SimpleG: A Design of a beginner-friendly procedural content generator for the Unity Tool

Cristina Ampatin Pascua
School of Information Technology Mapua University, Philippines
capascua@mapua.edu.ph

Ivan Gavino
School of Information Technology Mapua University, Philippines
ipgavino@mymail.mapua.edu.ph

ABSTRACT

This study covers how uncommon Procedural Content Generation (PCG) is in today's industry and why there are only a limited number of people that use it. The game design proposal in this study contains a summary regarding a Unity Extension geared towards an easier gateway for new developers to get started with Procedural Content Generation. This extension was tested to collect feedback about their experiences with PCG and their opinions for how the extension could be improved.

CCS CONCEPTS

• **Theory of Computation**; • **Generating random combinatorial structures**;

KEYWORDS

Procedural Content Generation, Unity Extension, Unity Tool

ACM Reference Format:
Cristina Ampatin Pascua and Ivan Gavino. 2023. SimpleG: A Design of a beginner-friendly procedural content generator for the Unity Tool. In *2023 The 11th International Conference on Computer and Communications Management (ICCCM 2023), August 04–06, 2023, Nagoya, Japan.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3617733.3617739

1 INTRODUCTION

A video game is a form of entertainment which is composed of several elements such as the players, objectives, rules, resources, settings and several other elements which creates the video gaming experience when combined. In the early stages of video game development, games were limited to the memory of the devices available during that time, which resulted in games being very small and limited. However, some games managed to get past that limitation by using unique algorithms to generate levels automatically. These types of algorithms led to the beginning of Procedural Content Generators a.k.a. PCG, which has now evolved to be a very competent tool in today's game development industry. By now, computing technology has advanced far enough that games do not need to be limited as much as it was in its early stages. Despite these advancements, players today are still feeling the lack of content, thus the need for PCG arises. Although, the demand for PCG is present, a large majority of new developers have not heard of the term PCG or they are not skilled enough to use it in their work.

1.1 Objective of the Study

This study aims to design a gamified platform which provides a simpler gateway to introducing PCG to new developers by creating a Unity Extension that provides PCG functionality through an interface that does not require extensive programming expertise.

1.2 Scope and Limitations

The scope of this project is aimed towards providing a tool capable of generating multiple game levels and maps via PCG through an interface that does not require much skill in programming. It will provide significant assistance to game enthusiasts who are interested in developing casual games but have little to no knowledge of creating their own assets.

2 REVIEW OF RELATED LITERATURE

2.1 Procedural Content Generation

Procedural Content Generation is the automated process of producing content. Content could be anything a person would create, typically drawings, music or films. Procedural Content Generation in Video Games is the use of algorithmic methods to produce assets that are usually produced by artists. These assets could include sounds, terrain, stages, players, items, objectives, or even rules [1]. By the terms "procedural" and "generation'', it is implied that computer algorithms designed to create are involved. When a procedural content generator is active, a computer will create something with or without human assistance.

2.2 Early Usage of Procedural Content Generation

Computers in the early 1980s constrained the space available to store games due to their limited computing capabilities. A few notable examples of its early usage are Elite (1984) and Rogue (1980) which used PCG to generate a massive amount of levels while keeping the relative size of the game as small as possible. Since then, PCG were also used for modern titles such as Diablo, Spore, the Civilization Series, and Minecraft [2].

2.3 Properties of a Procedural Content Generator

Despite the numerous advancements in PCG, each form of PCG has its own limitations and each application has its own specific uses.

ICCCM 2023, August 04–06, 2023, Nagoya, Japan

ACM ISBN 979-8-4007-0773-5/23/08…$15.00
https://doi.org/10.1145/3617733.3617739

Although there are several types of existing PCG, they all have a few properties in common which are its [3]:

- Speed: How fast it can generate content
- Reliability: The desirability of each output
- Controllability: The amount of control the user has over the results
- Diversity: The difference between each output
- Creativity: How human-like the output looks like

2.4 Online versus Offline Games

Many games have utilized PCG, such as Civilization, Minecraft, & some others. Despite being used in several successful games, PCG is just a rising topic, and not many developers are aware of the benefits of utilizing PCG. Almost 40 years after its early examples, PCG applications are so uncommon that mentioning the topic can turn projects into worldwide media sensations, like No Man's Sky, with its virtually infinite generated universe. The lack of widespread usage is due to the lack of algorithms, and specialized professionals [4]. Although, algorithms for basic content such as textures are very common, refined, and widely used, higher complexity content such as levels, dialogue, mechanics, & etc., have a more primitive approach. In other words, ready-to-use tools are only available for basic/simple content, while more complex content algorithms are not easily accessible due to it requiring multiple modifications for it to work on different projects [5].

2.5 Reliability of Results

Procedural Content Generation is the automated process of producing content. Content could be anything a person would create, typically drawings, music or films. Procedural Content Generation in Video Games is the use of algorithmic methods to produce assets that are usually produced by artists. These assets could include sounds, terrain, stages, players, items, objectives, or even rules [6]. By the terms "procedural" and "generation", it is implied that computer algorithms designed to create are involved. When a procedural content generator is active, a computer will create something with or without human assistance.

2.6 Using Pseudo Random Numbers in Unity

A pseudo random number isn't completely random but it is a long list of generated numbers based on a seed. For example a seed of 1234 would generate the numbers 34, 55, 12, -2, 78, and so on. This way, we can replicate the output by using the same seed. A simple way to replicate randomness is to use the current time. All of this can be achieved in Unity via Random. seed and Random. range. Random. seed sets the seed while Random. range defines the minimum and maximum value that can be returned [7].

2.7 Graph Grammars

Graph grammar is a technique for generating graphs from rules, which can then be translated into levels as shown in Figure 1. Graph grammars were originally used for specifying data types and patterns, but today they are being used for visual representations and generations [8]. Basically, graph grammar works by a search and replace mechanism. It follows a set of rules assigned by the user, for example a -> b, which means a is replaced with B. Some rules can be more random by adding multiple results, i.e. a -> b|c, in which a is replaced by b or c [9].

2.8 Chunk-Based Procedural Generation

Chunk-based procedural generation is a procedural generation method that is fairly common for infinite casual games, typically infinite runners. These are considerably easier to grasp compared to other methods [10]. Most procedural generation methods have papers and articles written about them but chunk-based algorithms vary so much that it cannot simply be categorized the same way. A few examples of chunk-based procedural games are Crossy Road, Flappy Bird, Temple Run [11]. There are usually three steps in this method. First, is to generate a pool of objects, obstacles or whatever content fits in the situation. Second, a random object is selected from the pool and is inserted into the game. Lastly, once the object is no longer needed or is off the screen, it is sent back to the object pool [12].

3 METHODOLOGY

3.1 Iterative Incremental Model

The proponents have decided to use the Iterative Incremental Model as the software development lifecycle model as shown in Figure 2 since this project requires a functional initial release that improves and increases functionality through each release. The Iterative Incremental Model allows the project to develop prioritized requirements first and allows a quick product delivery. This approach helps provide important functionality to the clients at an early stage while incrementing the functionalities as future iterations are being worked on. This project was done in 7 stages, and was divided into 4 phases.

3.2 Phase 1a - Requirements

Figure 3 is a survey form that the proponents conducted a survey to gather data that would help determine how many people are interested in games, and among those people, how many are interested in developing games. The respondents consisted of individuals that showed interest in video games. To gather respondents, the proponents used the purposive method to select 20 respondents that are involved with game development. The survey consists of yes or no questions regarding their interest in making a game, their ability to create content (2D Sprites/3D Models/Level Design/Map Design), and their awareness of PCG. The survey was designed to count the number of respondents who are interested to develop a game, and the number of respondents who can already create their own games.

3.3 Phase 1b - Planning

The results have shown that the majority of respondents are not capable of creating 2D Sprites, followed by Level Design, 3D Models, and Map Design accordingly, which is why the project will proceed to create an extension for Unity for creating assets mainly for 2D Sprites, while being capable of creating the other types of assets. The proponents plan to use the Wave Function Algorithm (WFA) in an Immediate Mode Graphical User Interface (IMGUI) to introduce new developers to Procedural Content Generation

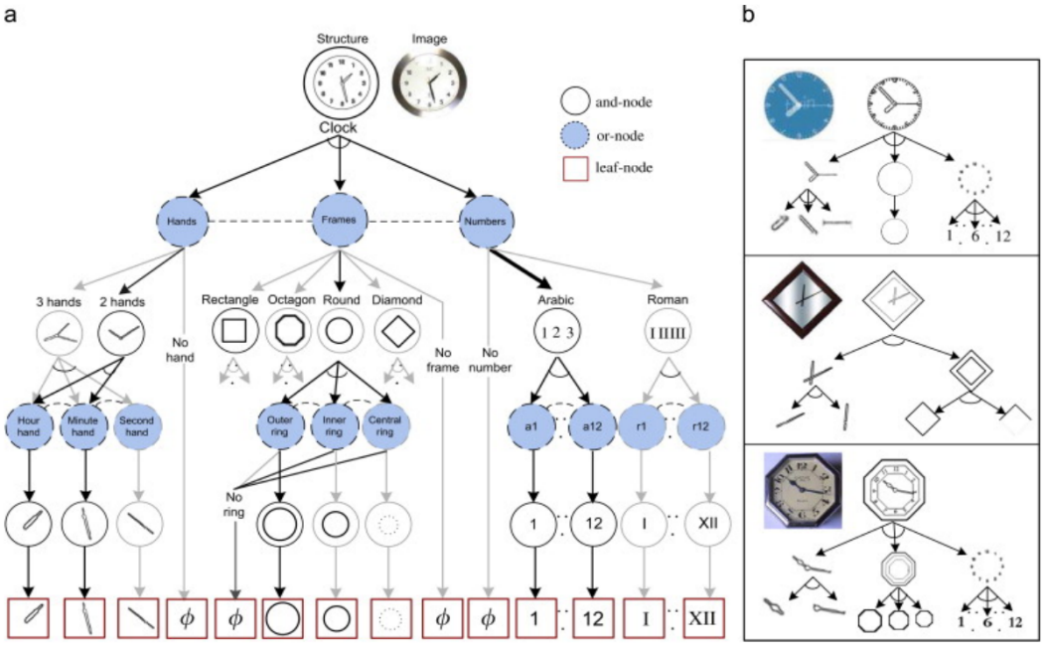


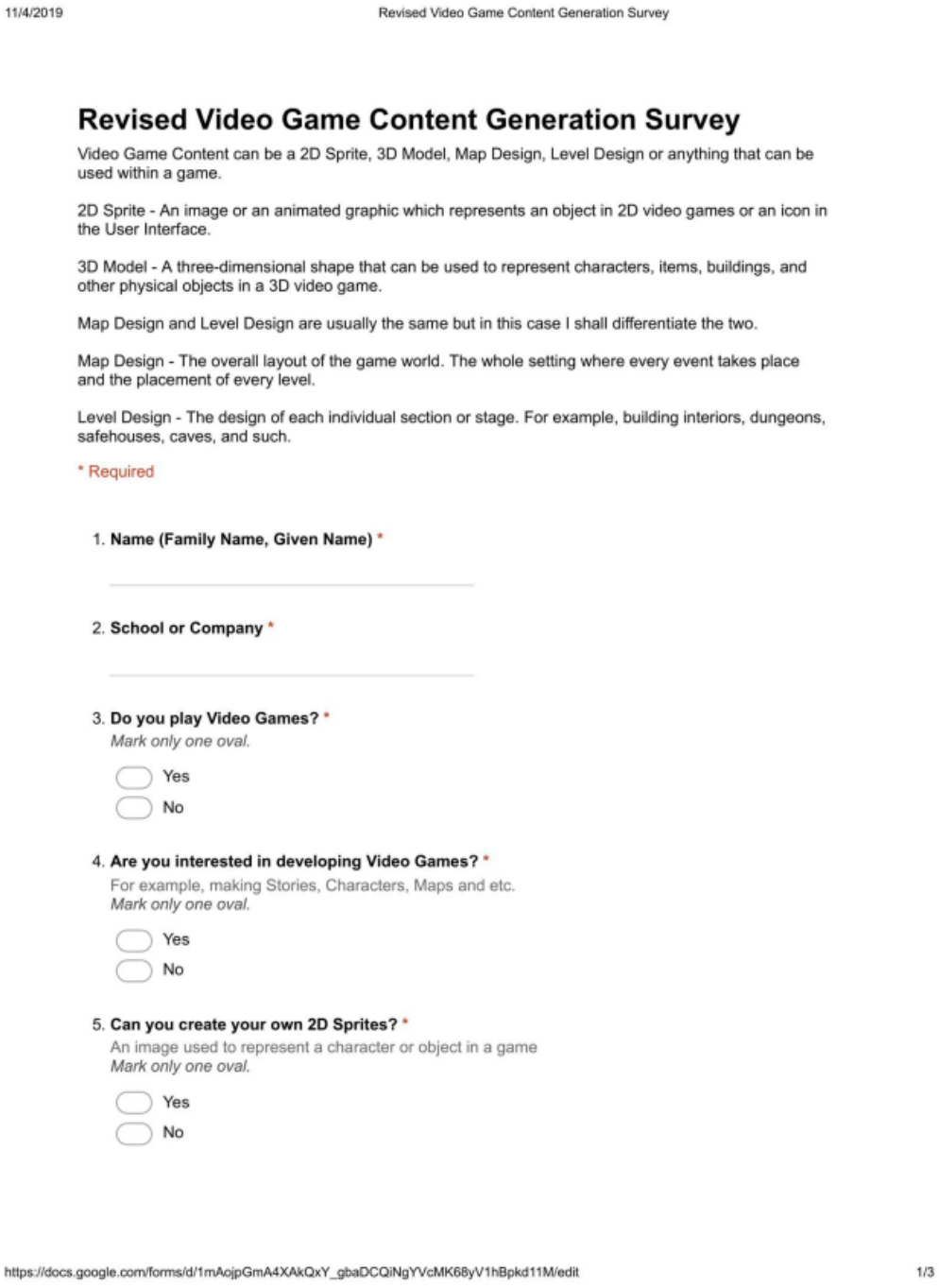
Figure 1: Graph Grammar Example

Figure 2: Game Development Life Cycle Model

in generating Level Designs and Map Designs. WFA is a method that uses straightforward set of initial values and a set of rules to generate graphics, text, audio, and nearly anything else procedurally as shown in Figure 4. However, the proponents could not find an algorithm that fits the needs for generating 2D Sprites and 3D Models while still being generally useful in most situations.

3.4 Phase 2a - Design

The design of this project focused on maximizing accessibility and convenience while still being able to provide a lot of functionality as a toolkit. The target audience for this project are developers with little to no knowledge on PCG which is why this extension only requires minimal input, and includes multiple guides and explanations on how each function works. Input fields mainly consist



11/4/2019 Revised Video Game Content Generation Survey

Revised Video Game Content Generation Survey

Video Game Content can be a 2D Sprite, 3D Model, Map Design, Level Design or anything that can be used within a game.

2D Sprite - An image or an animated graphic which represents an object in 2D video games or an icon in the User Interface.

3D Model - A three-dimensional shape that can be used to represent characters, items, buildings, and other physical objects in a 3D video game.

Map Design and Level Design are usually the same but in this case I shall differentiate the two.

Map Design - The overall layout of the game world. The whole setting where every event takes place and the placement of every level.

Level Design - The design of each individual section or stage. For example, building interiors, dungeons, safehouses, caves, and such.

* Required

1. **Name (Family Name, Given Name) ***

2. **School or Company ***

3. **Do you play Video Games? ***
Mark only one oval.
Yes
No

4. **Are you interested in developing Video Games? ***
For example, making Stories, Characters, Maps and etc.
Mark only one oval.
Yes
No

5. **Can you create your own 2D Sprites? ***
An image used to represent a character or object in a game
Mark only one oval.
Yes
No

https://docs.google.com/forms/d/1mAojpGmA4XAkQxY_gbaDCQiNgYVcMK68yV1hBpkd11M/edit 1/3

Figure 3: Revised Video Game Content Generation Survey Page 1

of required data for Wave Function Collapse algorithm to work. Additional toolbars are also included to allow users to switch between map or level generation. The released version of the project should be capable of generating outputs at a reasonably quick time, creating randomly generated 3D maps/levels that are designed for outdoor or indoor environments, generating outputs as specified by the user through the user interface. However, this project will be limited to only generating 3D maps or levels, and it will not include any 2D generations such as sprites or backgrounds. The sample generated environment is shown in Figure 5.

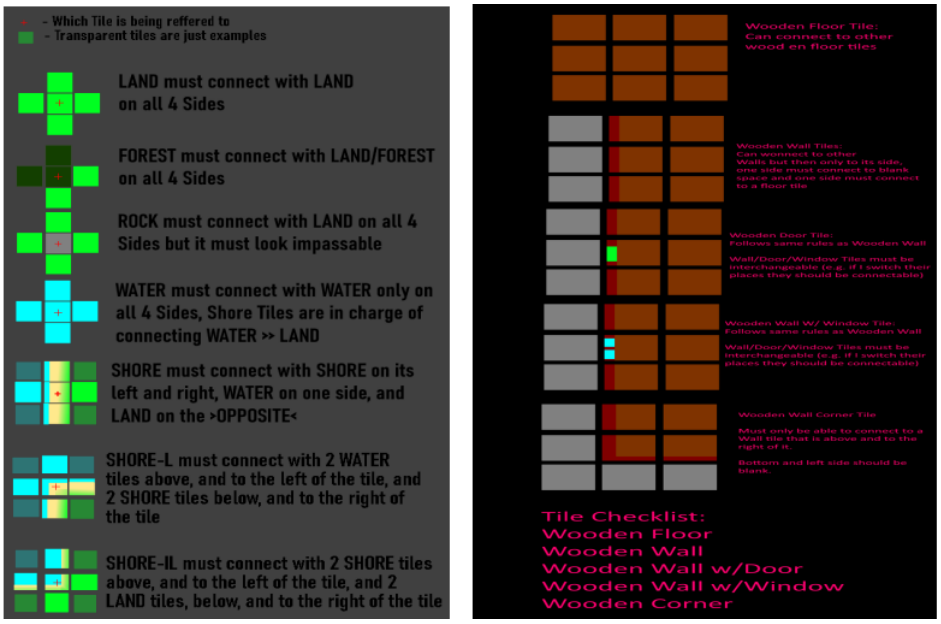


Figure 4: Planning Snippets

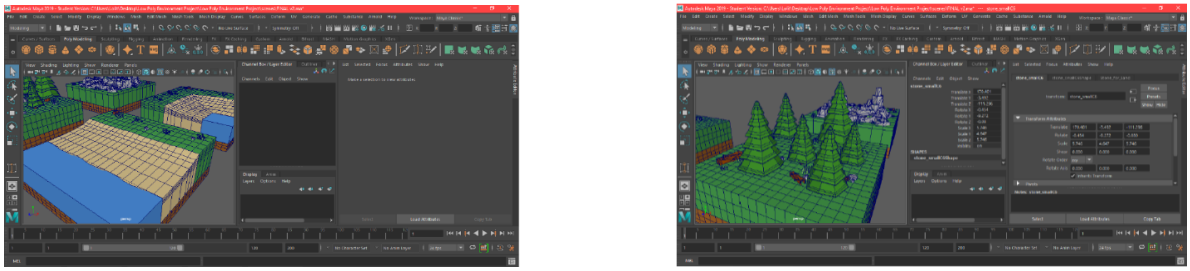


Figure 5: Asset Development

3.5 Phase 2b - Implementation

The development of this project began with creating placeholder OBJ files, which were used as the building blocks to be utilized by the algorithm. After the placeholders have been created, the proponents created the code for the Unity Extension via "Immediate Mode" GUI as shown in Figure 6. The first function that the proponents managed was the ability to create 3D Arrays via Wave Function Collapse, followed by the ability to import custom prefabs for the algorithm to use. After that has been done, the proponents have managed to create the code for generating Level Designs, 3D Models, and Map Designs also via IMGUI.

At the beginning of this phase, the project was developed on an Acer Aspire VX5-591G with a Windows 10 OS, an Intel i7-7700HQ CPU, a NVIDIA GeForce GTX 1050 GPU, and 12GB of RAM, while most of the bug fixing were performed on a desktop with a Windows 10 OS, an AMD Ryzen 5 3600 CPU, a NVIDIA GeForce GTX 1650 SUPER GPU, and16GB of RAM.

3.6 Phase 3a - Deployment

Figure 7, shows that upon completion of the current project release, the project will be uploaded to Itch.io and it will be available to all users for free. The release will be the first iteration of this project.

3.7 Phase 3b - Testing

Upon release, 2 selected test subjects will be asked to make a few small games as they normally would, and recreate the same game using the extension. Each recreation will be graded by comparing it to its original game. Recreations will be graded by their quality, variety, and overall visual appeal. This test will be recorded for references of known bugs and for listing down possible improvements. Feedback on the websites the project will be uploaded and will also be monitored. Data gathered throughout this stage will be added to a list of bugs and changes which would then be evaluated before being added to the next iteration. In the case of a new Algorithm being released to the public, it will be evaluated in the next phase to decide on whether it should be added or if it should replace the current framework.

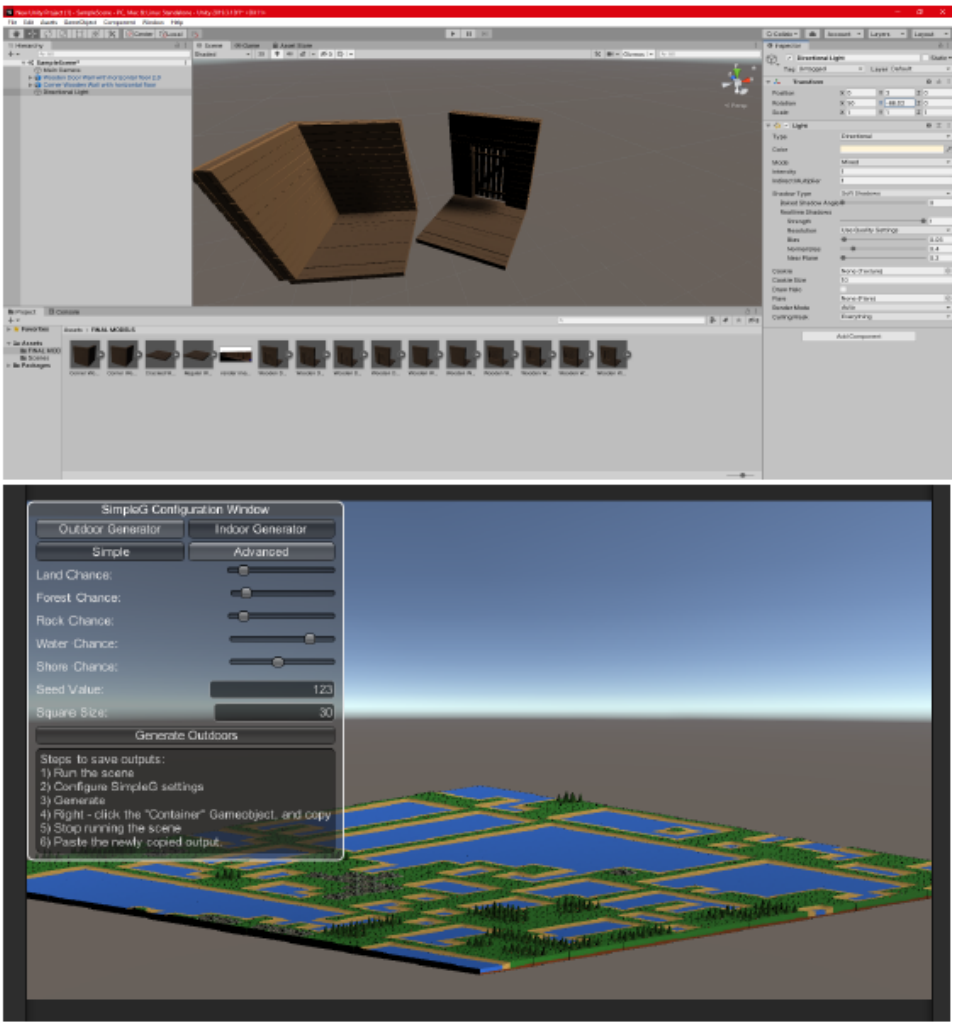


Figure 6: Unity Implementation

3.8 Phase 4 - Evaluation

As shown in Figure 8, the program will be evaluated by having a selected group of respondents tried the application and answer a Google Form shortly afterwards. The questions included in the form will mainly focus on the applications Assets, User Interface, Understandability, and User Experience. For the project's final evaluation, 24 respondents from Metro Manila who are interested in game development were selected to participate. 20 of these respondents were selected from the pre-development survey, since they fit the profile, but the remaining 4 were random students from the same batch of the proponents that also fit the profile. These 24 respondents are within the range of 20-25-year-old. To begin with the evaluation each respondent was given the google form and the link to the released SimpleG build on itch.io and they were given a week to respond. After a week has passed, the form will be set to no longer accept answers, and the results will then be analyzed by transferring the data to Google Sheets. Once the data has been reviewed, the results will give a clearer indication of which features require further improvement. The statistical treatment that will be used to analyze the results will be descriptive statistics, in which the data will be summarized and presented as a graph.

4 RESULTS & DISCUSSION

4.1 Pre-development Survey

Based on the Iterative Incremental Model (IIM), knowing the requirements of a project is a significant step for its development. With that in mind, the developers conducted a pre-development survey as show in Table 1 for the sake of discovering not only the familiarity of other students but also the necessary features for the development of a quality application.

4.2 User Experience

SimpleG aims to be a convenient tool for users that are unfamiliar with the concepts of PCG, which is why it is essential for the user's experience to be very streamlined. Table 2 shows that by testing

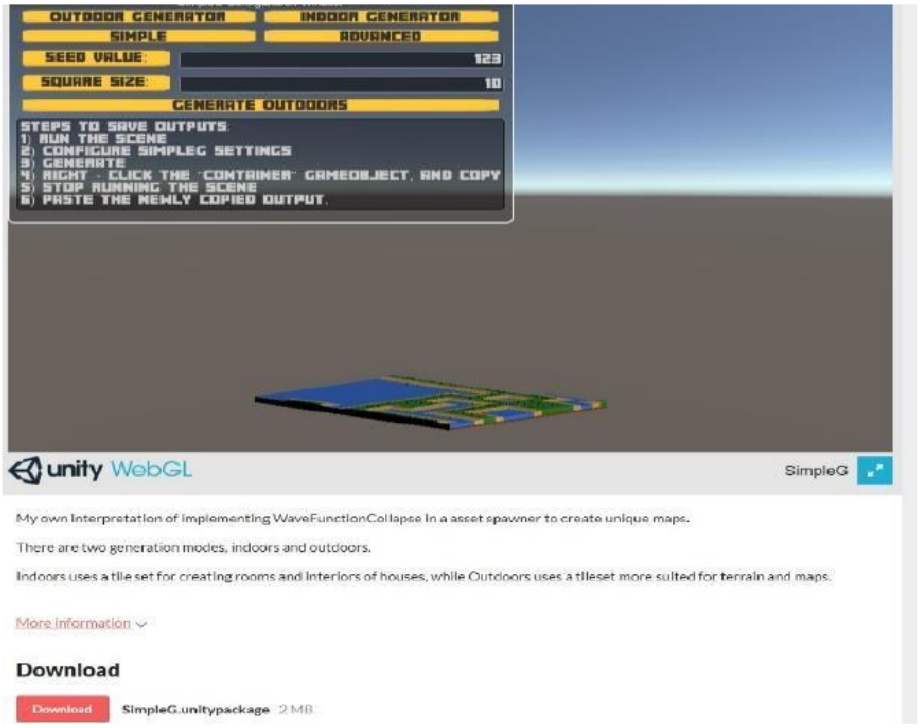


Figure 7: Project uploaded on itch.io

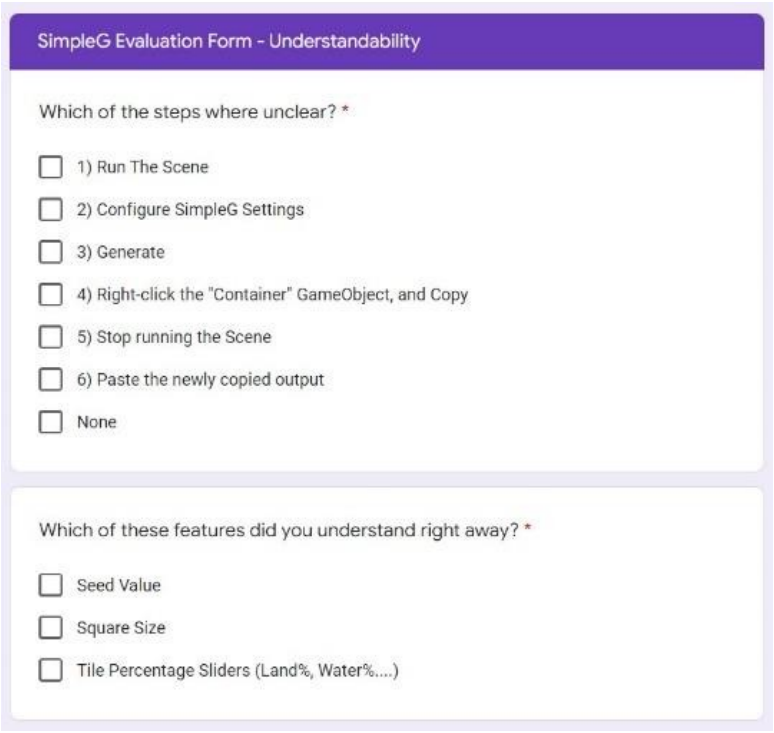


Figure 8: SimpleG Evaluation Form

Table 1: Survey Results Summary

Question	Yes	No
Do you play Video Games?	100.00%	0.00%
Are you interested in developing Video Games?	90.00%	10.00%
Can you create your own 2D Sprites?	35.00%	65.00%
Would you like to try using a program to assist you with making your 2D Sprites?	100.00%	0.00%
Can you create your own 3D Models?	35.00%	65.00%
Would you like to try using a program to assist you with making your 3D Models?	90.00%	10.00%
Is it easy for you to design Levels?	25.00%	75.00%
Would you like to try using a program to assist you with designing Levels?	100.00%	0.00%
Is it easy for you to design Maps?	20.00%	80.00%
Would you like to try using a program to assist you with designing Maps?	95.00%	5.00%
Have you heard about 'Procedural Content Generation'?	40.00%	60.00%
Have you used any of the PCG Methods in any of your projects or practices before? If yes, please input the name or description.	0.00%	100.00%

Table 2: User Acceptance Testing (UAT) Results

#	Question	Agree	Neutral	Disagree	Desired Results
1	The output is generated fast enough	91.67%	4.17%	4.17%	Above 75% Agree
2	The tiles of the output connect well with each other	95.83%	0.00%	4.17%	Above 75% Agree
3	Changing the Land % slider affects the amount of Land Tiles	91.67%	4.17%	4.17%	Above 75% Agree
4	Changing the Forest % slider affects the amount of Forest Tiles	95.83%	0.00%	4.17%	Above 75% Agree
5	Changing the Rock % slider affects the amount of Rock Tiles	95.83%	0.00%	4.17%	Above 75% Agree
6	Changing the Water % slider affects the amount of Water Tiles	91.67%	4.17%	4.17%	Above 75% Agree
7	Changing the Shore % slider affects the amount of Water Tiles	91.67%	4.17%	4.17%	Above 75% Agree
8	Changing the Seed Value creates a completely different output	91.67%	8.33%	0.00%	Above 75% Agree
9	Changing the Square Size creates a completely different output	95.83%	4.17%	0.00%	Above 75% Agree
10	Switching between Indoor and Outdoor generator is done smoothly	91.67%	8.33%	0.00%	Above 75% Agree
11	Switching between Simple and Advanced mode is done smoothly	95.83%	4.17%	0.00%	Above 75% Agree
12	The generated output is easy to access within the Unity Editor	83.33%	16.67%	0.00%	Above 75% Agree
13	Generating a new output leaves no traces of the previous output	95.83%	0.00%	4.17%	Above 75% Agree

different features of the application, the developers can have a clearer indication if the application has met the expectations based on the design of the project.

5 CONCLUSION & RECOMMENDATION

Based on all the information gathered in this research paper, the developers can conclude that the developed application is of sufficient quality. However, it is also important to note that there is a lot of room for improvement which is shown through the evaluation results.

During the project's development, it can be observed that there were certain limitations that came in the way, with that in mind the developers has recommendations for future researchers.

First of all, a massive limitation would be the fact that integrating PCG into Unity is too CPU-intensive and it would not be viable for all users to have high end CPUs. The recommendation of the developers for this is to create third party program for a more optimal experience for all users as this significantly reduces the CPU load.

In addition to that, SimpleG originally intended to include the functionality of creating 2D Sprites and 3D Character Models, along with Outdoor Maps and Indoor Level Design, but along the development phase, the proponents could only support the generation of Outdoor Maps and Indoor Level design. The reason for not including the 2D Sprites and 3D Character Model functionality is that there currently is no algorithm available that is capable of adapting to multiple art styles and to different needs. With that in mind, the developers believe that the lack of algorithms is a limitation to this project, and recommends that if any algorithms were to be made available in the future then developers should utilize it.

REFERENCES

[1] Barriga, N. A. (2018). A Short Introduction to Procedural Content Generation Algorithms for Video games.
[2] Shaker, N., Togelius, J., & Nelson, M. J. (2016). Procedural Content Generation in Games: A Textbook and an Overview of Current Research. Springer.
[3] Viitanen, H. (2016). Procedural City Generation Tool with Unity Game Engine.
[4] Vrtacic, E. (2014). The Grand Narratives Of Video Games: Sid Meier'S Civilization.
[5] Korn, O., & Lee, N. (2017). Game Dynamics: Best Practices in Procedural and Dynamic Game Content Generation.
[6] Watkins, R. (2016). Procedural Content Generation for Unity Game Development.
[7] Wawrzyn. (2019, November 12). Procedural Shape Generation, Retrieved January 6, 2020, from https://assetstore.unity.com/packages/tools/modeling/procedur al-shape-generation-123369.
[8] Beyer, T. (2017). Story Guided Procedural Generation of Complex Connected Worlds and Levels for Role Play Games.
[9] Bond, L. (2017). Procedural generation: an algorithmic analysis of video game design and level creation.
[10] Gaisbauer, W., Raffe, W. L., Garcia, J. A., & Hlavacs, H. (2019, October). Procedural Generation of Video Game Cities for Specific Video Game Genres Using WaveFunctionCollapse (WFC). In Extended Abstracts of the Annual Symposium on Computer-Human Interaction in Play Companion Extended Abstracts (pp. 397-404). ACM.
[11] Naveen, A., & Vare, S. (2018). Playing FlappyBird with Deep Reinforcement Learning.
[12] Compton, K., & Mateas, M. (2006). Procedural Level Design for Platform Games.