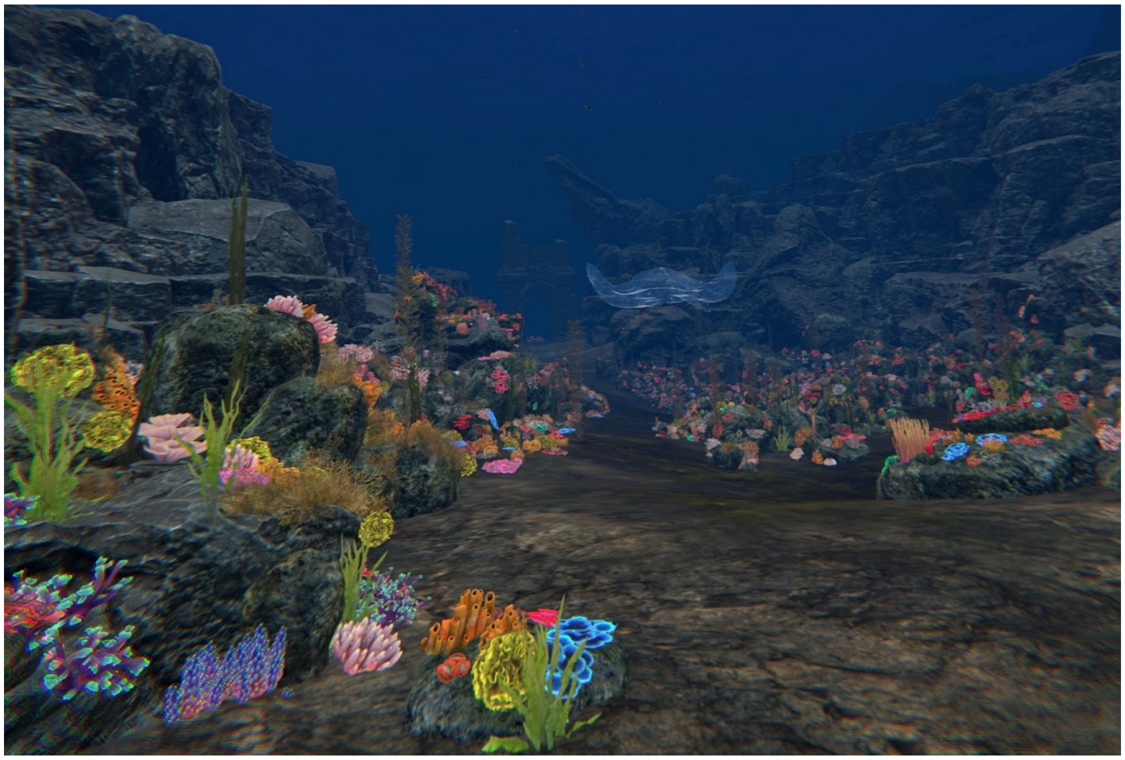


Figure 1 Screenshot from my VR game project Harmony, I built this underwater scene by manually placing each 3D asset on the landscape.

Yiming Zheng, 2023

The entire world-building process spanned several weeks to complete, and such world-building methods made changing level design halfway to a challenging task, as it required partially redoing the labor-intensive world-building process to modify the environment. In such cases, I suggest that procedural content generation (PCG) could potentially serve as a solution to the problems encountered in traditional world-building workflows by generating 3D assets in predefined patterns and automatically snapping them to the terrain. My personal experiences and challenges in building virtual worlds for video games have motivated me to develop a PCG world building system that provides an accessible approach for people with or without professional experience in virtual world building to create virtual environments with PCG in a relatively short amount of time.

1.2. Project Goals

The primary goal of this project is to develop an interactive world building system built on top of the Unreal Engine's Procedural Content Generation Framework. Users will be able to interact with the PCG system through a mouse interface to select and generate pre-made procedural contents from a top-down perspective. The procedural content includes the environments such as forest, burnt forest, canyons, campsites, pathways, and rivers.

In addition to enabling users to create virtual environments using PCG, this project features an immersive experience by allowing users to explore and examine the design of the virtual world from a first-person perspective and enable them to revise their design by going back to the world building top-down perspective at any points of the exploration. The documentation of this project serves as a comprehensive record of my research, design, and development. It also provides valuable insight and reference for my future development in PCG frameworks in Unreal Engine.

By achieving these goals, this project has provided me with an in-depth understanding of building interactive runtime PCG systems within Unreal Engine. Moreover, it has resulted in a functional prototype that serves as a proof of concept, demonstrating that a runtime PCG system can offer an accessible and user-friendly world-building approach, allowing audiences with or without professional backgrounds in virtual world building to participate in the creation of diverse and immersive virtual environments.

Chapter 2.

Related Works

2.1. Procedural Content Generation

In recent years, the video game industry has establishing itself as a highly innovative and fast-growing sector that has become a dominant player in the entertainment industry. Taking advantage of technological advances, video games are becoming more visually appealing, realistic, and immersive. The increasing scale of the game demands a significant amount of effort and resources to create intricate virtual environments (Adeel et al., 2021). While traditional hand-built virtual environments are rich in detail and creativity, such environments often require extensive time, resources, and artistic expertise. To reduce the intensive labor and time cost of hand-building virtual environments, Procedural Content Generation (PCG) has been introduced to game development to accelerate the creation of video games by algorithmically generating environments from a combination of human-created assets and computer-generated randomness (Lech et al., 2021; De Seta & Alviano, 2022).

A prime example of successful PCG implementation is seen in Minecraft (Figure 3), an open-world crafting sandbox game that utilizes PCG to create visually infinite terrain and biomes. The game world is composed with procedural generated cubical blocks that relied on different materials to visually identify the nature of the blocks. In generating the landscape, the game utilized scaled 3D Perlin noise with linear interpolation (Procedural Content Generation Wiki, 2012) to create pseudo-random heightmap, enabling the generation of large and continuous terrain with seamless blending near boundaries (Jain, A et al., 2022). To create biomes in its cubical world, Minecraft used Whittaker diagram to determine what plants and terrain features (such as sand and snow) should be created in a particular area based on the height map (Figure 3) (Procedural Content Generation Wiki, 2012).

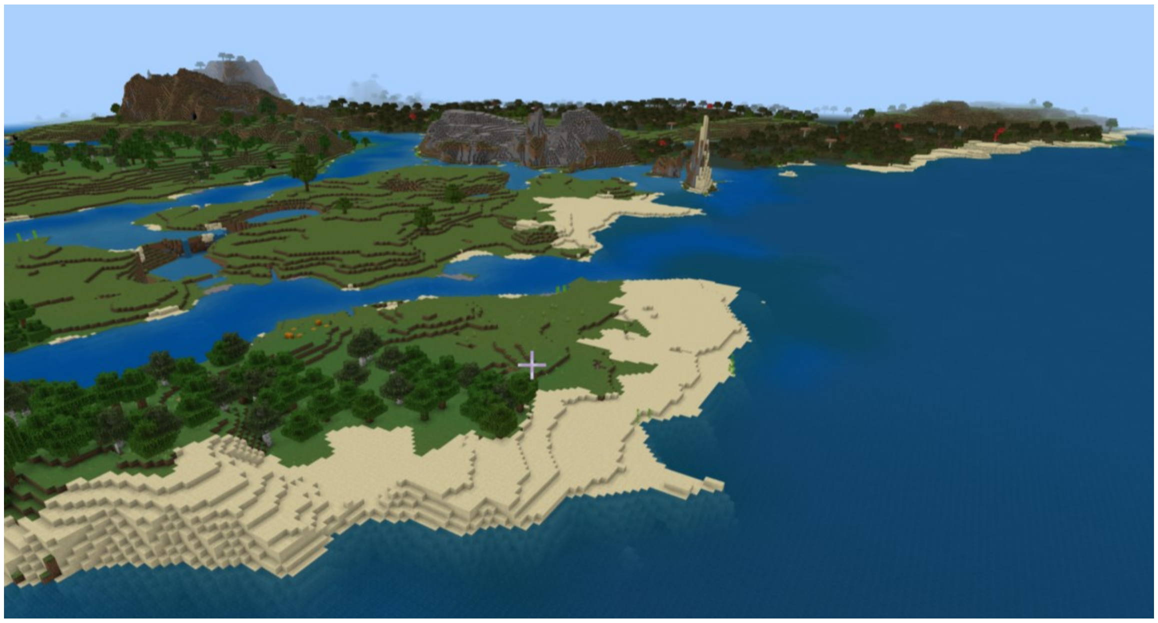


Figure 2 A screenshot of Minecraft showcasing its diverse biome system based on Whittaker diagram

Screen Shot, Mojang, 2024. Accessed April 24, 2024.

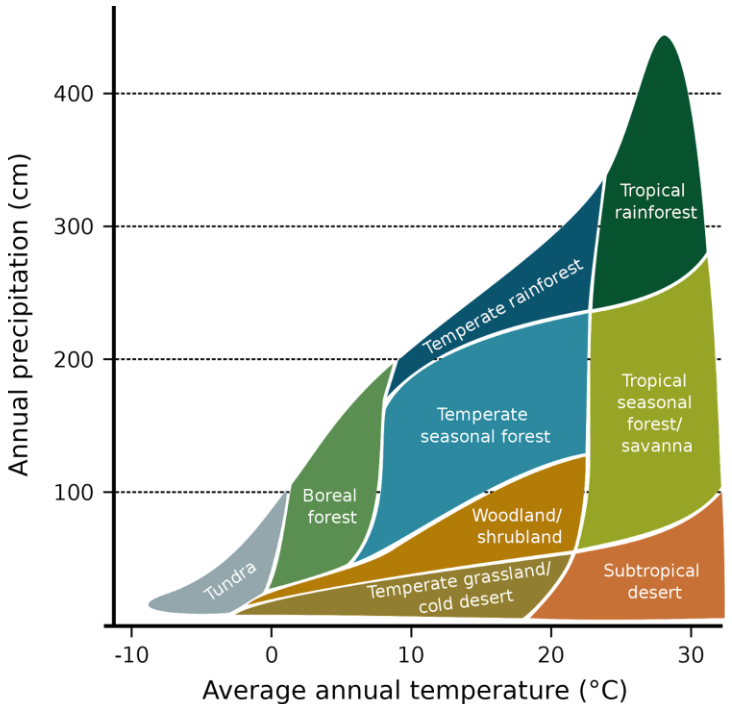


Figure 3 Whittaker biome diagram demonstrates distribution of vegetation based on temperature and precipitation.

Frankemann, CC BY-SA 4.0 <https://creativecommons.org/licenses/by-sa/4.0>, via Wikimedia Commons

2.2. Barriers and Challenges

While PCG helped numerous video games to scale up art production in the development pipeline (Chia, A, 2022), there are challenges and barriers that remain unresolved with current technology. In "Procedural Content Generation in Games" (Shaker et al., 2016), the authors listed out three visions that highlight the limitations of PCG and present interesting research opportunities for the future. The three key problems discussed by the authors are multi-level/multi-content PCG, PCG-based game design, and generating complete games with PCG.

2.2.1. Multi-level, multi-content PCG

This concept refers to the creation of a comprehensive set of game elements, such as lore, quests, characters, items, terrain, using procedural content generation within a game's ruleset and engine (Shaker et al., 2016). *No Man's Sky* (2016) is a notable example of the use of PCG to address such issues (Figure 5). The game features a universe-scale exploration experience that heavily relies on PCG, with pre-built algorithms creating its environments, from large assemblies such as spaceships and alien creatures, to smaller details such as planetary vegetation (Tait, E. R., & Nelson, I. L. 2022). While this approach aims to increase variety and replayability, it can result in content that feels repetitive and predictable, diminishing player engagement in the playthrough. The vastness and uncontrollability of these elements can sometimes result in a world that feels lifeless and insignificant to players, ultimately affecting their interest and immersion in the game (D. Heaven, 2016).

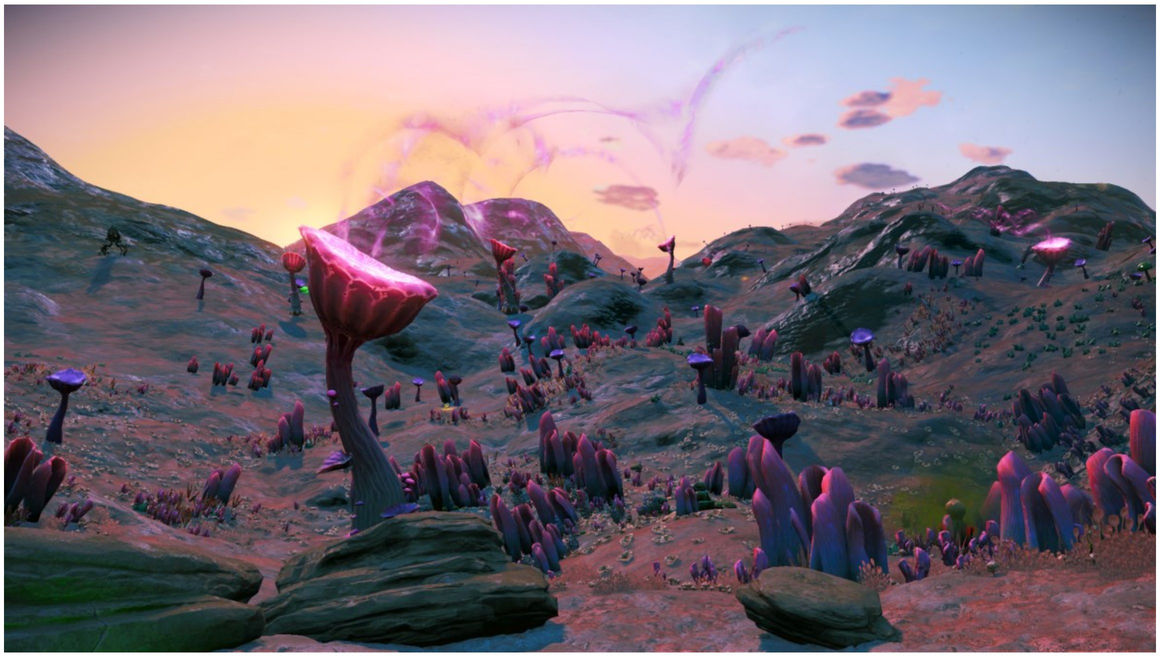


Figure 4 **Screenshot from *No Man's Sky* showcasing 3D models randomly generated on terrain.**

Screen Shot, Hello Games, 2024. Accessed April 25, 2024

2.2.2. PCG-Based Game Design

PCG-based game design refers to the design and creation of video games whose core mechanics are based on PCG. This means that the game, or the genre as a whole, would be fundamentally different or would not exist without PCG (Shaker et al., 2016). Many games use PCG to generate content alongside handcrafted assets, but the core gameplay loop remains independent of PCG (Togelius et al., 2013). For example, while No Man's Sky is known for its rich exploration powered by PCG, it could still be playable with hand-built assets. However, the integration of PCG as an essential part of the game design requires a new approach to both the game design and the parameters of the PCG algorithm, which should respond to the player's actions in a meaningful way (Togelius et al., 2013).

Regarding the vision of using PCG in the core loop of the game, Galactic Arms Races, created by Evolutionary Complexity Research Group at University of Central Florida, features a unique algorithm named content-generating NeuroEvolution of Augmenting Topologies (cgNEAT) that gradually evolves player's weapons system based on player's performance (Figure 6) (Hastings et al, 2009).

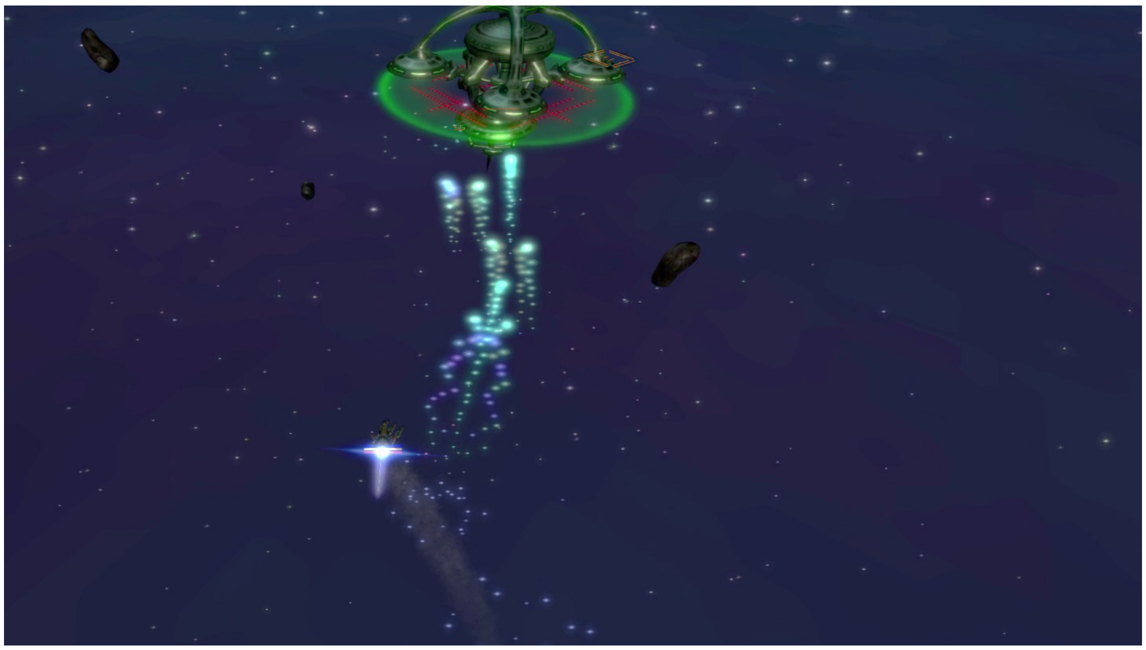


Figure 5 Screenshot of *Galactic Arms Races* showcasing player's weapon

Screen Shot, © Evolutionary Games 2014. Accessed April 26, 2024.

2.2.3. Generate Complete Games

Beyond generating in-game content, this vision refers to a game generator that will generate complete games, including game mechanics, levels, and game assets (Shaker et al., 2016). Generating complete games is considered to be much harder than generating content for existing games (Togelius et al., 2014) and requires encoding existing game rules into a format that can be processed by software systems and could be a challenging task without specifying the category of the game. A feasible starting point is to pick a specific genre and allow the game generator to generate variants of the game (Shaker et al., 2016). While many approaches focus on generating different rules though a predefined algorithm, the rules generated from the algorithm are rather simple compared to a man-made game (Shaker et al., 2016).

2.3. Conclusion

These three challenges illustrate the current limitations of PCG in creating diverse, engaging, and fully integrated game experiences. Multi-level/multi-content PCG struggles with generating a comprehensive set of game elements that feel cohesive and meaningful to players. PCG-based game design aims to make PCG an essential part of the core gameplay loop, requiring new approaches to game design and PCG algorithms. Finally, generating complete games with PCG remains a daunting task, as it involves encoding complex game rules and mechanics into a format that can be processed by software systems.

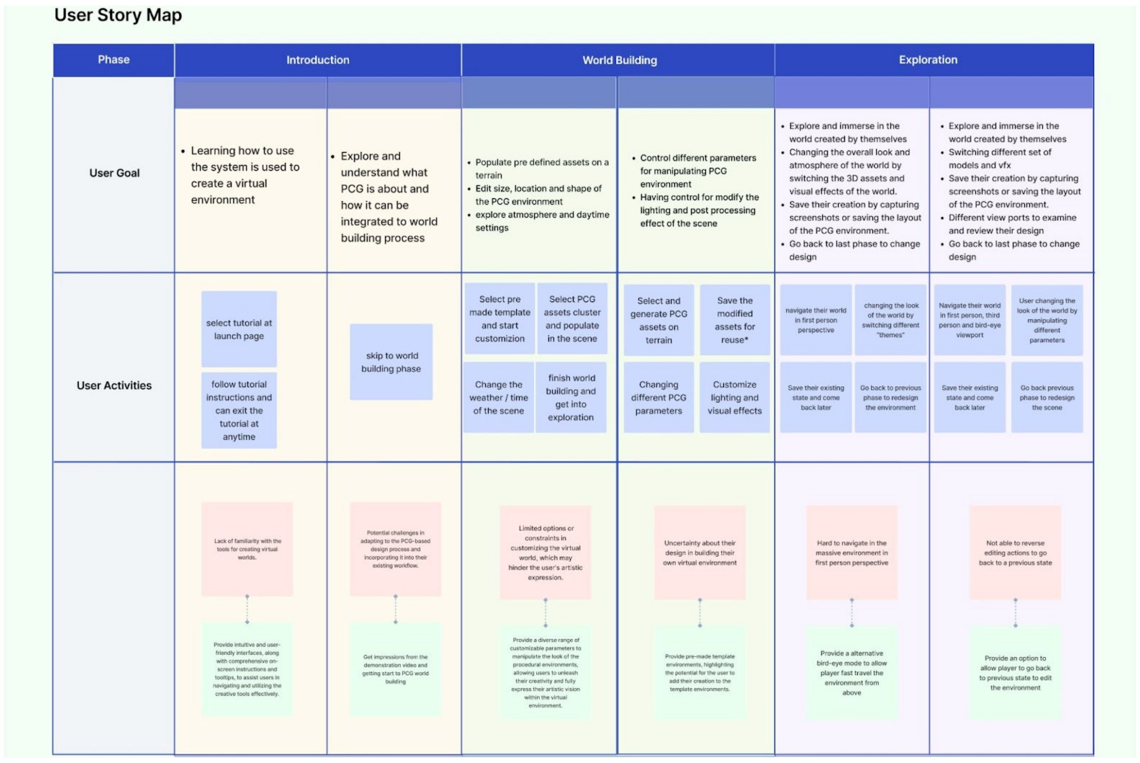
In the following chapter, I will discuss how "Brief Dreams" addresses the first two challenges by developing a runtime PCG framework that utilizes PCG as an essential feature of the program, enabling users to create diverse and engaging virtual environments through user-friendly interfaces.

Chapter 3. Methodology and Project Iteration

The methodology being used "Brief Dreams" is Research Creation. A curiosity-drive, interdisciplinary approach that transcends the boundaries between practice and theory and (Loveless, 2019). This method will allow me to utilize the artifact I developed and the documentation from the developing process as my project/research outcome. In developing this interactive experience, I centered around an iterative and incremental model, incorporating agile development principles. This approach is particularly suited for developing the PCG framework, allowing for the modular development of world-building components. Each component undergoes cycles of planning, implementation, and self review to ensure continuous refinement based on visual feedback and technical demands.

3.1. Planning

In planning the development of Brief Dreams, I wrote several user stories for my target users: Beginners and novices to the virtual world building and more experienced intermediate environmental artists and level designers in the video game industry (Figure 7). These user stories tell the user goals for three phases of the project: Introduction, World Building and Exploration.



User Story Map

Phase	Introduction		World Building		Exploration	
User Goal	• Learning how to use the system is used to create a virtual environment	• Explore and understand what PCG is about and how it can be integrated to world building process	• Populate pre defined assets on a terrain • Edit size, location and shape of the PCG environment • explore atmosphere and daytime settings	• Control different parameters for manipulating PCG environment • Having control for modify the lighting and post processing effect of the scene	• Explore and immerse in the world created by themselves • Changing the overall look and atmosphere of the world by switching the 3D assets and visual effects of the world. • Save their creation by capturing screenshots or saving the layout of the PCG environment. • Go back to last phase to change design	• Explore and immerse in the world created by themselves • Switching different set of models and vfx • Save their creation by capturing screenshots or saving the layout of the PCG environment. • Different view ports to examine and review their design • Go back to last phase to change design
User Activities	select tutorial at launch page follow tutorial instructions and can exit the tutorial at anytime	skip to world building phase	Select pre made template and start customizion Select PCG assets cluster and populate in the scene Change the weather / time of the scene finish world building and get into exploration	Select and generate PCG assets on terrain Save the modified assets for reuse* Changing different PCG parameters Customize lighting and visual effects	navigate their world in first person perspective changing the look of the world by switching different "themes" Save their existing state and come back later Go back to previous phase to redesign the environment	Navigate their world in first person, third person and bird-eye viewport User changing the look of the world by manipulating different parameters Save their existing state and come back later Go back previous phase to redesign the scene
	Lack of familiarity with the tools for creating virtual worlds.	Potential challenges in adapting to the PCG-based design process and incorporating it into their existing workflow.	Limited options or constraints in customizing the virtual world, which may hinder the user's artistic expression.	Uncertainty about their design in building their own virtual environment	Hard to navigate in the massive environment in first person perspective	Not able to reverse editing actions to go back to a previous state
	Provide intuitive and user-friendly interfaces, along with comprehensive on-screen instructions and tooltips, to assist users in navigating and utilizing the creative tools effectively.	Get impressions from the demonstration video and getting start to PCG world building	Provide a diverse range of customizable parameters to manipulate the look of the procedural environments, allowing users to unleash their creativity and fully express their artistic vision within the virtual environment.	Provide pre-made template environments, highlighting the potential for the user to add their creation to the template environments.	Provide a alternative bird-eye mode to allow player fast travel the environment from above	Provide an option to allow player to go back to previous state to edit the environment

Figure 6 **There are three main phases in my project: Introduction, World Building and Exploring. I break down each phase to specify and identify the features of the project.**

Yiming Zheng, 2024

In the introduction phase, to give users a first impression of the project. I'm planning to start with a landing page with visual elements including pre-captured screenshots of the PCG world and an introductory video to provide a short tutorial for beginners to get familiar with the user interfaces. More experienced users with professional backgrounds can skip the tutorial and start building the world with PCG directly.

The world building phase is the main phase of the project. In this section my goal was for the user to be able to create virtual environments by selecting 3D assets and populating them with PCG clusters on the provided terrain through a simple drag and drop mouse user interface. They would be able to modify the properties of the generated assets, including shapes, scales, and rotation of the PCG clusters. In addition, users can select one of the predefined lightings/seasonal settings to change the atmosphere of the scene. For beginners, they could start with predefined world templates, customize and blend their design to the scene. Advanced users can choose to expose more parameters to manipulate PCG environments, such as adding waypoints to the road, randomizing seeds for PCG, defining the density of certain 3D assets in the PCG clusters, etc.

After learning pain points and opportunities from the user story map, I created a user flow map to design the program workflow and identify various features to implement in the development phase (Figure 8).

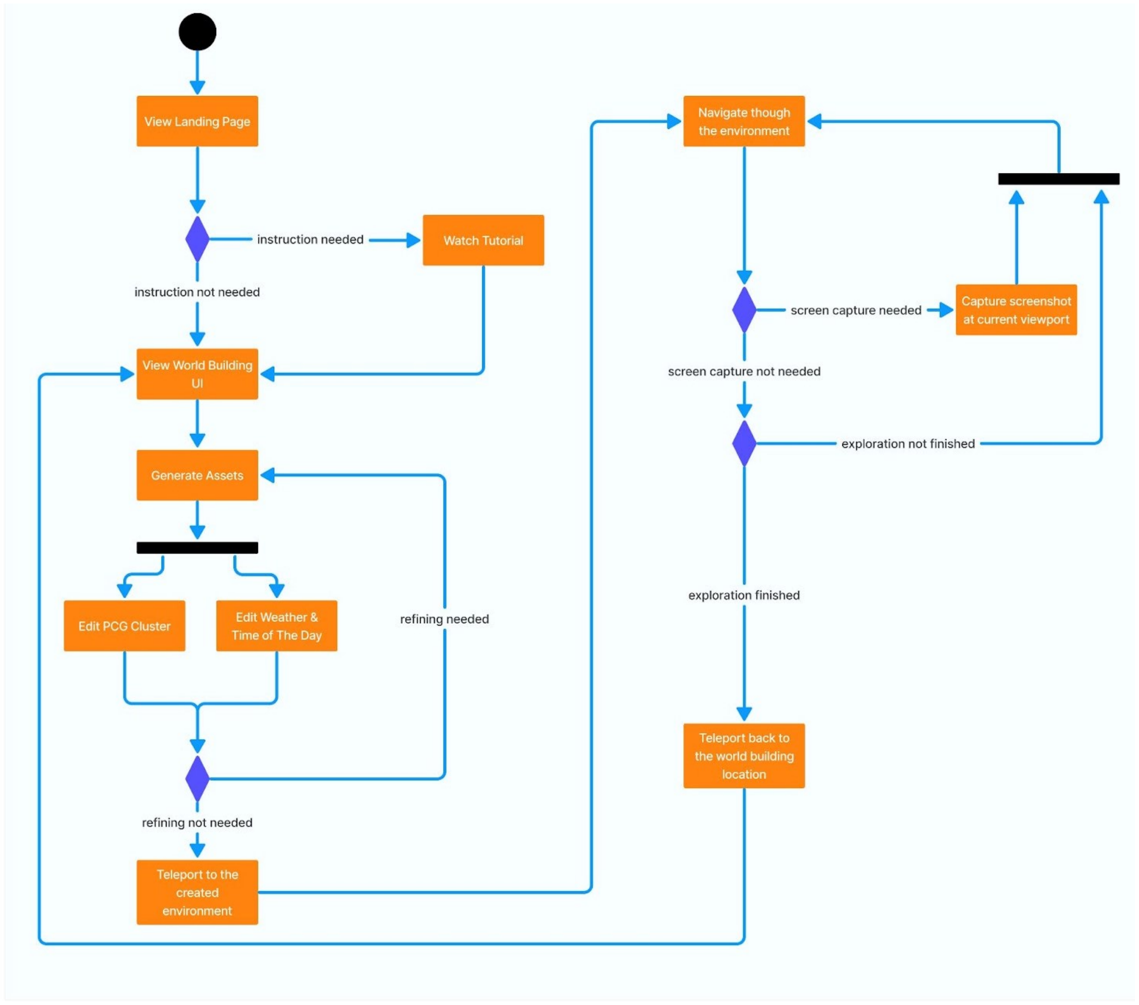


Figure 7 **The user flow map shows the basic workflow of the project and identifying the interaction loops in different phases of the experience**

Yiming Zheng, 2024

Furthermore, I created mood boards to narrow down the aesthetics of the procedural contents for selecting 3D assets and building visual effects (Figure 9). These mood boards are curated collections of images, notes, 3D scans from various online sources and personal documentation of real-world environments (nature reserves, housing developments, parks, etc.) in the Metro Vancouver Area.

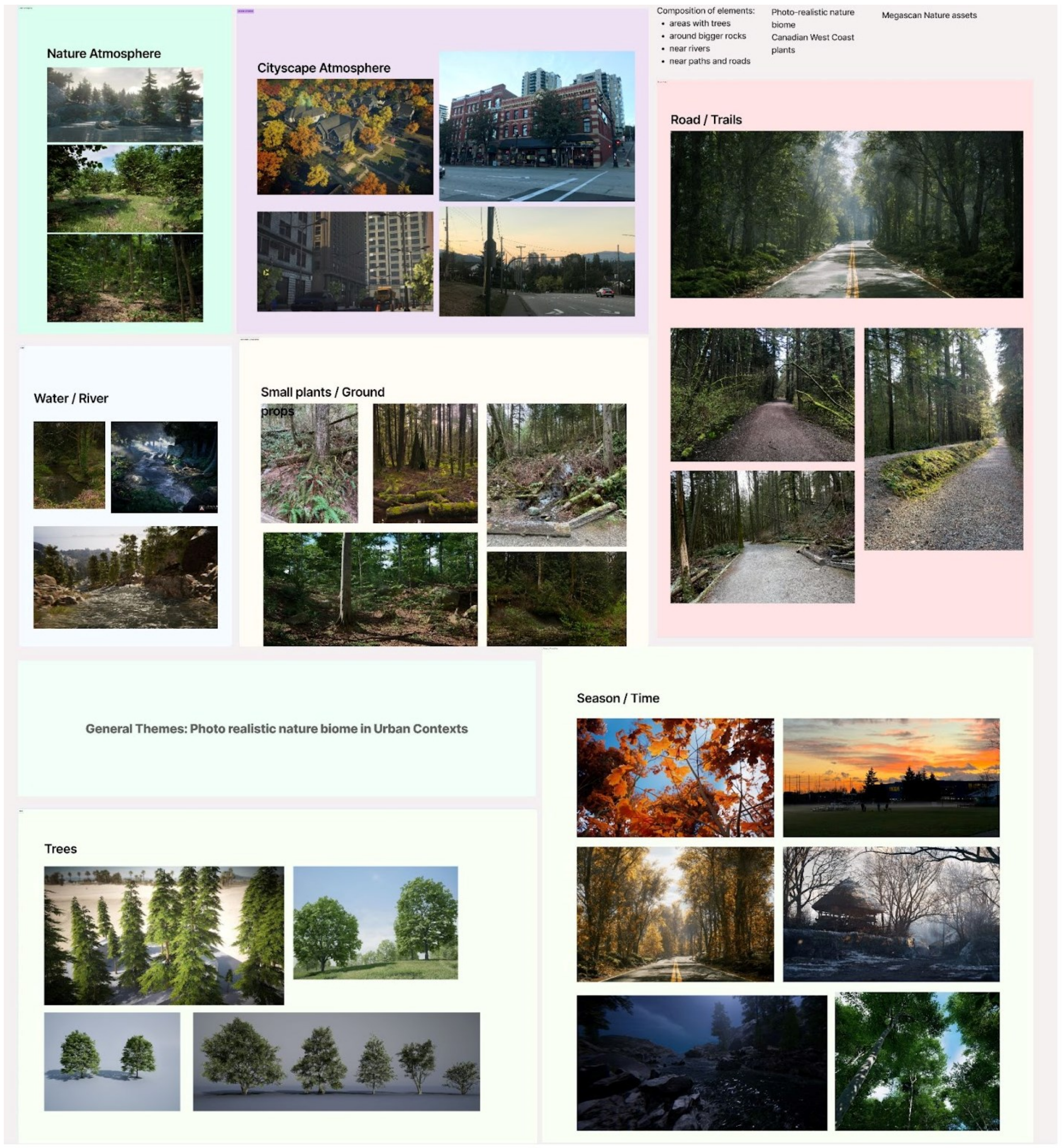


Figure 8 **These mood boards serve as a visual guide for the composition of the procedural environments to be developed, as well as a reference for the selection of individual 3D models.**

Yiming Zheng, 2024

In selecting 3D assets, it is not only to make the assets aesthetically consistent with the mood board reference, but also to ensure that individual models don't have a large impact on performance, as procedural content will be generated repeatedly in the world. In addition to providing aesthetic direction, the mood boards ensure that the procedural content remains grounded in real-world references, enhancing the authenticity and relatability of the virtual environments. This phase is critical to establishing a clear, cohesive vision for the project and aligning subsequent development phases with the original creative intent.

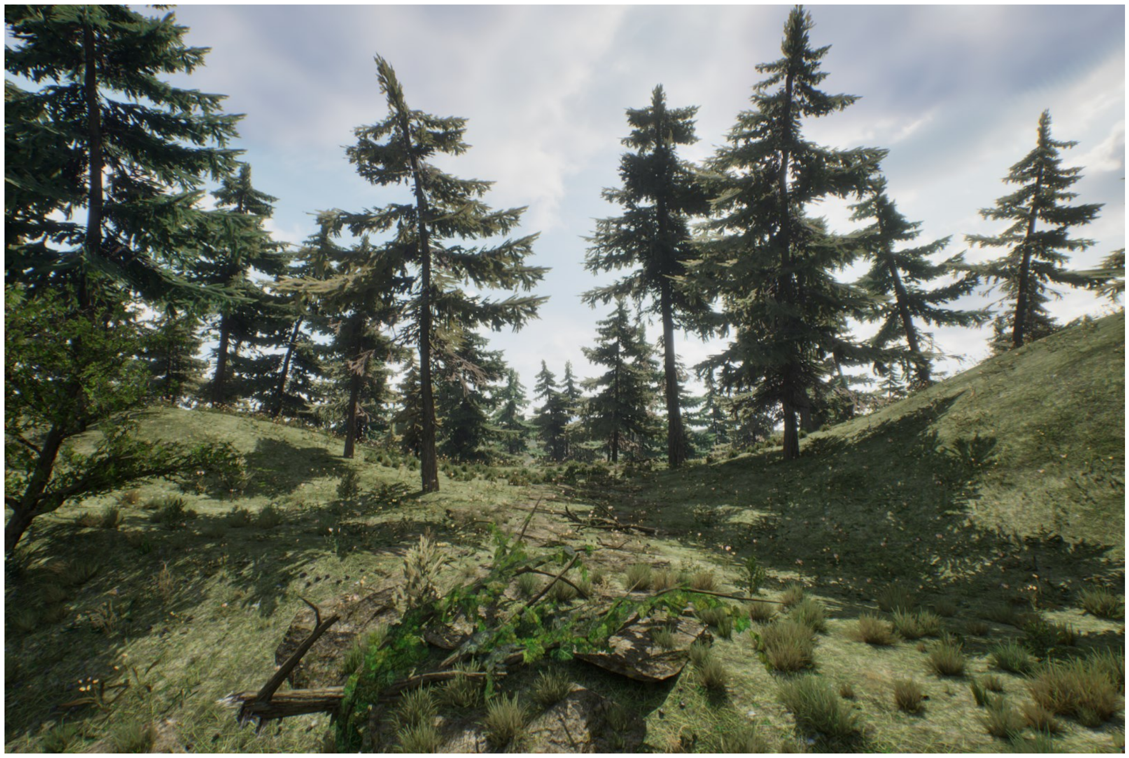


Figure 9 Screenshot of PCG forest overlapping with PCG trail in first iteration of *Brief Dreams*

Yiming Zheng, 2024

3.2. Design and implementation of PCG Cluster

The implementation phase translated the vision established in the planning phase into adaptive procedural elements within the Unreal Engine. This phase involved the use of visual scripting language (PCG Graph and Blueprints) to define the algorithm for populating 3D assets within the PCG clusters.

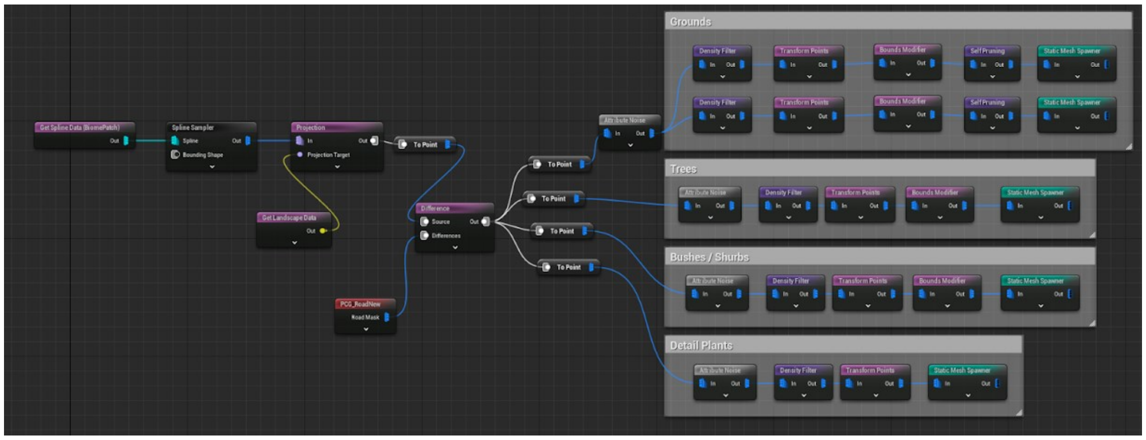


Figure 10　**PCG Graph is a visual scripting language to create algorithms for PCG clusters. In this graph I created the algorithm for a forest biome.**

Yiming Zheng, 2024

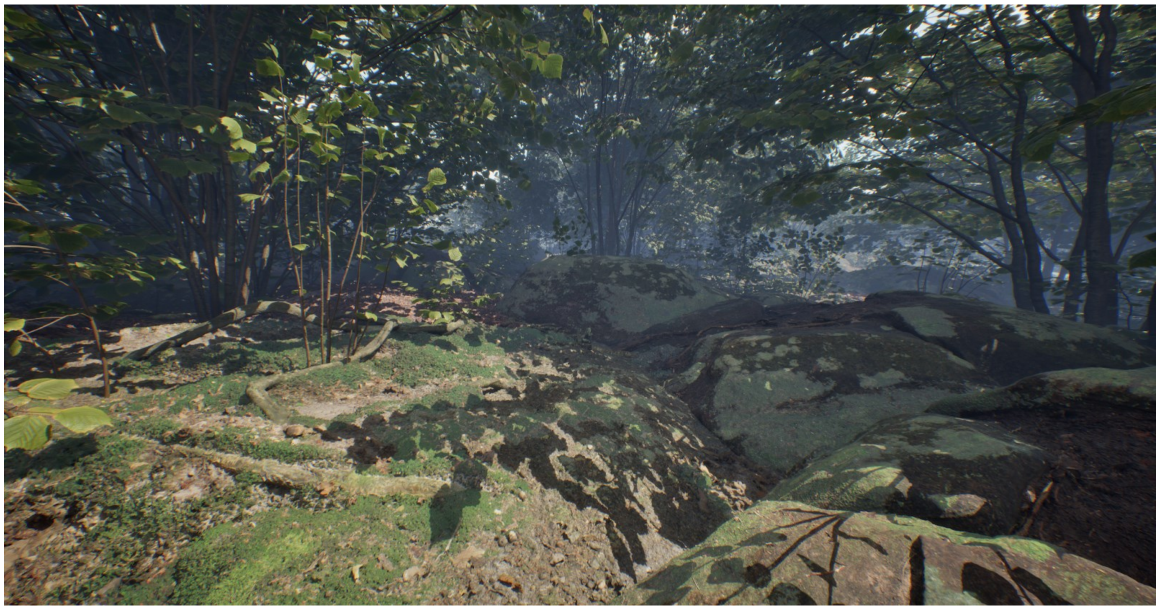


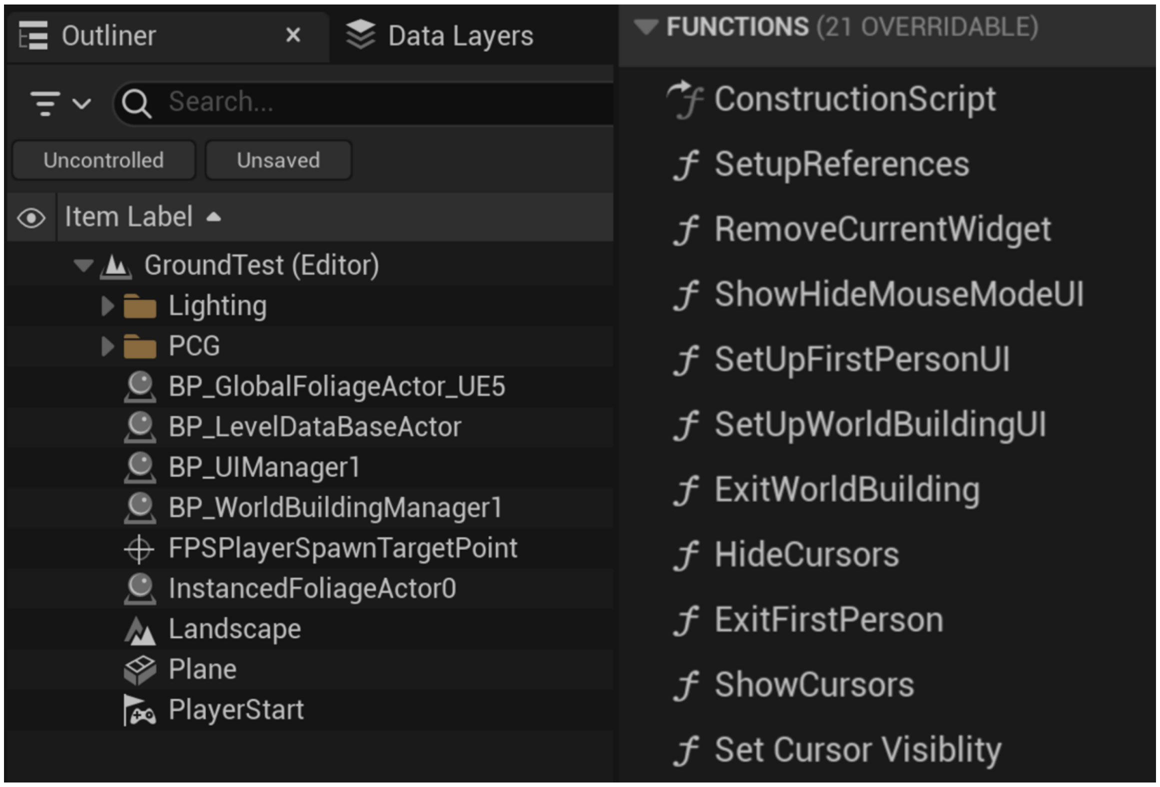
Figure 11　**In the forest biome created from the PCG graph in Figure 11, I generate a variety of ground plates above the terrain to seamlessly blend foliage into the ground environment.**

Yiming Zheng, 2024

The implementation phase was iterative in nature, allowing adjustments and refinements to be made as needed to ensure that the procedural content is seamlessly aligned with the mood board design and fully functional for generation in the environment.

3.3. Singleton Pattern

In designing the game architecture for this project, I used the singleton pattern that restricts variables, instance related functions to a singular instance that is created to manage a specific field of functions. This design pattern requires an initialization for each singleton instance and permit call-by-need which provides a relatively clear and straightforward approach in developing core functions for the game (Figure 13, 14).



GroundTest (Editor)
Lighting
PCG
BP_GlobalFoliageActor_UE5
BP_LevelDataBaseActor
BP_UIManager1
BP_WorldBuildingManager1
FPSPlayerSpawnTargetPoint
InstancedFoliageActor0
Landscape
Plane
PlayerStart

ConstructionScript
SetupReferences
RemoveCurrentWidget
ShowHideMouseModeUI
SetUpFirstPersonUI
SetUpWorldBuildingUI
ExitWorldBuilding
HideCursors
ExitFirstPerson
ShowCursors
Set Cursor Visiblity

Figure 12 Unreal Editor outliner showcasing singletons are instanced in the game and functions within the "BP_UIManager" singleton.

Yiming Zheng, 2024

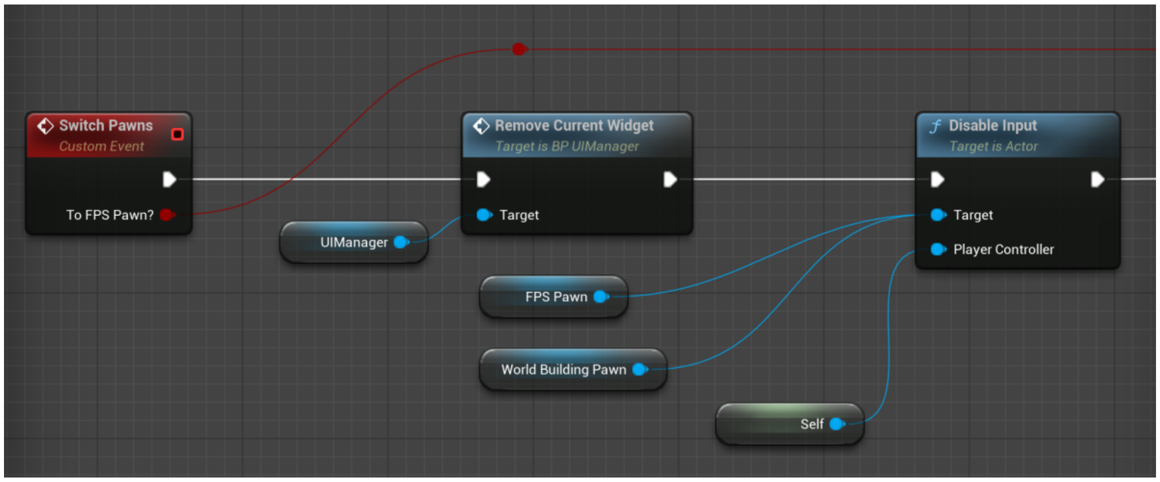


Figure 13 **Custom event in BP_MainPlayerController singleton that performs viewport transition and called functions from other singletons.**

Yiming Zheng, 2024

Chapter 4. Artifact Description

4.1. Development of PCG Clusters

The main feature of Brief Dreams is to enable users to generate 3D assets with predefined algorithms captured in a visual script over any existing terrain. The development of the PCG clusters requires the use of the experimental procedural content generation framework plugin in Unreal Engine 5.3. The visual description of the algorithm is composed of six clusters (Figure 15): (a) road generator, (b) river generator, (c) desert generator, (d) campsite generator, (e) forest generator, (f) tower generator. The road and river generators populate assets along a path, while forest, desert and campsite generators populate assets in a user defined perimeter. The tower can be inserted to the scene at any point on the terrain. This framework is flexible enough to add different terrain characteristics later on.

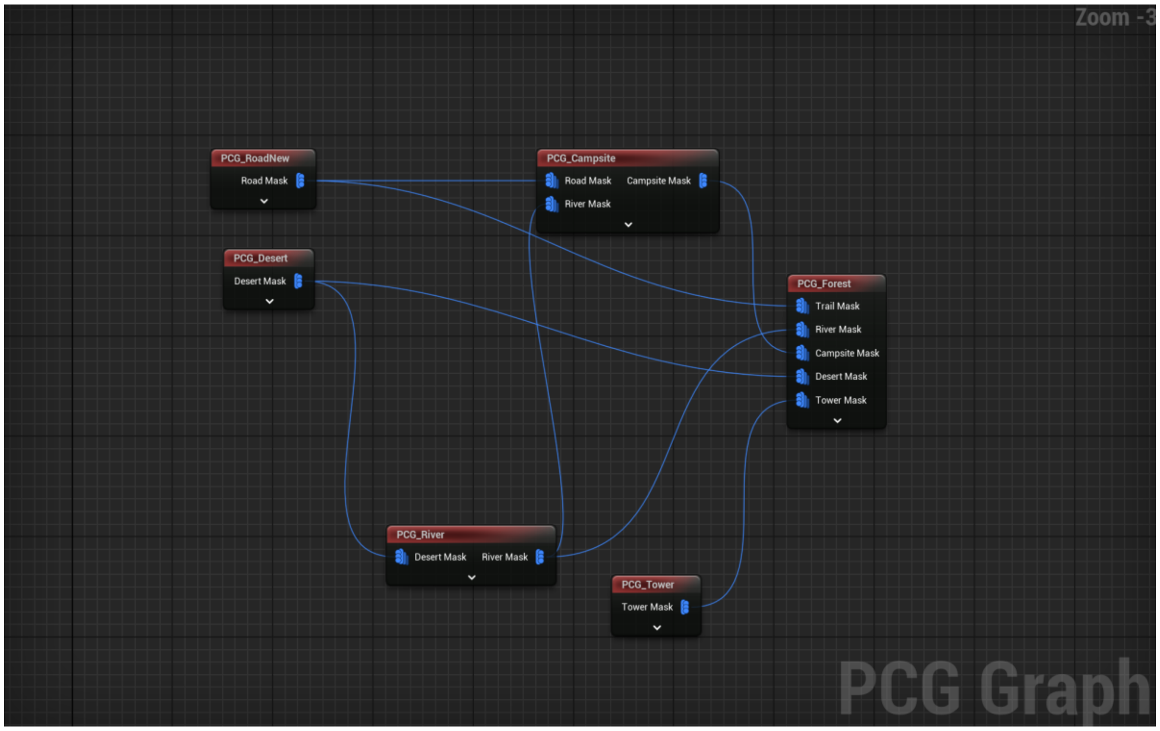


Figure 14 The main PCG graph for Brief Dreams, which I used to transfer mask data between different clusters to avoid overlapping.

Yiming Zheng, 2024

The following describes the forest generation clusters in detail (Fig 14). In the forest PCG cluster the algorithm receives the perimeter of the forest as a closed spline curve defined by the user. The spline sampler and project nodes generate point grids from the spline data within the forest spline loop and then project and snap to the terrain. The forest graph also receives multiple point grids as masks from other PCG clusters to constrain forest generation in the projected curve on the terrain and over the random point grid again projected on the terrain. This process is mainly performed in the Projection and Input nodes (Figure 11).

To be able to generate and manipulate different patterns of point grids and later generate 3D meshes on top of coordinates of these points, Brief Dream visualizes the point grids on different nodes, and generates placeholder cubes with in grayscales that represent the transformation and density parameter of each point in the grids (Figure 16).

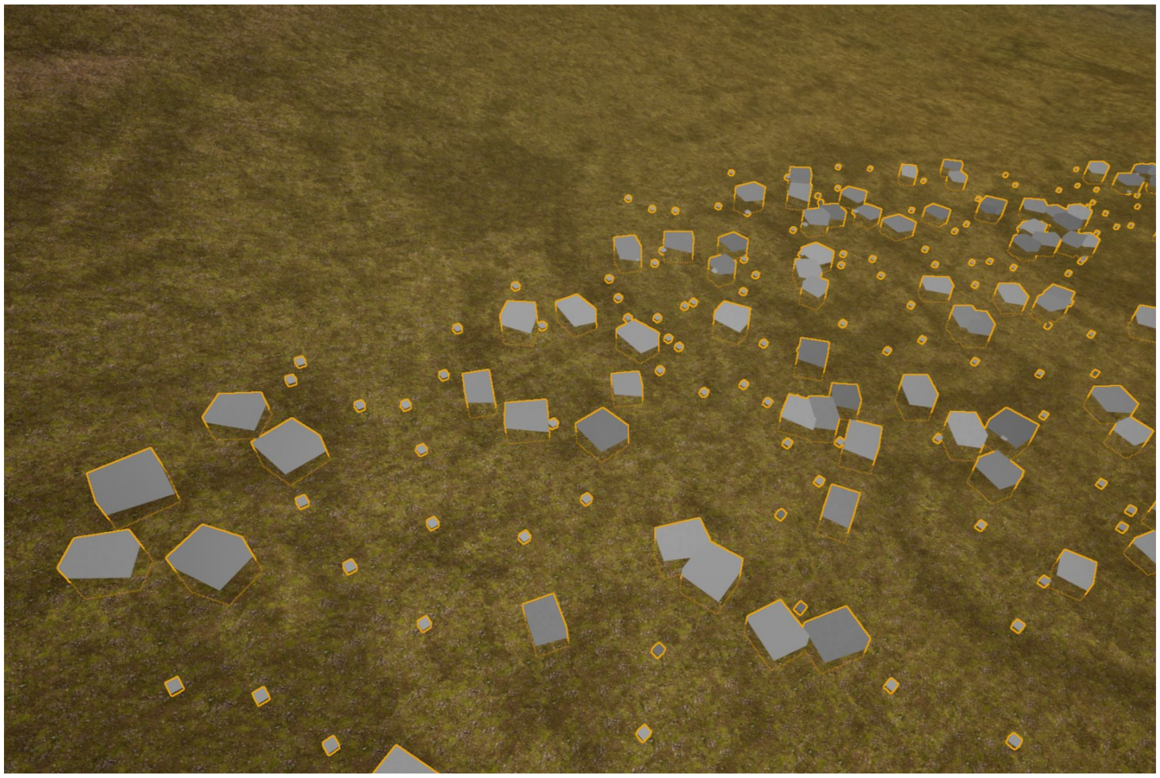
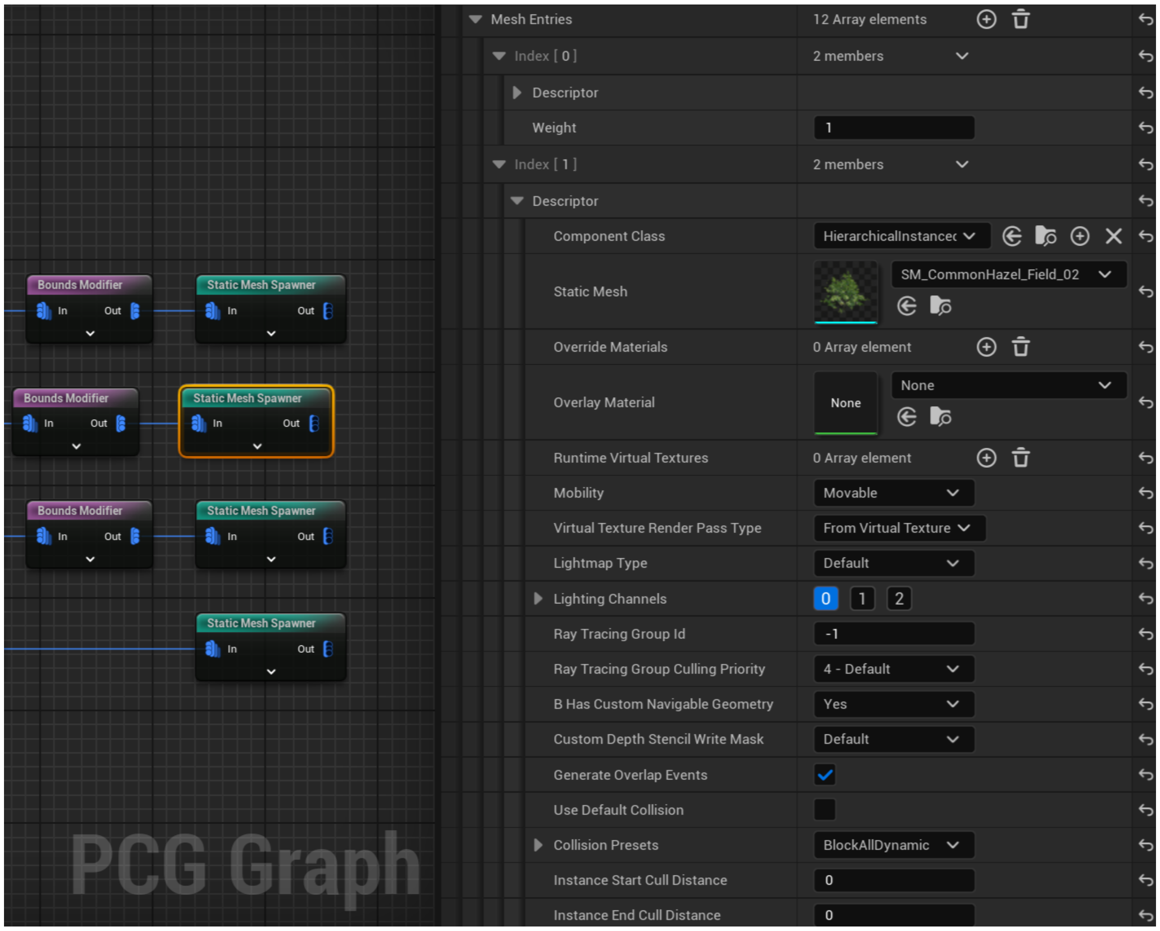


Figure 15 **PCG forest cluster in debug mode. Large cubes represent point grids for generating small to medium 3D assets such as shrubs. Small cube represents the coordinate for generating the ground meshes.**

After creating and manipulating the point grids, static mesh spawner nodes are added to substitute points with 3D assets such as trees, shrubs, mushrooms (Figure 17).



Mesh Entries	12 Array elements
Index [0]	2 members
Descriptor	
Weight	1
Index [1]	2 members
Descriptor	
Component Class	HierarchicalInstancec
Static Mesh	SM_CommonHazel_Field_02
Override Materials	0 Array element
Overlay Material	None
Runtime Virtual Textures	0 Array element
Mobility	Movable
Virtual Texture Render Pass Type	From Virtual Texture
Lightmap Type	Default
Lighting Channels	0 1 2
Ray Tracing Group Id	-1
Ray Tracing Group Culling Priority	4 - Default
B Has Custom Navigable Geometry	Yes
Custom Depth Stencil Write Mask	Default
Generate Overlap Events	✓
Use Default Collision	
Collision Presets	BlockAllDynamic
Instance Start Cull Distance	0
Instance End Cull Distance	0

Figure 16 A screen shot of forest PCG graph showcasing the static mesh spawner node.

Yiming Zheng, 2024

In this node, the users can add different 3D assets to the static mesh array, set attributes such as collision and material to the meshes, and assign the weight parameter to set the probability of selected 3D assets being generated in the cluster. Figure 18 shows a PCG cluster spline projected on a terrain defining forest boundary which is populated by various 3D assets that can usually be seen in a forest.

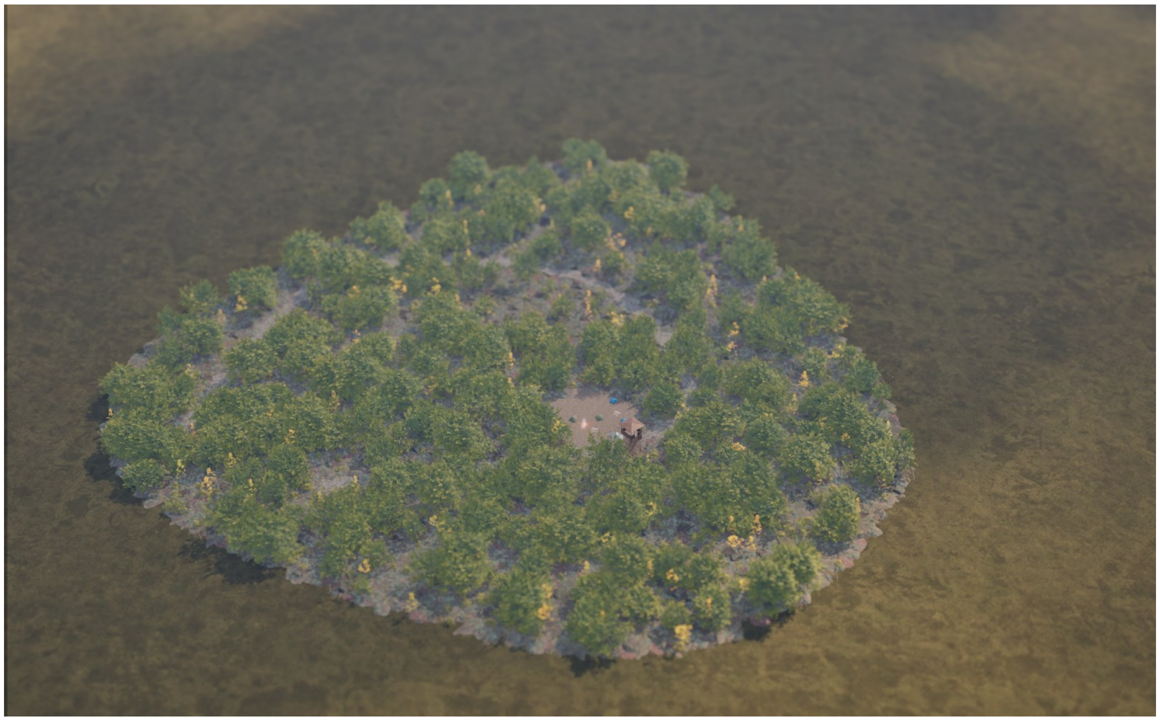


Figure 17 **A Top-down view of forest PCG cluster. Foliage, ground plate and props are populating on point grids. Trail spline and campsite cluster are being masked out to avoid overlapping with the forest cluster.**

Yiming Zheng, 2024

4.2. Brief Dreams Interfaces

4.2.1. World Building UI

The goal of the user interface in Brief Dreams is to enable users to generate and manipulate procedural contents at runtime and turning the application into a game for building virtual worlds.

The World Building interface allows users to create virtual environments from a top-down or bird-eye view (Figure 19). A user interface prototype was created to test the runtime capabilities and direct input of the PCG clusters. In this prototype, users can drag and drop 2D widgets from the left column onto the terrain using the mouse cursor, which generates the corresponding PCG cluster that is linked to the widget. A size slider at the bottom of the interface will allow users to adjust the scale of the PCG cluster.

Additionally, three buttons on the right side enable users to change the 3D models within the PCG, which completely alter the visual appearance of the scene (Figure 20).

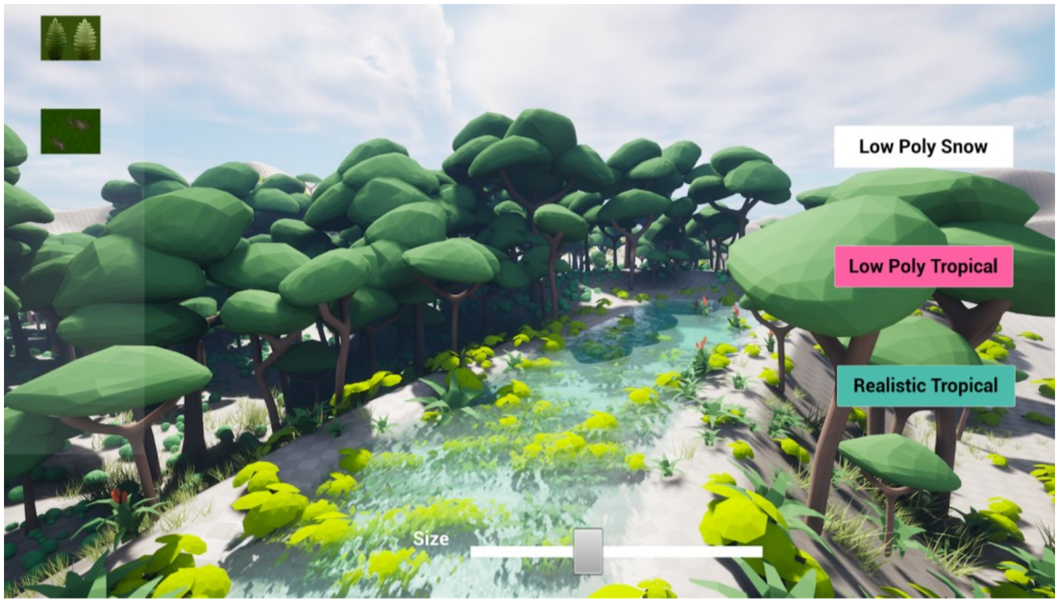


Figure 18 **An initial UI prototype that allows the user to generate procedural content at run time.**

Yiming Zheng, 2024

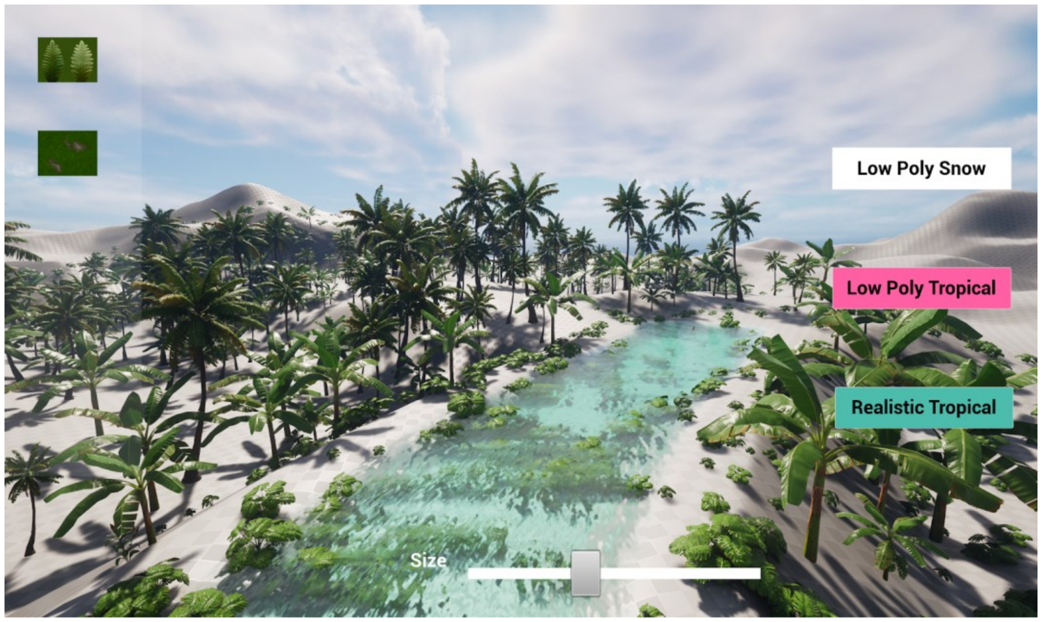


Figure 19 **Selecting the button on the right side of the screen will change the 3D models in the PCG cluster to a different set of 3D models.**

Yiming Zheng, 2024

To update PCG clusters after making changes at runtime, an update PCG function was created to clean up all procedural generated content in the scene and regenerate the content with updated changes (Figure 21). This function is called whenever the user modifies the PCG content in the scene.

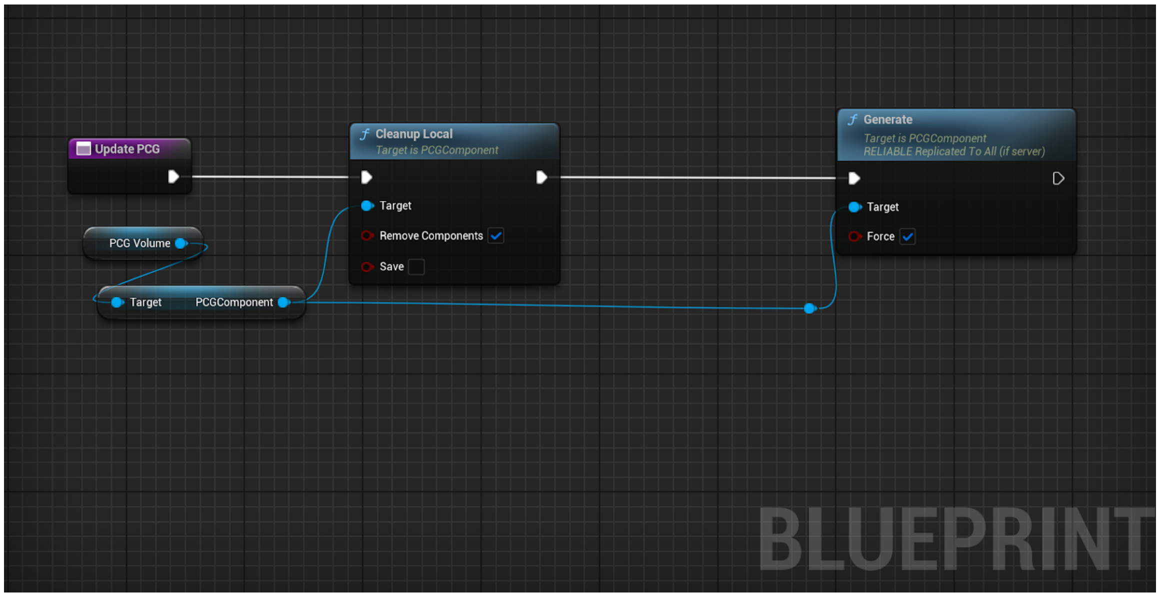


Figure 20 Update PCG function in the MainPlayerController. This function is called when the user changes the PCG content in the scene.

Yiming Zheng, 2024

In the later stage of the user interface development, various cursor shapes were introduced in the world-building phase to enable users to perform drag-and-drop functions for modifying the location and shape of the PCG clusters and road splines (Figure 22). To effectively highlight the changes made to the PCG clusters and splines in real-time, a guiding outline feature is introduced to dynamically render borders around the clusters and road splines by procedurally generating linear runtime spline meshes with emissive color materials. The guiding outlines provide users with immediate visual feedback of the modifications to the PCG clusters.

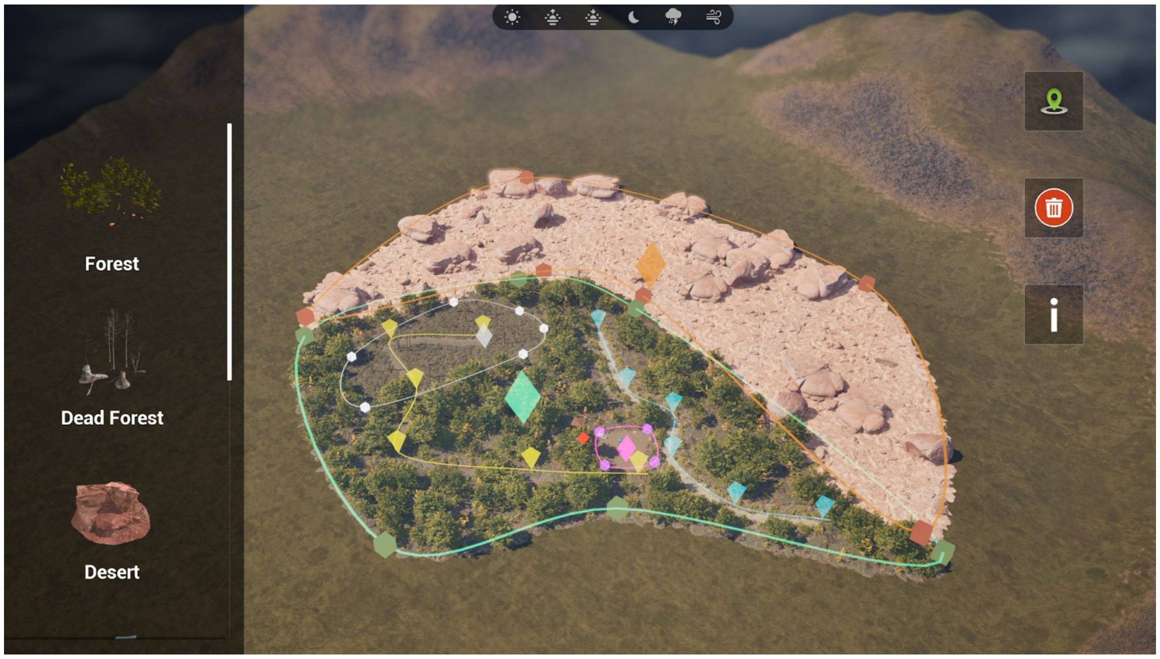


Figure 21 **User interface from the final prototype of Brief Dreams. Users will be able to drag and drop diamond and cubic shaped "cursors" to manipulate the location and shape of PCG clusters.**

Yiming Zheng, 2024

4.3. Character Controller Design and Development

To ensure smooth interaction and navigation within the game experience, two distinct character controller systems were introduced: a top-down, drag-and-drop controller for the world-building phase and a first-person controller for the exploration phase.

The top-down controller allows users to orbit around the center of the terrain from a bird's-eye perspective (Figure 23). This viewport enables users to generate procedural content through the user interface without constantly changing the camera location, providing an overview of the landscape.

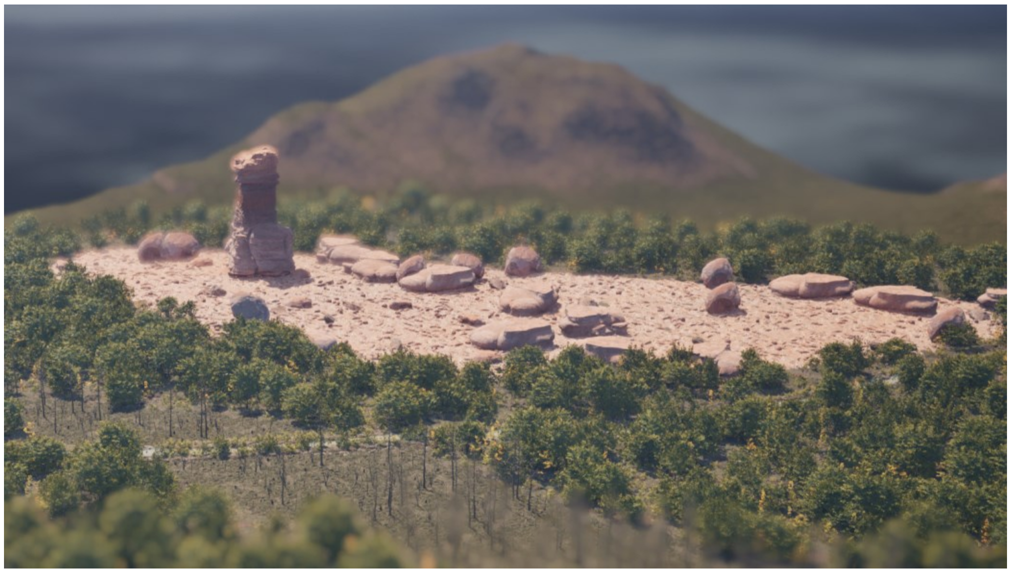


Figure 22 Screen capture from the world building phase, showcasing the top-down view.

Yiming Zheng, 2024

Once users have completed designing the environment in the world building phase, they can seamlessly transition to the first-person perspective by dragging the exploration icon into the scene. This action switches the control to the first-person controller (Figure 24), allowing users to explore the environment from an immersive, eye-level first person viewpoint (Figure 25). The first-person controller enables players to directly control the camera movement by using mouse and keyboard inputs, which provides an intuitive and engaging way to navigate though the world.

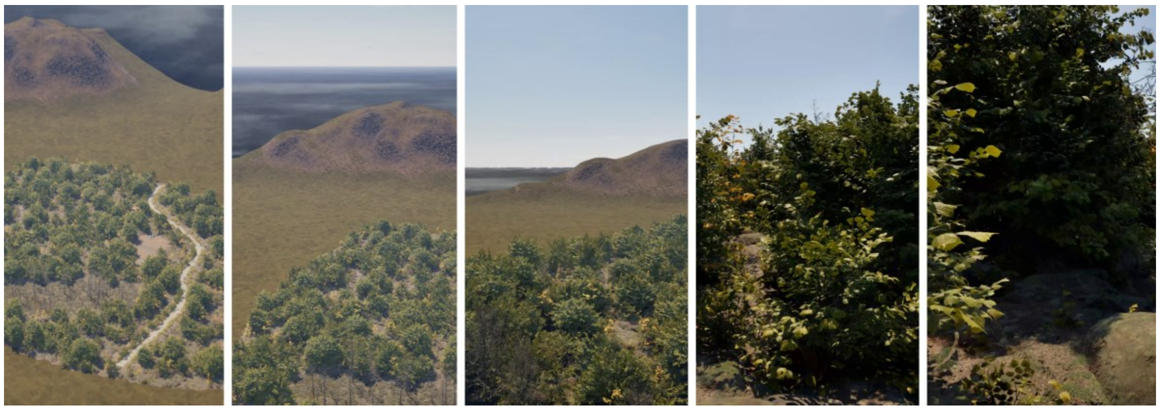


Figure 23 Keyframes from camera transition from world building view to first person view.

Yiming Zheng, 2024

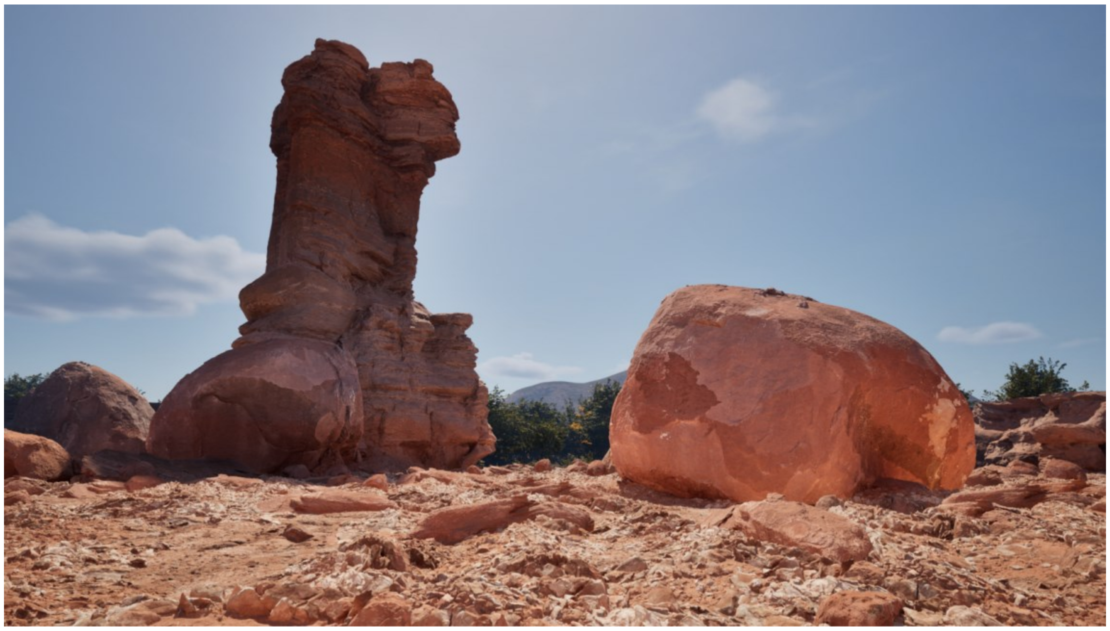


Figure 24 Screenshot captured during the exploration phase.

A first-person perspective allows users to immerse themselves in the photorealistic environment by navigating and interacting with the virtual world from an eye-level viewpoint. Yiming Zheng, 2024

4.4. Weather and Time of the Day

To enhance the visual diversity of the generated environments, a time and weather system is introduced to enable users to change the time of day and weather conditions using the "Ultimate Dynamic Weather" plugin (refer-website). Users can control the time and weather settings by selecting the corresponding buttons in the panel at the top of the screen (Figure 26). Each button is connected to the plugin and assigned a specific float value that adjusts the relevant parameters of the plugin. For example, selecting the “sun” button sets the time value of the plugin to 1200, representing midday. This feature offers users an additional layer of customization and realism, empowering them to explore the generated world under a variety of atmospheric conditions (Figure 27).

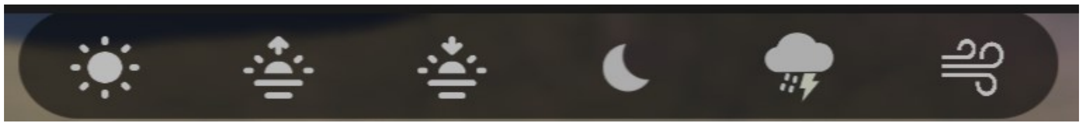


Figure 25 Time of day and Weather buttons for changing the visual effect of the scene.

Yiming Zheng, 2024

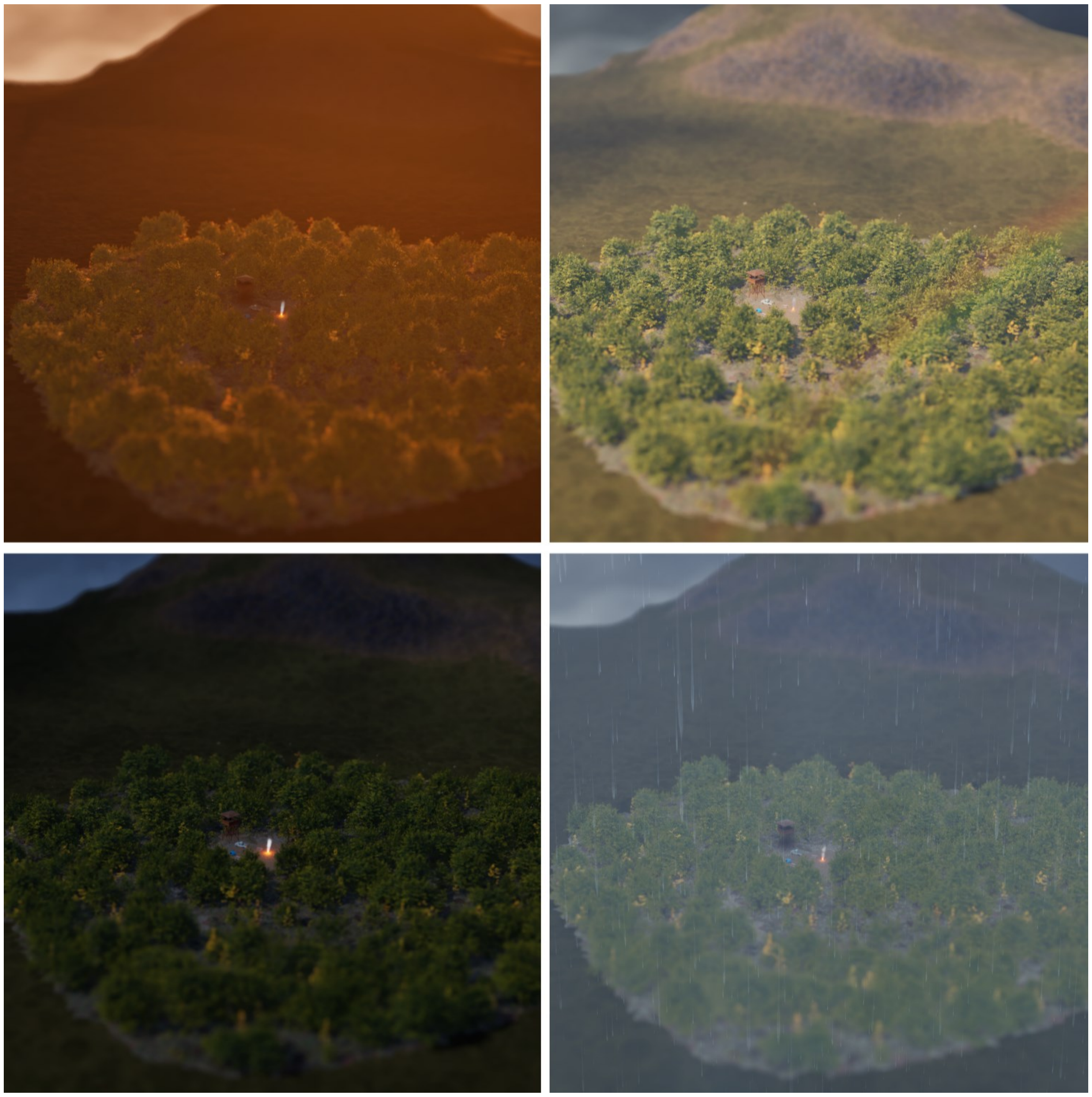


Figure 26 Screen captures showcasing the visual effects of different times of the day and weather conditions.

A video trailer for showcasing these features can be viewed at
https://yimingzheng.carrd.co/#briefdreams Yiming Zheng, 2024

Chapter 5. Discussion

5.1. Reflections

The development of Brief Dreams has demonstrated that the procedural content generation (PCG) framework can be integrated with a user-friendly interface, allowing users to have direct input to the PCG system and to create intermediate-scale, customizable 3D environments without the need for manual placement of 3D assets. The current prototype also showcases runtime capability in both the play mode of the Unreal Editor and packaged builds, suggesting that future development of the PCG framework could be treated as a core mechanic in world building games.

This project was originally planned to feature different sets of PCG environments and allow users to completely change the environment during the exploration phase, bringing awareness to current environmental hazards in BC, such as altering the forest cluster to a burning wildfire cluster with fire and audio effects. However, due to time constraints, this feature was not fully implemented and was partially altered to include a burnt forest cluster featuring static 3D assets of dead trees, charcoal, and rough grounds that can be placed on the terrain in the world building phase.

5.2. Future Work and Improvements

In the next iteration of Brief Dreams, the development will focus on enabling users to import custom 3D assets into the project and create personalized PCG clusters. This feature will significantly expand the creative possibilities for users without requiring modifications to the existing PCG cluster algorithms. Additionally, providing support for low-poly assets will improve the program's performance, ensuring a smooth and efficient user experience even with custom assets. A streamlined process for asset integration will be implemented, making it easy for users to import and utilize their desired 3D models within the Brief Dreams framework.

Another potential area for improvement is the manipulation of PCG cluster patterns. Implementing user-controlled parameters for adjusting the density and distribution of foliage, this could be achieved by exposing additional backend settings within the user interface, such as sliders or numerical inputs, which would directly influence the algorithms responsible for generating the PCG clusters.

During an internal critique session with research fellows in the SFU SIAT Making Culture Lab and criticalMediArtStudio, an intriguing question was raised regarding the possibility of generating 3D assets on top of any 3D model, such as a photogrammetry scan of a sculpture. This idea presents an opportunity to extend the capabilities of Brief Dreams beyond virtual world creation. By enabling the system to recognize and populate 3D assets on user-created 3D models, users could employ Brief Dreams to create new variations of their assets. For instance, users could procedurally generate moss and mushroom models on the surface of a tree trunk model. This functionality would allow users to enhance and modify their existing 3D assets with procedurally generated elements, opening new possibilities for customizing small scale assets.

Looking to the future, the PCG clusters in Brief Dreams could be extended to other types of content, such as animated characters, creatures, and props. By incorporating additional algorithms and user-defined parameters, the system could generate entire virtual worlds populated with diverse and dynamic elements, enhancing the versatility and functionality of the generated environments.

The development of Brief Dreams has been a rewarding and enlightening experience, showcasing the immense potential of procedural content generation in creating vast and immersive virtual worlds. As Brief Dreams continues to grow and evolve, I am excited to see how it will shape as a procedural world builder and inspire users to create and explore the endless possibilities of PCG-generated environments.

References

Adeel Zafar, Hasan Mujtaba, & Omer Beg. (2021). Procedural Content Generation for General Video Game Level Generation. Inteligencia Artificial, 24(68).

C. M. Kanode and H. M. Haddad. (2009) Software Engineering Challenges in Game Development, Sixth International Conference on Information Technology: New Generations, 2009, pp. 260-265, https://doi.org/10.1109/ITNG.2009.74.

Togelius, J., Kastbjerg, E., Schedl, D., & Yannakakis, G. (2011). What is procedural content generation? ACM International Conference Proceeding Series, 1–6. https://doi.org/10.1145/2000919.2000922

Shaker, N., Togelius, J., & Nelson, M. J. (2016). Procedural Content Generation in Games by Noor Shaker, Julian Togelius, Mark J. Nelson. (1st ed. 2016.). Springer International Publishing: Imprint: Springer.

Steward, J. (2023). Procedural Content Generation: Generating Procedural Game Content Using Machine Learning (Doctoral dissertation, California State University, Northridge).

Volkmar, G., Alexandrovsky, D., Eilks, A. E., Queck, D., Herrlich, M., & Malaka, R. (2022). Effects of PCG on Creativity in Playful City-Building Environments in VR. *Proceedings of the ACM on Human-Computer Interaction*, *6*(CHI PLAY), 1–20.

Gravina, D., Khalifa, A., Liapis, A., Togelius, J., & Yannakakis, G. N. (2019). Procedural Content Generation through Quality Diversity. 2019 IEEE Conference on Games (CoG), 2019, 1–8. https://doi.org/10.1109/CIG.2019.8848053

Lech, B., Azad, S., Welnitz, J., Jonasson, J., & Martens, C. (2021). Designing a Combined World and Story Procedural Content Generation Engine.

De Seta, A., & Alviano, M. (2022). An Application of ASP for Procedural Content Generation in Video Games. In CEUR WORKSHOP PROCEEDINGS (Vol. 3204, pp. 134-140). CEUR-WS.

Procedural Content Generation Wiki. (2012). Minecraft, Use of Procedural Content Generation Methods. http://pcg.wikidot.com/pcg-games:minecraft

Jain, A., Sharma, A., & Rajan, R. (2022). Adaptive & Multi-Resolution Procedural Infinite Terrain Generation with Diffusion Models and Perlin Noise. ACM International Conference Proceeding Series.

Procedural Content Generation Wiki. (2012). Whittaker Diagram, Use of Procedural Content Generation Methods. http://pcg.wikidot.com/pcg-algorithm:whittaker-diagram

Chia, A. (2022). The artist and the automaton in digital game production. *Convergence (London, England)*, *28*(2), 389–412

Tait, E. R., & Nelson, I. L. (2022). Nonscalability and generating digital outer space natures in No Man's Sky. Environment and Planning. E, Nature and Space (Print), 5(2), 694–718. https://doi.org/10.1177/25148486211000746

D. Heaven.(2016) "When infinity gets boring: What went wrong with No Man's Sky"

https://www.newscientist.com/article/2104873-when-infinity-gets-boring-what-went-wrong-with-no-mans-sky/.

Togelius, J., Champandard, A.J., Lanzi, P.L., Mateas, M., Paiva, A., Preuss, M., & Stanley, K.O. (2013). Procedural Content Generation: Goals, Challenges and Actionable Steps. Artificial and Computational Intelligence in Games.

Hastings, E. J., Guha, R. K., & Stanley, K. O. (2009). Automatic Content Generation in the Galactic Arms Race Video Game. IEEE Transactions on Computational Intelligence and AI in Games., 1(4), 245–263. https://doi.org/10.1109/TCIAIG.2009.2038365

Togelius, J., Nelson, M. J., & Liapis, A. (2014). Characteristics of generatable games. Proceedings of the FDG Workshop on Procedural Content Generation in Games, Fort Lauderdal.

Loveless, N. (2019). How to make art at the end of the world: a manifesto for research-creation / Natalie Loveless. Duke University Press. pp. 37