Multi-Agent Based Settlement Generation In Minecraft

Albin Esko
Johan Fritiofsson

Computer Science

Bachelor Thesis

15 ECTS

VT/2021

Supervisors: José Font, David Täljsten

Abstract

This thesis explores the uses of a multi-agent system(MAS) for procedural content generation (PCG) in the Generative Design in Minecraft (GDMC) competition. The generator constructed is capable of surveying the terrain and determining where to start building a road network. Extendor and connector agents build the road network used for the settlement. A plotting agent surveys the area around the created roads for plots appropriate for building houses. A house building agent then generates basic buildings on these plots. Finally a furniture agent places furniture in these buildings. The result of the thesis shows that the generator is capable of generating an interesting road network that is appropriate to its terrain. The buildings have potential but are lacking in form of adaptability to the current biome and buildings are overall too similar to be interesting, causing it to get low scores in the user study and competition. The generator was entered to the GDMC-competition in 2021 where it placed 17th of 20th place.

Keywords: Multi-Agent system, PCG, Minecraft, GDMC

Multi-Agent based settlement generation in Minecraft

1. Introduction

In modern game development the creation of in-game content is currently one of the larger bottlenecks. This is due to the increasing fidelity and amount of assets the developers must create to continue improving the expressiveness in demanding games. Additionally, the point has been reached where the classic solution of adding more artists fails to scale relative to the demands [9], [6]. However, this problem is alleviated with the use of a subfield of Artificial Intelligence (AI) where developers are able to create large amounts of content for their games such as; levels, weapons, companions, items, textures or even music with relatively little effort. The subfield is called Procedural Content Generation (PCG) and it is the research field for automatic creation of content using AI with limited or indirect user input [1]. PCG has been used since the 1980s with the first game credited to use PCG is Rogue [7]. Rogue used PCG for procedurally creating level layouts to have an almost infinite number of levels in the game. Since then, PCG has further developed and is widely used to generate levels among other use cases.

Although powerful creative tools, modern PCG still operate the same way as it did in Rogue, they create levels from scratch with very few parameters to dictate the level's development [3]. Examples of inputs for these types of PCG is a number seed which creates the foundation of all random elements in the algorithm. A Minecraft [13] world is generated with such a seed. These systems which create levels from scratch are limited to what they create the first time and its development is only governed by the initial parameters. While this type of generation is useful to create completely new artefacts, there are artefacts which are heavily dependent on context. For instance settlements require a terrain to be placed on and the properties of the terrain affect how the settlement develops. To generate a believable

settlement, the terrain's characteristics need to be accounted for. This is one kind of outside influence which affects the settlement's generation and something that needs to be adapted to. The method described is an already established sub-field of PCG called adaptive PCG which is not as well explored as the normal kind of PCG and warrants further investigation [3].

Minecraft [13] is a survival sandbox voxel game where the player gathers resources, crafts tools, explores and builds constructions. The game world is procedurally generated with different landscapes, features, and structures. One kind of structure is a settlement where NPCs known as villagers live. These structures are generated by selecting a location according to some prerequisite conditions and then generating the village. Consequently, the villages are regularly faulty such as houses being inaccessible for the in-game characters or there is a giant ravine through the village without a bridge to cross it.

Salge, Canaan, Green and Togelius [3] introduce a competition called Generative Design in Minecraft (GDMC). This competition provides an established environment to use an adaptive and holistic approach when creating PCG systems for generating settlements in Minecraft. As of writing this paper the competition is in its fourth year of running and with approximately 20 entries for the previous three years and an additional 20 entries this year (2021). These entries [4] have utilized different approaches to generate settlements. Some examples are structured algorithms, graph grammars [5], and agents. One of the previous entries commonly used as a comparison is Filip Stwarski's 2019 entry. That generator randomly samples locations for buildings then tries to connect them all with roads using an A-star pathfinding algorithm.

However, not many entries have utilized multi-agent PCG which is a technique where multiple agents perform simple changes on the environment and create complex structures through emergent behavior. The fundamental idea is creating a PCG system which is capable of generating believable settlements in any Minecraft world.

The purpose of this thesis is to implement and compete in the GDMC-competition with a multi-agent PCG system. The generator created will not take part in the chronicle challenge in the GDMC-competition. This work postulates that multi-agent systems can be used to adaptively create a city from a bottom-up process and consequently create a cohesive settlement. Previous work in city generation using multi-agent systems (MAS) have explored the aspects of road generation and zoning for residential and commercial areas [6]. The work of this thesis applies the road generation previously mentioned in Minecraft to expand the expressivity of the method.

Research Question:

How can a MAS in the style of Lechtner et. al. [6] be used to implement a settlement generator in Minecraft?

Sub-Research Question:

How does an implementation of this compare to other GDMC entries when using the complete judging process?

This research is done following the Design Science Research Methodology, which entails an artifact is created and evaluated based on a specific problem. Evaluation of the artifact is based on Salge et. al [3] criterias for the competition and is done in two situations. The first is a user study which is performed as a part of this thesis. The second is participation in the GDMC competition for 2021. The research results in an experience report regarding the feasibility of using a multi-agent PCG system [11] which is capable of adapting to a multidimensional input. After the introduction a more in-depth look into the research framework and methodology will follow. Thereafter the results and a conclusion shows and discusses the accomplished research. The thesis is then finalized with suggestions for any future research.

2. Research framework

This work is within the PCG field of research where it explores a way to generate settlements on a preexisting terrain. To accomplish this, the field of PCG was researched to identify a niche which would benefit from further research.

2.1. PCG

PCG was brought to the limelight in the early 1980s where games such as Rogue [7] and Elite [8] successfully utilized PCG to create groundbreaking features. Rogue is a dungeon crawling adventure game with procedurally generated levels, making every play session unique in terms of level layout, monster encounters and treasures. Another way PCG was used to create the “endless” game experience is exemplified in the game Elite [8]. Due to the limited capabilities of the hardware at the time, information about planets, galaxies, space stations etc. had to be procedurally generated [2].

During the 1990s PCG was also used to automate the designing of assets to increase the amount of content in order to improve replayability, notable examples of this is Diablo [10] with its random dungeon layout and loot [2].

Other uses for PCG are to generate assets in game. This is done in development tools such as SpeedTree [32]. SpeedTree provides tools to rapidly create unique trees, from individual plants up to a forest in size. This is accomplished with PCG where the developers can choose to let the program generate trees freely or they can modify them after generation. This kind of use case is utilized in other programs or tools to generate, for example cities or characters. It is also the aspect of PCG this thesis focuses on.

2.2. City Modeling

In 2001, Parish et al. [12] published a paper where cities were procedurally generated. Their paper created a foundation to expand the PCG field to the realm of city generation. Previous work primarily focused on creating natural artifacts such as trees, plants, and landscapes. Following Parish et al.'s publication, additional techniques were developed and used to generate cities. Lechtner et al. [6] created a multi-agent system which when given a heightmap, water level, and style can create a natural looking city, see Figure 1. Jing, et al. [14] utilized different templates created from, for example, Voronoi noise and grids, and a combination of these to be able to model a traffic network which is an essential element for cities.

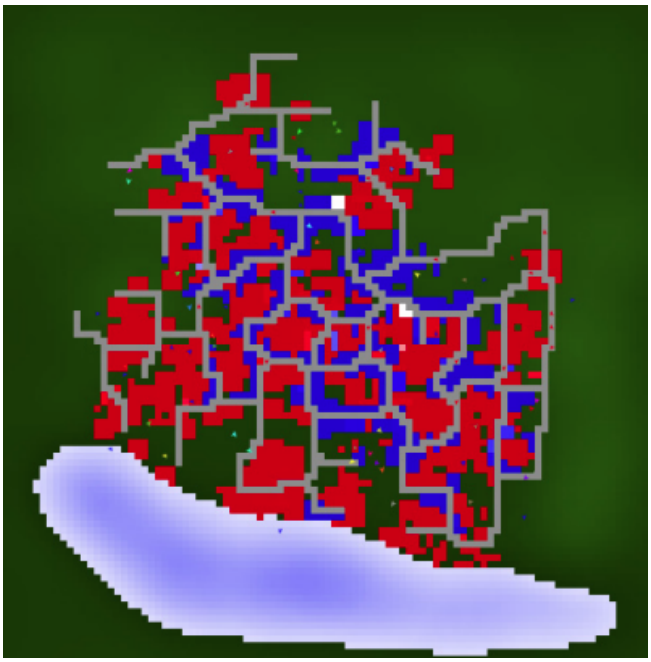


Figure 1
Example of procedural city modeling. Source: Adapted from [6]. This shows the output of the MAS generator with different zones. Red for residential, blue for commercial, then the brighter colors indicate higher density.

Procedurally generating cities is an attractive proposition to many stakeholders in the games industry. Since the customers expect growth both in scale and detail in order to utilize the growing processing power available to them. However, using the normal production cycle

of 3D modeling by hand would increase the costs beyond what is sustainable or profitable [9], [6]. One of PCG's purposes is to reduce the substantial work hours required to create a good city. However, games are set in more kinds of environments than large cities. For instance, villages and hamlets are common features in games and are usually more numerous than the large cities if compared in the context of open-world games such as The Witcher 3 [24]. Although sometimes more numerous, village and hamlet generation have not been as thoroughly researched compared to the generation of larger metropolises of modern day cities [15].

2.3. Multi-Agent PCG

Multi-agent systems (MAS) is a problem solving technique used in many fields of research such as computer science and civil engineering [21]. It is attractive due to its properties such as scalability, and flexibility. The scalability comes from the agents' local vision which is constant and prevents exponential time complexity, and the flexibility stems from the infinite possible behaviors attributed to any number of agent types. In the field of generation, MAS has been used to generate continents [11] and city layouts [6], see Figure 2 for an example continent. These papers use deterministic agents to generate distinct and believable outputs due to the emergent properties of multiple agents working simultaneously. The work from Lechtner et al. [6] is particularly interesting to this paper since it generates cities through a MAS. This thesis expands on Lechtner's work by implementing a MAS based generator for 3D space.

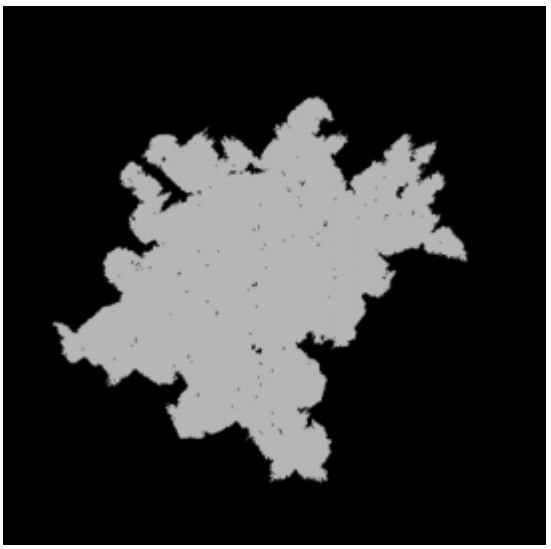


Figure 2
An Agent created coastline. Source adapted from [11]. Shows a landmass (white) raised from the water (black) by using agents to raise the terrain at several different positions.

2.4. GDMC Competition

Competitions are commonplace in the AI field of research. There are competitions such as DARPA Challenges [16] and the Robocup [17]. Where the former focuses on self-driving cars and bipedal robots and the latter on robots completing various kinds of tasks, like playing soccer, sprinting, or racing. Expanding further, there are also various competitions for AI agents to play different kinds of games, such as GVGAI [18]. The GDMC-competition is another variant, created by Salge et al. [3]. The GDMC-competition differs from the earlier mentioned competitions, by measuring expressiveness instead of raw performance, whether it be the fastest time or highest score.

A game like Minecraft benefits from measuring expressiveness, where much of the enjoyment of the game comes from exploration and creation. It does not lend itself well to the other competitions' measurements of speed and high scores. These measurements are dependent on the entries being able to be quantitatively evaluated with a built in scoring system or timer. Hence why the GDMC-competition utilizes a human jury and qualitative evaluation. The evaluation accounts for different aspects; adaptability, functionality,

narrative, and aesthetic. Each aspect is scored between 0 and 10 then each submission gets an overall score from the average score of the aspects.

The GDMC-competition tasks the contestants with creating a settlement generator. The generator should be capable of generating all required elements of a Minecraft settlement, this includes roads and buildings but also farms for growing food, lights to keep inhabitants safe to name a few. The aim and challenge in the competition is to create a system that is capable of adapting to the many different terrains in Minecraft and to create a settlement that is larger than the sum of its parts. With this competition, Salge et al. provides an opportunity to further the research in adaptivity to content, and further research in holistic PCG.

An optional Chronicle challenge is also a part of the GDMC-competition. The challenge premise is to use the generator to create a narrative story and put it in the game through a written book. This story could include how the settlement came to be, notable citizens in the settlement, events that took place or any other thing that is notable or has involvement to the settlement development or its inhabitants. The challenge relates to the criteria mentioned earlier such as adaptability and narrative.

2.4.1. MCEdit

One of the ways to submit an entry to the competition is to use the MCEdit framework. This is the method used for the thesis. MCEdit [20] is a powerful world editor that can manipulate all aspects of a Minecraft world. This is done through the use of various built-in tools. The tool used for the competition is the filter tool. With it, users can apply various filters on the selected terrain to modify it according to the filter's specification. A filter is a collection of files with one python file that has a "perform" function which McEdit calls when that filter is executed. The perform function acts as the program's start function and from there can import and utilize any number of files such as python scripts and other

data storages enabling the filter to be as complex as required. The preform function requires three input parameters, the current Minecraft level, the selected volume, and user specified settings.

2.4.2. Earlier Contestants

Previous contestants have used several different techniques and analyzing these provides insights in different ways to create a settlement generator. This part describes some common or notable techniques used by the top three placing submissions [22] from each year of the GDMC-competitions.

One of the most common techniques is to scan the selected area to create maps which are used in later stages. The most common map is the heightmap which allows modelling the game world in 2D thus simplifying the implementation. This consequently limits the generation from using caves or overhangs since it assumes the spaces under the noted value on the heightmap are solid blocks which is not necessarily the case in Minecraft.

Another common technique relates to building placement. This one samples possible positions for different buildings and evaluates those positions with regards to some parameters, usually flatness of the ground and the space available. This way creates a settlement with similar population density everywhere which is plausible in smaller villages and hamlets but not in towns and cities where a town center is distinct. An example in Figure 3.

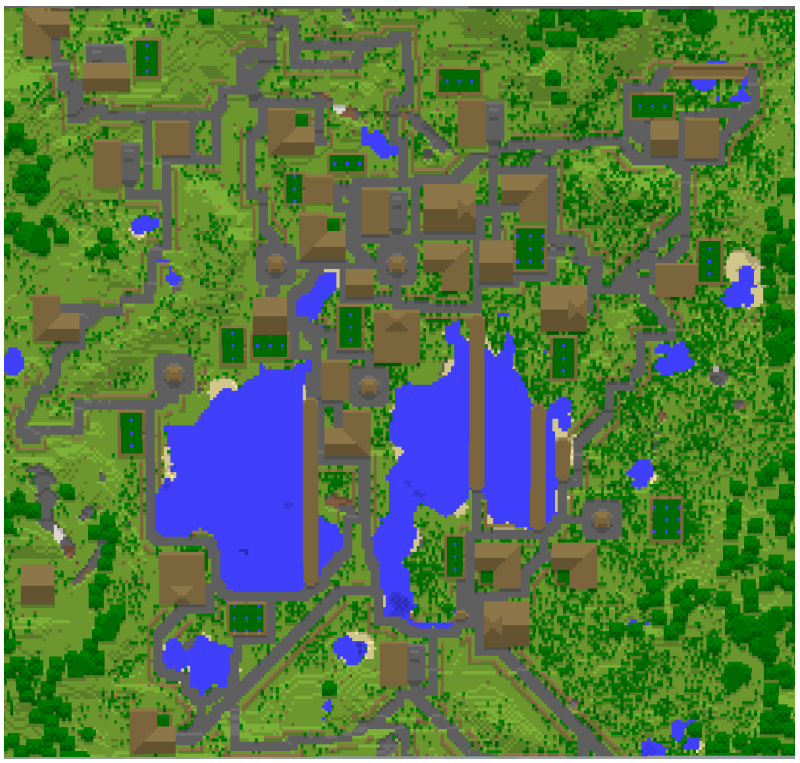


Figure 3
Layout over settlement created with Filip Stwarski's 2019 generator. Used in the user survey as a competitor.

Another notable way to place houses is with the use of Voronoi partitions. This has been used in previous city generation techniques to create a good representation of a city [14]. In the contest an entry used Voronoi to create different districts and house yards. This created a town which was structured with distinct sections but not as rigid as a grid, see Figure 4.

Figure 4
Voronoi partitions used to create a settlement in MCEdit.

Road generation is commonly solved with the A-star pathfinding algorithm to create roads between buildings. The resulting road network is functional as most buildings are connected with each other. However, it also creates unnecessary parallel roads which could be merged into one road. Examples are seen in Figure 5 and Figure 3

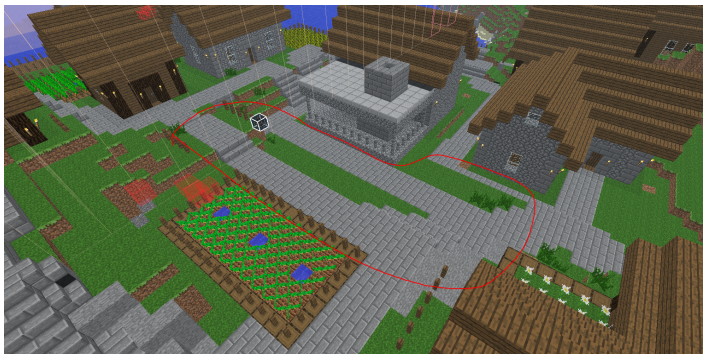


Figure 5
Merge-able parallel roads created by A-star algorithm.

Building generation is commonly accomplished with the use of templates where the creator has specified the specific layout of a building and connected that layout to a type. These templates are then created in the designated place. A way to create some variety on the templates is to change the materials used in its construction dynamically. This can be done

randomly or by analyzing the nearby terrain to select the material. Additionally selection can depend on the narrative for the particular building where richer houses use more exotic materials.

Only one of the previous submissions was found to use a multi-agent system generator. That system functions differently from the one described by Lechtner et al. [6] and consequently differs from the one created for this thesis. The multi-agent system used in the competition created by Brightmoore works by each agent simulating a named habitant of the settlement [30]. The agents roam the selected area and are capable of building structures based on the resources around it, water allows for farms, wood allows for cottages and lava for blacksmiths and more. The agents also have distinctive material and pattern style reflected on the structures built, exemplified in Figure 6. The agents are goal based, they survey their vicinity and generate a goal such as building a house, in contrast to the agent system proposed in this thesis. These agents are task based, meaning that the agent only serves a single purpose, such as building something specific like a road, a building or a piece of furniture. Another significant difference is also that Brightmoores agents represent inhabitants of the settlement while the agents of this thesis are only creating the settlement and are not tied to its narrative.

Figure 6
Examples of distinctive patterns of buildings from Brightmoore's MAS generator.

2.5. User Study Competitors

The user study conducted utilizes three generators excluding the one created. This part provides an overview of how they function. This information is gathered from the creator's descriptions of the generator and, if warranted, a code analysis.

The first generator is named Filip Skwarski 2019 (Skwarski) [32]. This generator attempts to generate the most suitable placement for structures by sampling plenty of random locations and then selecting the best position and building template based on several criteria. Then the new building is connected to any previous buildings with A-star and an evaluation if it is a suitable road. It also modifies the terrain to increase available flat area and reduce obstacles; this modified terrain is used near placed buildings. The building materials are also selected to match the surroundings.

The second is named Troy, Ryan, & Trent (TRT) [33]. This generator reserves plots for its buildings by analysing the height map and finding suitable locations by searching for areas with small differences in height. The plot is also tested to prevent unrealistic spawning of buildings. An A-star algorithm creates roads between the buildings. The buildings' shapes are handcrafted and are able to change their building materials depending on the environment, they also serve different functionality such as blacksmiths, farms and shops. The whole village is encircled by a wall.

The third and final is named David Mason (Mason) [35]. Mason uses voronoi to partition its inner and outer sections into districts. Then within the districts designated as residential another voronoi partition occurs to divide the district into plots and the houses are placed within these plots. Building material is selected to match the surroundings. The other districts in the outer section can be designated as farms and one district in the inner is designated as the quarry.

The method chapter explains why and how these generators were selected.

3. Method

This chapter will explain the methodology used in this thesis. The Design Science Research Methodology (DSRM) [23] makes up the majority of the chapter, and the GDMC-competition uses the rest as the artefact is a participant of it.

3.1. Design Science Research Methodology

The DSRM process is established in the information system (IS) field when producing and evaluating artifacts for research purposes. The process consists of an iterative six step process, detailed below;

1. Problem Identification - Defining the specific research problem and justifying its value. To help with the problem definition it is possible to atomize the problem so that its complexity can be captured. Justifying the value of the solution motivates the researcher’s and audience's pursuit of the solution and its results.
2. Objectives of a solution - Derived from problem identification objectives for the solutions should be defined. The objectives can be either quantitative or qualitative.
3. Design and development - Consist of determining the desired functionality and architecture as well as creating the actual artifact.
4. Demonstration - Display the efficacy of the artifact’s solution to the problem. This can be done through experimentation, simulation, case study, or other appropriate activity.
5. Evaluation - Observing and measuring the results from the applied solution. Also involves comparing the objectives of the solutions to the results observed from step 4. Once satisfactory results are met proceed to the next step, otherwise iterate back to step 3.
6. Communication - Communicate the problem and its importance, the artifact, its utility and novelty, rigor of its design and effectiveness to relevant audiences.

3.1.1. Problem Identification

Building on what Salge et. al [3] identifies as one of the problems with PCG. Usual implementation of PCG algorithms is to generate content from a vacuum, meaning that they take no regard to outside factors except from seed or parameters set for the generator before it runs. PCG algorithms that have to adapt to its surroundings are more challenging to implement. The GDMC-competition provides a suitable testbed in Minecraft and with its aim to push advancement in the PCG field for adaptability it solves the problem stated above.

This research focus is on the use of a multi-agent system to handle the challenges of generating a believable settlement on an existing terrain.

3.1.2. Objectives Of A Solution

Objectives for the solutions can be divided up into parts. Firstly the solution needs to be able to orient itself in the provided world i.e the Minecraft world. It must be able to analyze the provided world and make decisions where the agents in the MAS will originate from. Once started the agents perform according to their set task, provided that their requirements are met. Agents will be expanding the settlement through building roads and buildings. Agents must be able to work in unison and not overrun the other agents or hinder them from performing their task. The solution's expected runtime is in the range of a few minutes.

3.1.3. Design And Development

One of the ways to take part in the competition is to use the MCedit framework and use Python 2.7 for the code. A design document keeps all information about desired functionality, helper functionality and data structures needed for the solution to work. Features listed in the document are broken down into manageable pieces and implemented into the artifact.

3.1.4. Demonstration

The artefact is demonstrated in two different contexts. The first is a development context which is performed by predicting what the output of a specific part would be and then executing the relevant part. The output is then evaluated internally. The other context is a complete run. This run uses the completed artifact and executes it over an area which is the specified size. This is then evaluated by external individuals.

3.1.5. Evaluation

Evaluation of the developed solution is done internally, with focus on iterative steps to build the generator. Once the generator is deemed to produce adequate results a user study is conducted. The user study is structured similarly to the GDMC-competition. The participants were given a number of different levels created by four different generators and tasked to judge these generators based on four different aspects originating from the GDMC-competition[3].

- Adaptability, how well is the generator adapting to the environment that it is in? Does it take advantage of the terrain?
- Functionality, does the settlement provide protection from danger? Is there an easy way to get food?
- Narrative, does the settlement evoke an interesting story? Is it clear what kind of settlement is generated?
- Aesthetics, does the settlement look good? Is there an appropriate level of variation of the existing structures?

To guide the participants further in regards to judging each aspect additional descriptions of each aspect is provided, see Appendix A. Participants rate each aspect on a scale of 0-10 points, where 0 points generator takes no consideration to the aspect and 10 points means that the generator performs on a superhuman level. 5 points indicates that the generator performs on the same level as basic human constructions. The tests are conducted on the participants own PC.

Before any participant has any contact with the generated levels they are anonymised so that no biases should affect the judging of the generators by relation to the authors of the generator. The order in which the participant explores them is also randomized to circumvent biases based on what generator the participant explored first.

The generators that were chosen to be included in the user study were selected by first collecting the top three performing generators of earlier years and then randomly selecting three of these to be a part of this study.

The selected generators are :

- Filip Skwarski 2019
- Troy, Ryan, & Trent
- David Mason

Then several maps were generated in MCedit where they were evaluated on a set of criteria. The criterias were bodies of water, height differences, forests and variety. After generating multiple maps, two were selected to be the maps all the generators were to generate a settlement on. Afterward all generators were executed on the same area in the selected maps. Outputting a set of eight maps, each of the four generators executed once on each of the different terrains.

Participants to the study were found through the authors' classmates, friends and through the GDMC-competitions official Discord server.

3.1.6. Communication

Communication of the problem and its importance is done through this experience report. The solution can benefit contestants in the GDMC-competition and help understanding of the use of MAS in a PCG environment.

3.2. GDMC-Competition

The artefact of this thesis participates in the GDMC-competition of 2021. The optional chronicle challenge is not part of the scope for this thesis and thus the generator will not be entered for that part. The scoring is done by external participants unknown to the

authors of this thesis, and thus the results from the competition itself is only presented in the results chapter.

4. Artefact

The resulting artifact from this paper is a filter for the MCedit framework. The filter's inputs are the Minecraft level, a selected volume within that level and optional user options. For this generator the user options are not used. The filter is executed within the selected box and only affects the level within its bounds. The code is written in Python 2.7 and can be found on the authors' GitHub [36].

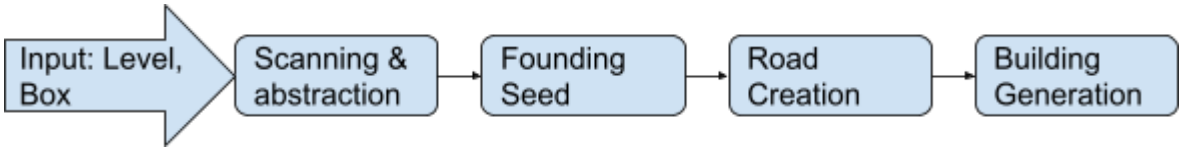


Figure 7
Generator sequence for artefact.

This system is based on previous multi-agent city generators [6] and is seeking to expand on it through implementing a similar system capable of working in a 3D space and by creating actual buildings in the settlement a user can view. An overview of the process can be seen in Figure 7.

4.1. Prerequisites

The artefact works in the context of a selection of a Minecraft level. The Minecraft level contains the terrain which consists of multiple different blocks in a 3D grid. These blocks are grouped in 16x256x16 packets called chunks. The selection has a size of 16x16 chunks which translates to 256x256x256 blocks. The selection is the boundary of the generator's effective area and is the input along with the whole level.

4.2. Scanning & Abstraction

The selected box within the Minecraft level is scanned and a heightmap and liquid-map is created. The scan iterates over every x and z coordinate and in every iteration it traverses from the maximum y value to the first solid ground block. The definition for a ground block is a naturally occuring block the player can stand on, except for wood or leaf blocks. The definition for a liquid block is either a lava or a water block. The heightmap is created by writing the height of the first ground block for each position in the x and z axis, the height value ranges between 0 and 256. To create the liquid map whenever a ground block is found the block above is checked whether it is any form of liquid. If it is a liquid, it gets noted in the liquid-map with a value of 1 for water and a value of 2 for lava, if neither is found the value remains 0. Examples of a heightmap and liquid maps is in Figure 8.

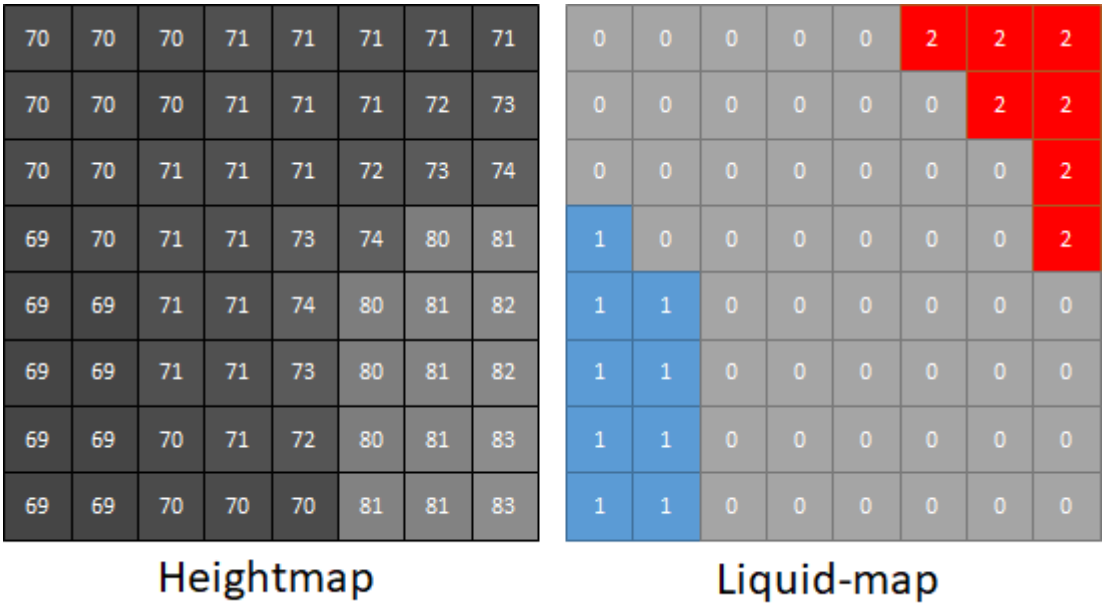


Figure 8
Representation of a heightmap and of a liquid map. The values in the heightmap represents the highest point on the surface for the different positions and the liquid-map represents if there is lava, water or neither on the surface of the different positions.

The different maps are then used to create a weighted directed graph[25] (digraph) where each node represents one position in the heightmap and the weight of the edges is

partly dependent on the difference in height between two nodes. The nodes also contain data from the liquid-map and roadvalue-map in Figure 9.

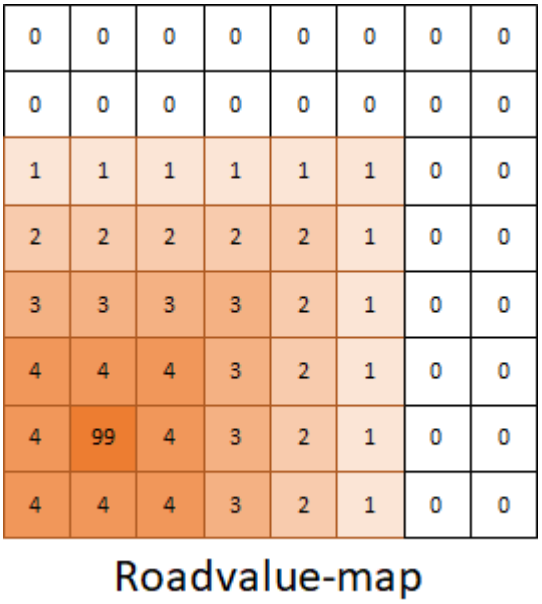


Figure 9
Example of roadvalue-map. Here one tile is designated as a full road, value 99, the other numbers represent how close to a road they are, where larger is closer.

4.3. Road Generation

The road generation for the settlement is done using a multi-agent system. The system consists of several extender and connector agents and the central data structure which is named the Road System.

The digraph's nodes hold the data from their respective positions from the different maps; the height map, liquid map and road value map. They also hold the coordinates and index it is representing and whether it is part of a plot. The edges in the digraph hold information of how difficult it is to travel to nodes.

Then a starting location is selected and finally all its agents execute to create the roads. The road value map is a map which holds the position of roads and the closeness of all tiles to a road.

4.3.1. Road System

The central data structure for the road generation is the road system, not to be confused with the road network which references the actual roads visible in the game world. The road system stores all necessary tile data in its graph to create the final road network. It also contains most functions which are used to modify the game world through the addition, removal or swapping of blocks.

4.3.2. Founding Seed

The first requirement for a settlement is selecting a starting position. This is done by utilizing several depth-first searches (DFS) [25] over the height- and liquid-maps. This search is restricted to create regions of similar height by tracking the average height of the region and disregarding any node which differs too much from the average. After one region is charted the next node which is not part of a region is found by looping through all tiles until one is not part of a region. This is then set to be the beginning of a new region. The complete process outputs multiple regions which denotes connected areas with similar height. Finally the starting location is set as a random position in the largest region which mostly guaranties there is space to grow the settlement.

4.3.3. Extendor Agents

The Extendor Agents extend the road network by performing random walks over the terrain. The agents' walk is only constrained by the initial bounding box and does not account for terrain or liquids. After every step the agents sample their positions to check if they are out of range of the road network. This is detected by comparing the road value in the node where the agent is and if that value is 0 that node is not serviced by the road network.

When an agent finds a position which is not serviced by the road network they use Dijkstra's algorithm [25] to find the closest piece of road and the path to the found piece. The

agent then suggests the path to the road system which evaluates the new road's suitability. The evaluation is done by first comparing the total weight of the suggested path against the product of the Manhattan distance and a tolerance multiplier. The distance is measured between the found unserviced position and the position of the closest road piece. Afterwards, the path's largest individual weight is evaluated whether it has a value of 2 or below. If these two conditions are fulfilled the path is created. Otherwise, it is rejected and the agent continues its walk. This sequence is visualized in Figure 10.

The graph is built so it handles liquids which streamlines the agent's behavior. The graph is configured to not connect any nodes with liquids to any other nodes. This prevents any path forming through any liquids. Additionally, if the agent currently is in a liquid no path is found since there are no edges to traverse through and the search to a road node fails immediately.

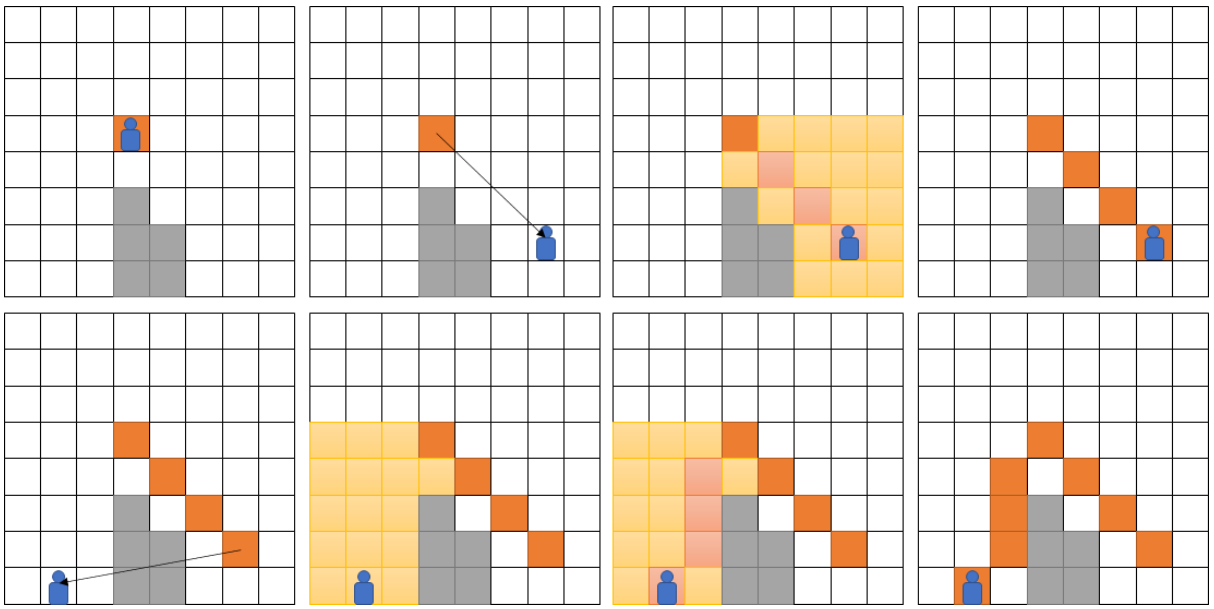


Figure 10

Extendor agent adding roads. Visualizes the extendor agent routine step by step. This shows two successful evaluations and additions to the road network.

4.3.4. Connector Agents

The Connector agents increase the level of connection in the road network, so that the settlement has direct routes between two nearby points. This is done by Connector agents traversing the existing roads and for every step sample a line in a random direction. The line is drawn with the Digital Differential Algorithm (DDA)[26] to ensure any intersection with a road is not skipped. If that line intersects a road within a given range, the agent evaluates the distance to the intersected point through the road system. This is done with Dijkstra's algorithm using the connections between nodes which are roads. Then this distance is compared to the Manhattan distance between the agent position and the intersection point. If this distance exceeds the product of the Manhattan distance and a tolerance multiplier the agent suggests a direct connection between the two points. This suggestion is put through the same evaluation conditions as the ones by the extendor agents. The agent's routine is found in Figure 11.

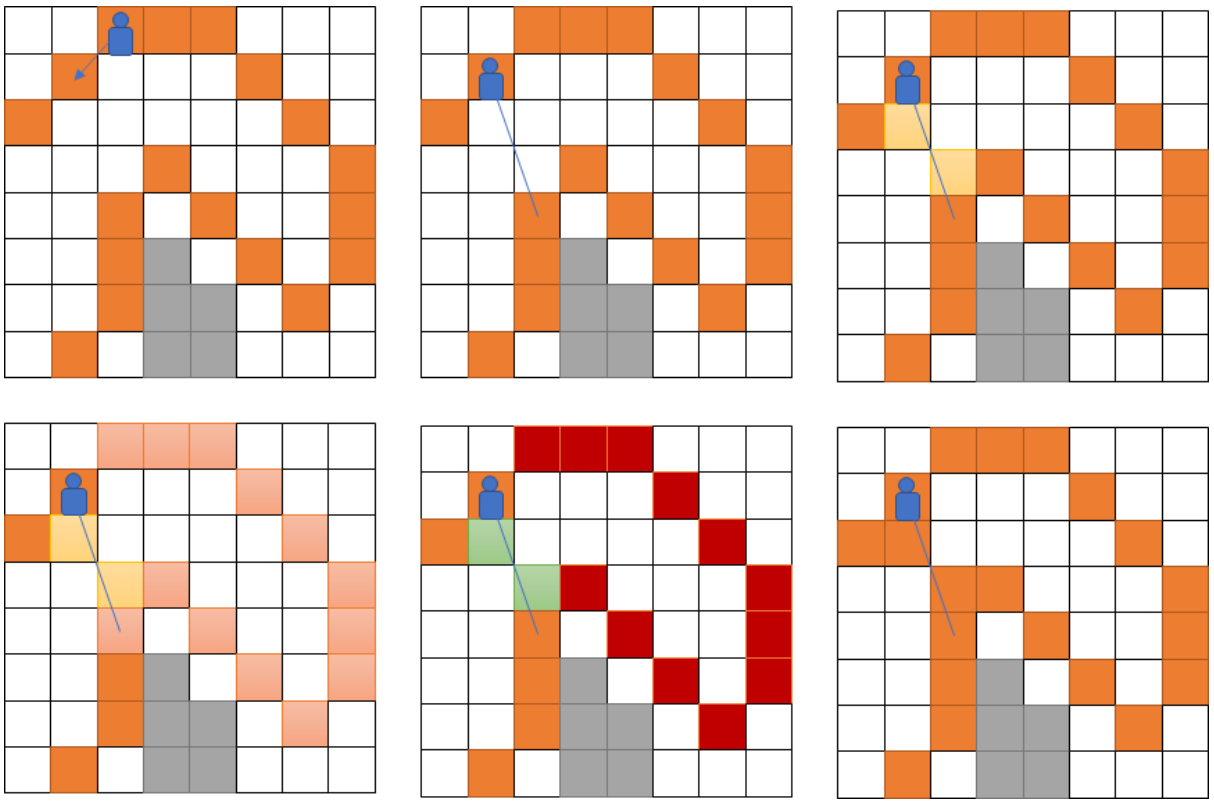


Figure 11

Connector agent adding connection. Visualizes the connector agent routine step by step. This is a successful evaluation.

4.4. Plot Generation

Plot generation happens after all the roads have been generated. This is also done with agents using a random walk. These, however, roam within the range of the road system and at every step check if they are not standing on a road and not overlapping an existing plot. If that is the case they evaluate if a plot can be created at that position. This evaluation begins by finding the closest road by following the surrounding tiles's road values until it reaches the road. Then it uses that position to inflate a plot with the use of a bounded depth-first search. The bounds are defined as the sides of a rectangle and are constrained to span a determined length in the x and y axis, see Figure 12.

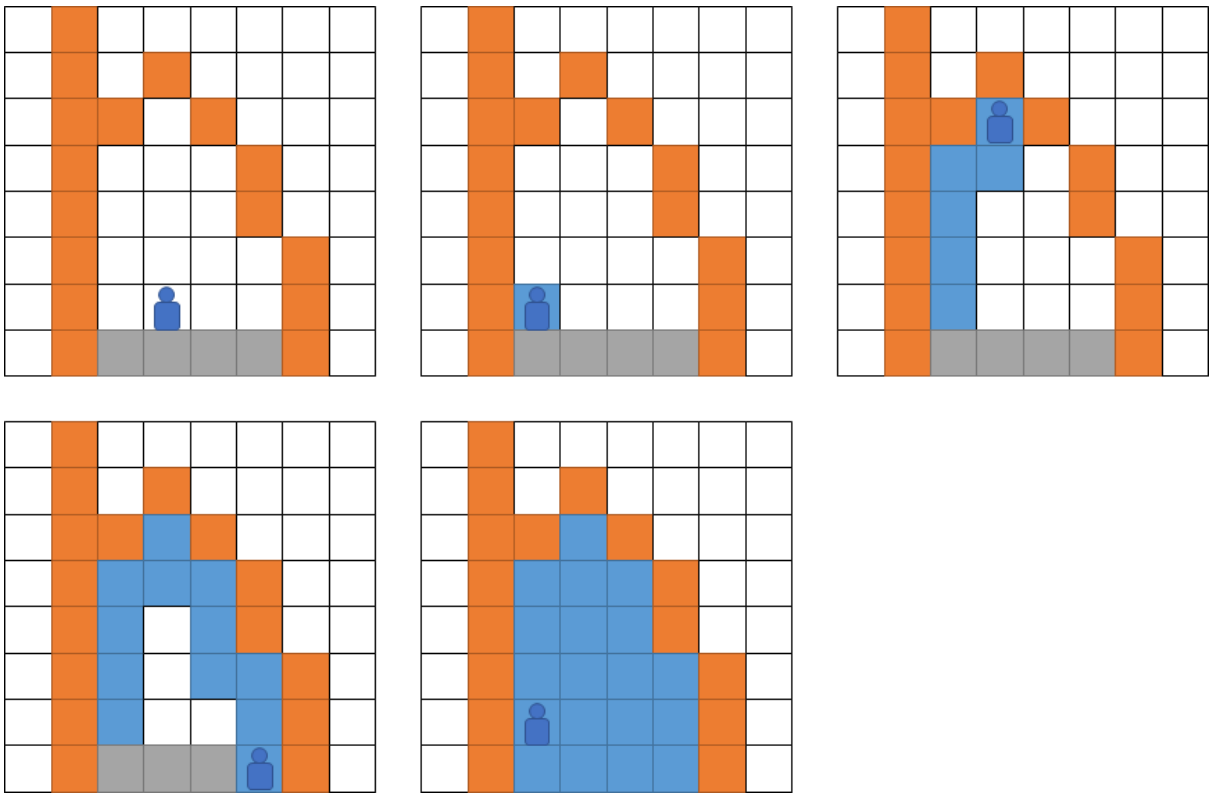


Figure 12

Shows how the plotting agent uses DFS to find a plot. Legend: orange: road, Gray: Tall terrain, blue: current plot white: normal terrain.

This outputs a list of tiles which are first sorted and then placed in a 2D array which recreates the shape of the plot by considering the tile's position in the selection box. The final step is to find the largest rectangle in the collection of tiles; this is what the building generator uses to create the houses. This is done by iterating through each tile in the plot and alternatingly expanding a rectangle to the right and down until no expansion can be done in either direction because it would incorporate a tile which is not part of the plot. Through the iterations the largest rectangle is found and saved, see Figure 13 and Figure 14. This rectangle becomes the input for the building generator.

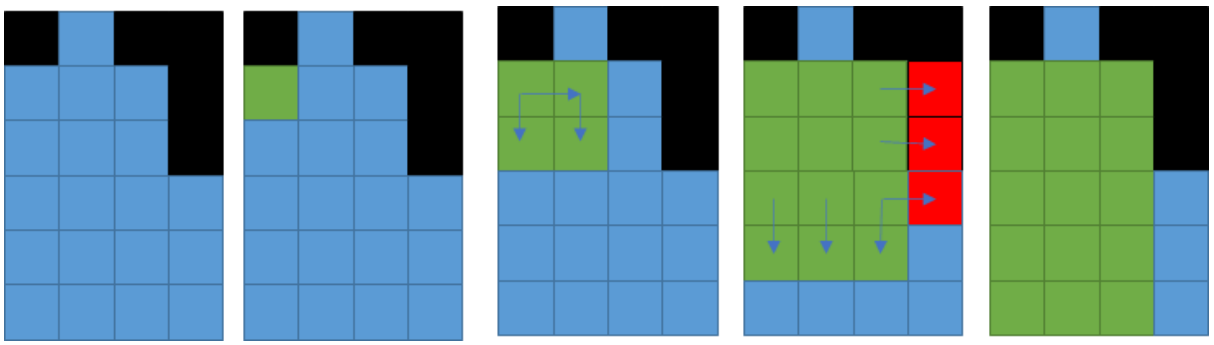


Figure 13

Finding rectangles in plot. It evaluates repeatedly if all tiles to the right and down from the added tiles are able to be added. The evaluation ends when no side is able to expand. Plot extracted from Figure 12. Shows iteration starting from the second plot tile.

Figure 14

Generated houses from the generator. The right one is two storeys and the left is one storey.

4.5. Building Generation

The building generation iterates through the list of plots generated in the previous stage. The generation uses a process with a few random elements to introduce a variance in the appearance. The process starts with creating a module for the building containing its

bounding box i.e. its position in the world and size of the box. There is a chance that the building will be a two storey building, if that is the case an additional module will be created on top of the first one. The module serves as a base for creating floor, walls and roof depending on how many floors the building has. An 1d array is made as an abstraction for each wall, the size is determined by the size of the wall and filled with “None” objects. See Figure 15. Depending on the size of the wall the generator will try to populate it with windows and a door, the door is only placed if the current wall corresponds to the direction to the road that is passed from the road creation. Windows are then placed in a random fashion in these arrays. The window widths are also randomised between 2 to 4 blocks.

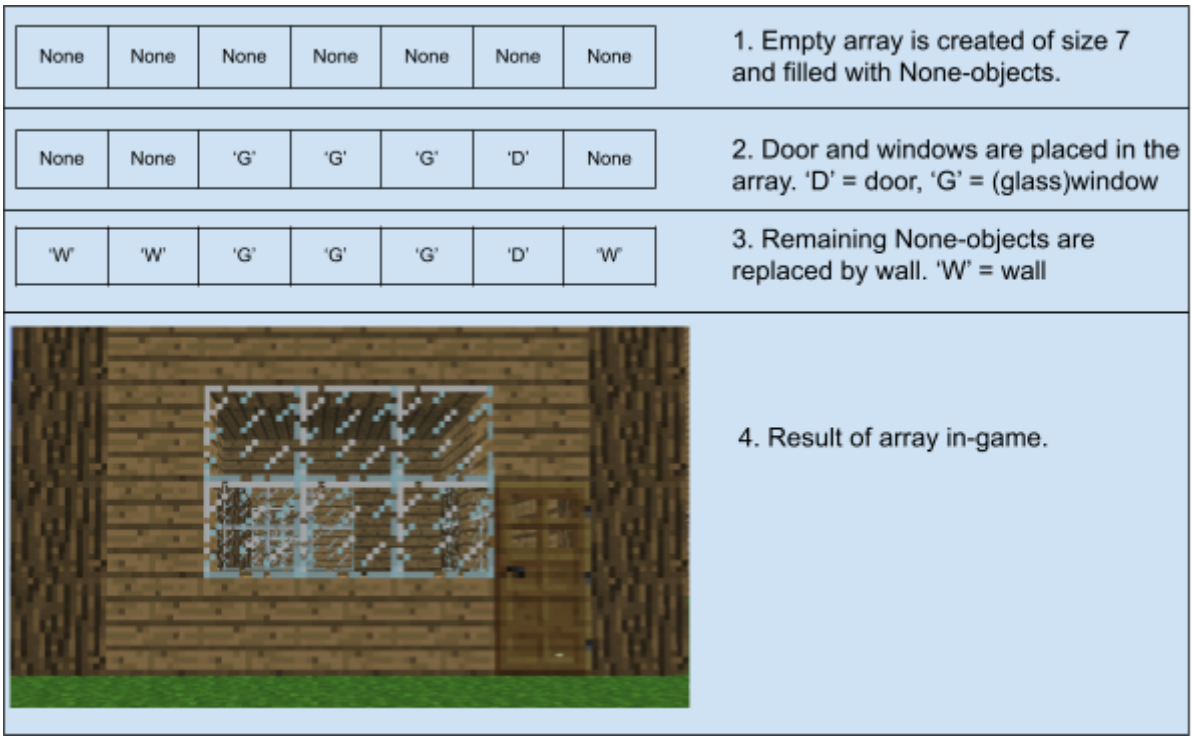


Figure 15
Explanation of how wall abstraction is used.

Each module gets their own floor plan, consisting of a 2d array where the size is based on the modules x and y size. Each wall array created before is converted to the floorplan array. If the building has two storeys an appropriate staircase is selected and included in the plan. The available stairs are made up of 8 pre-defined variants. Variants are

staircases placed in each of the four corners that make up the building as well as which direction it is facing, clockwise or counter-clockwise. Depending on where the door is located, stairs that would block the entrance to the house are removed from the pool of available stair configurations for the current building. See Figure 16 for stairs variants and Figure 17 for in game look. Figure 14 shows the final look of the houses in Minecraft.

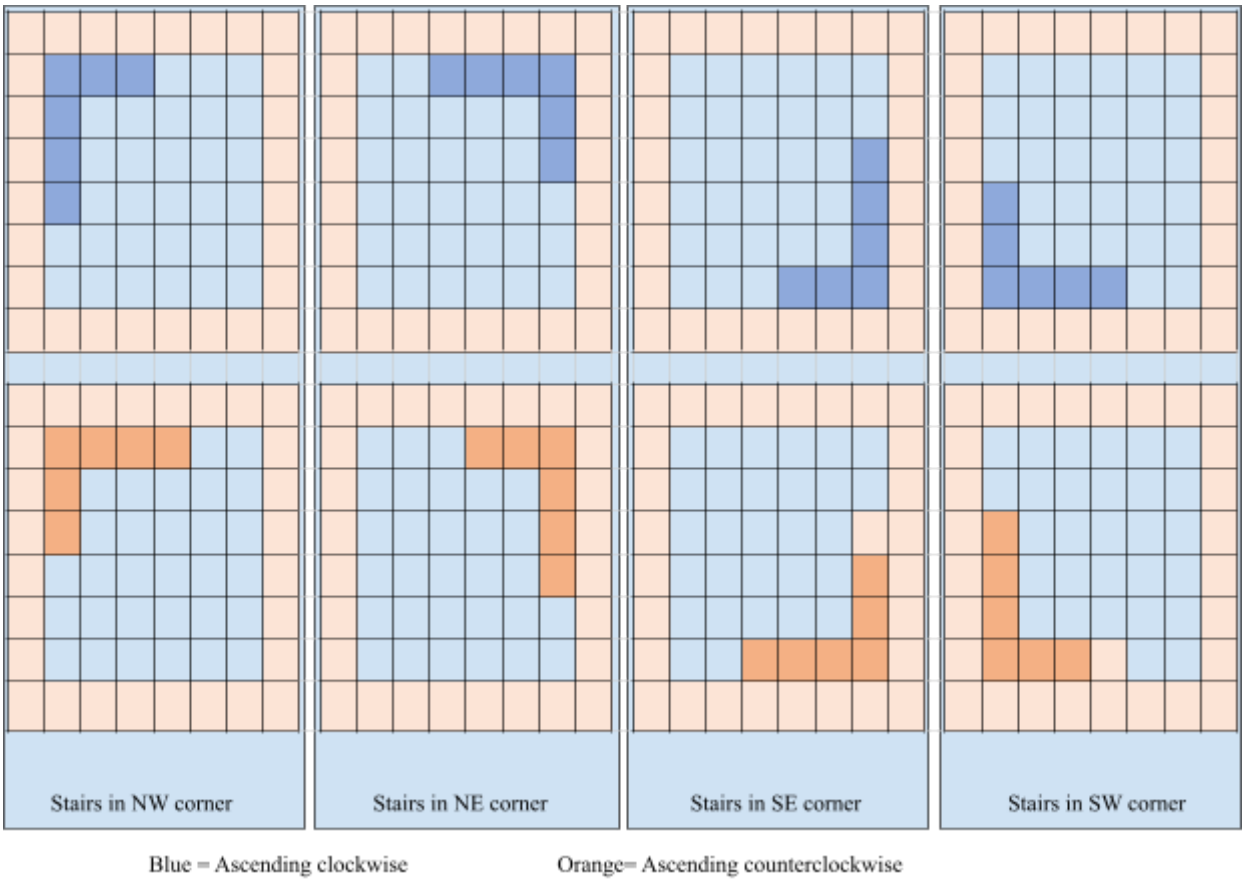


Figure 16
Variants of stairs in 2d representation.

Figure 17
Stairs in ascending-counterclockwise style.

4.6. Furniture Agent

The furniture agent furnishes the houses. It consists of two parts, the furniture the agent uses and the agent itself.

4.6.1. Furniture

To give the agent furniture to work with, a small selection of available furniture is created from the basic blocks that are provided in the framework. The collection of furniture available for the furniture agent consists of a bed, a furnace, a chest and a bookshelf (1-3 blocks), see Figure 18.

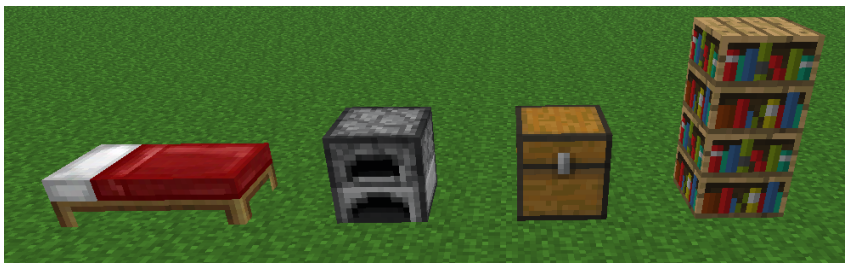


Figure 18
Available furniture in the Artefact. Bed, Furnace, Chest and Bookshelf.

4.6.2. Agent

An agent is placed in the floor plan array, its starting position is decided by looping through the array and locating the door, once located the agent is placed one step into the building. The agent then starts traversing the array in a clockwise fashion on free positions, see Figure 19. Every other step the agent takes it randomizes if it should log that position and use it for furniture placement. After completing a loop of the floorplan the agent is finished with that floorplan and continues to the other level if there is one present. On the second level the agent acts as before but in addition it also places fences around the opening in the ground that gets made to accommodate the stairs leading up the the second level.

Once all floors are traversed, the list of possible placements get iterated through and a randomly selected furniture is placed. To not overpopulate the building with furniture a limit on how many pieces of furniture are allowed per floor based on the building's size.

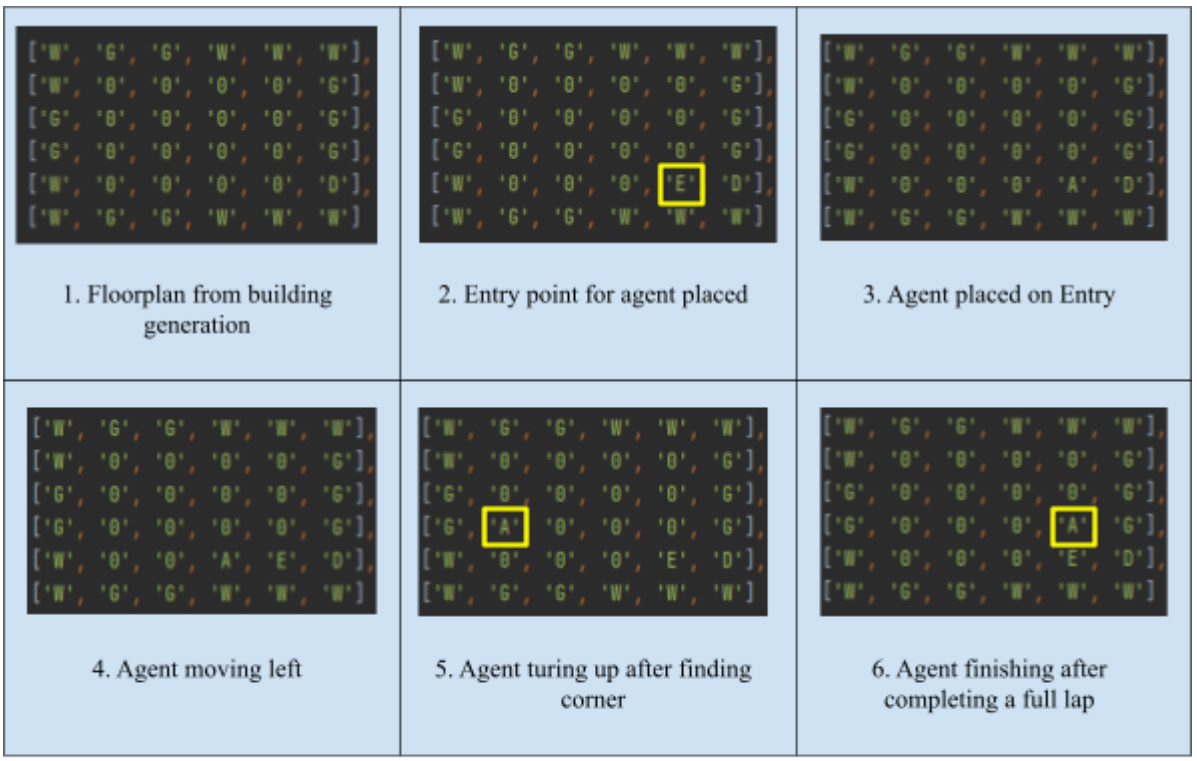


Figure 19
Agent traversing in the floor plan array.

5. Results

This chapter is dedicated to the resulting artefact and detailing what it is able to produce and presents the result from the user study.

5.1. Artefact

The resulting artefact manages to produce settlements with a traversable road network and houses. The settlements manage to adapt to the existing level by weaving the roads around obstacles as seen in Figure 20 where the roads curve alongside the lakes' edges and reinforcing and clearing some terrain to place structurally sound buildings. Figure 20 shows a large settlement where the roads sprawled around the terrain and avoided building on water. Some problems are noticed in the middle of the map with multiple roads and no clear purpose to them, also noticeable in Figure 23. Figure 21 shows a smaller settlement but with an efficient looking road connecting the buildings. Figure 22 shows how foundations are built to support the building when the terrain has too big variance in height.

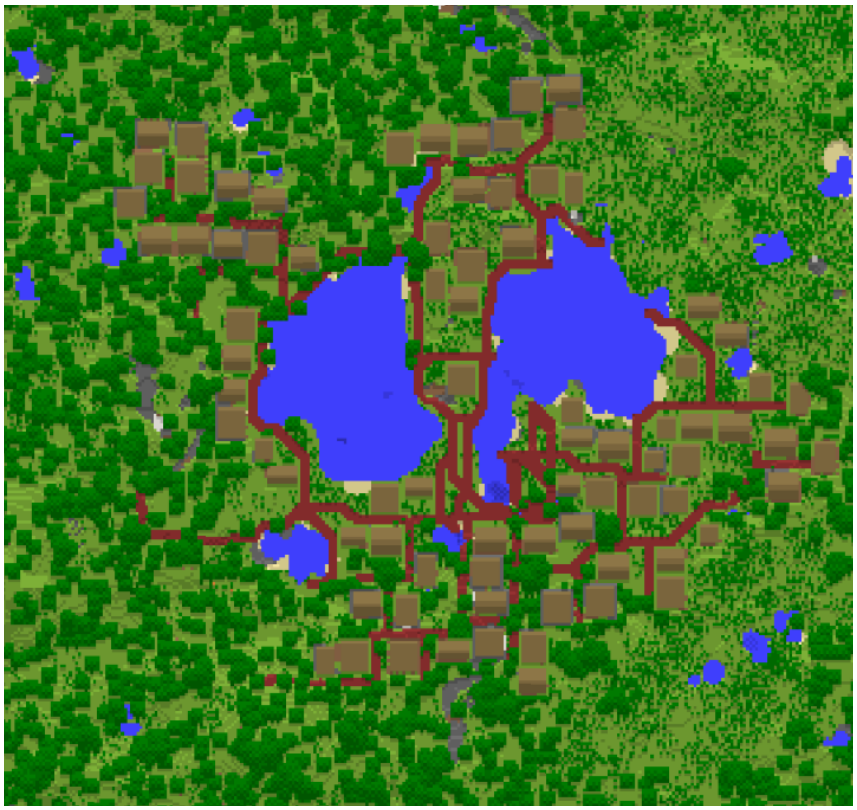


Figure 20

Overview of artifact output on map A, double lakes.

Figure 21

Overview of artifact output on map B,mountain.

Figure 22

Fundations are built to support the buildings.

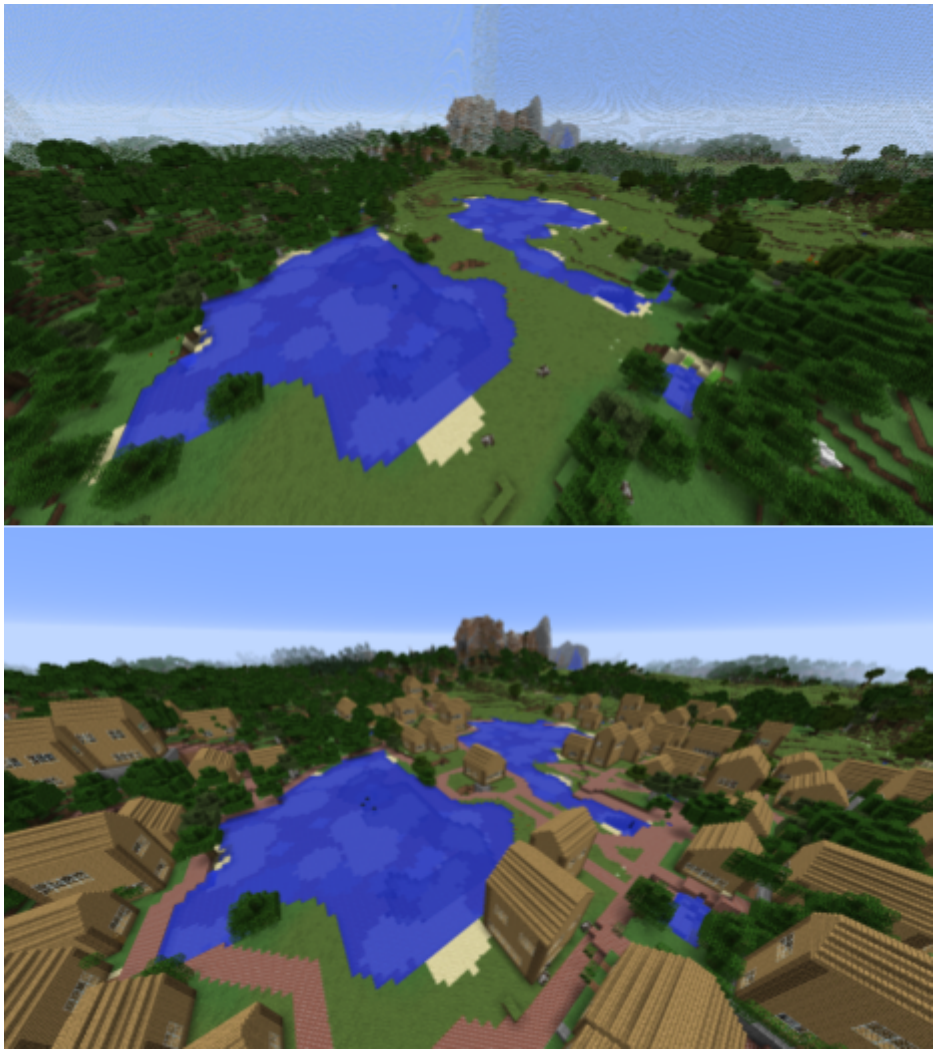


Figure 23

Map A double lakes, before and after artefact generation

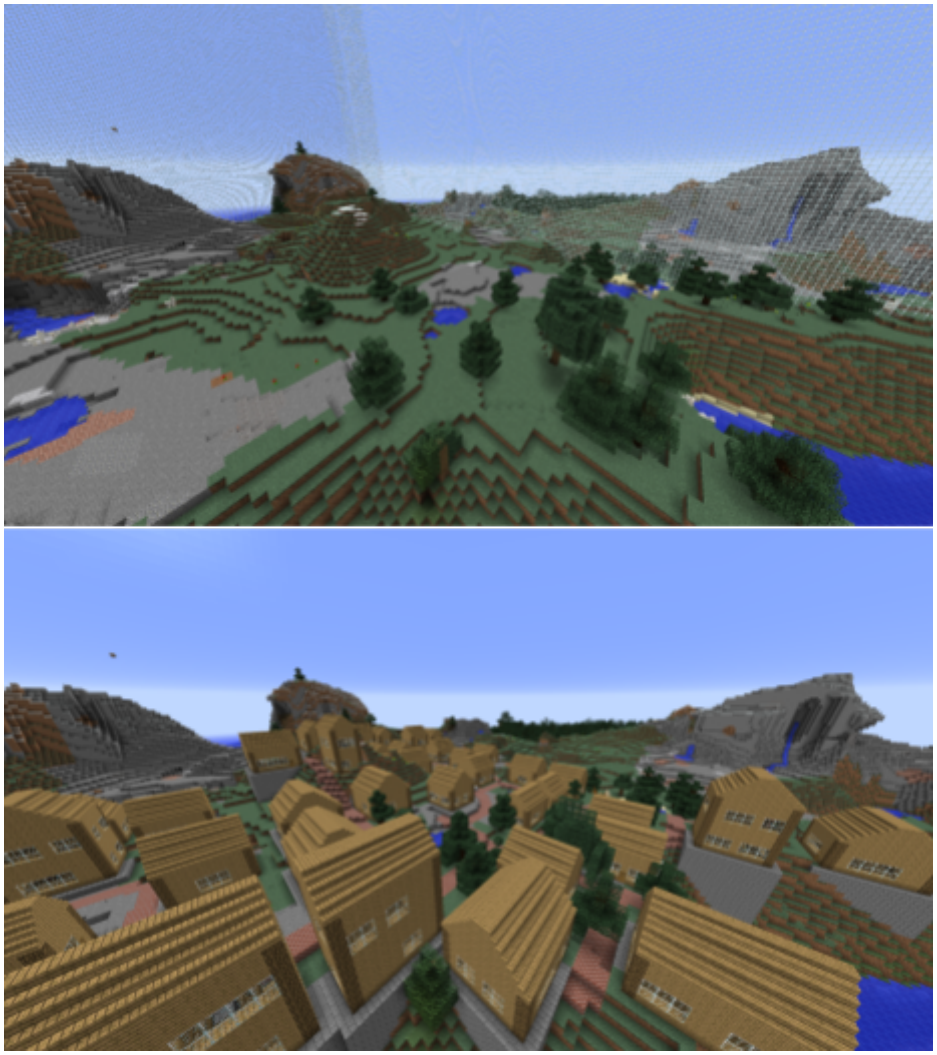


Figure 24
Map B Mountain, before and after artefact generation.

The buildings themselves show similarities to each other and are not uniquely different to one another, but get a small increase of customization with the different sizes of the building and with the placement of windows and doors. Building placement is realistic in the terrain, they don't appear in unreachable places and are not placed inside of ground or mountains. See Figure 25 and Figure 26.

Figure 25

Alternative view of settlement on Map A double lakes.

Figure 26

View of buildings on a hill on Map A double lakes.

The insides of houses are plain and typically only 3-4 instances of furniture. All furniture that was added to the generator failed to render in the maps that were a part of the study that is shown in Figure 27. This is a problem with how they were placed in the editor where the beds and chests needed an entity to render, an entity is generated once the blocks are updated in game.

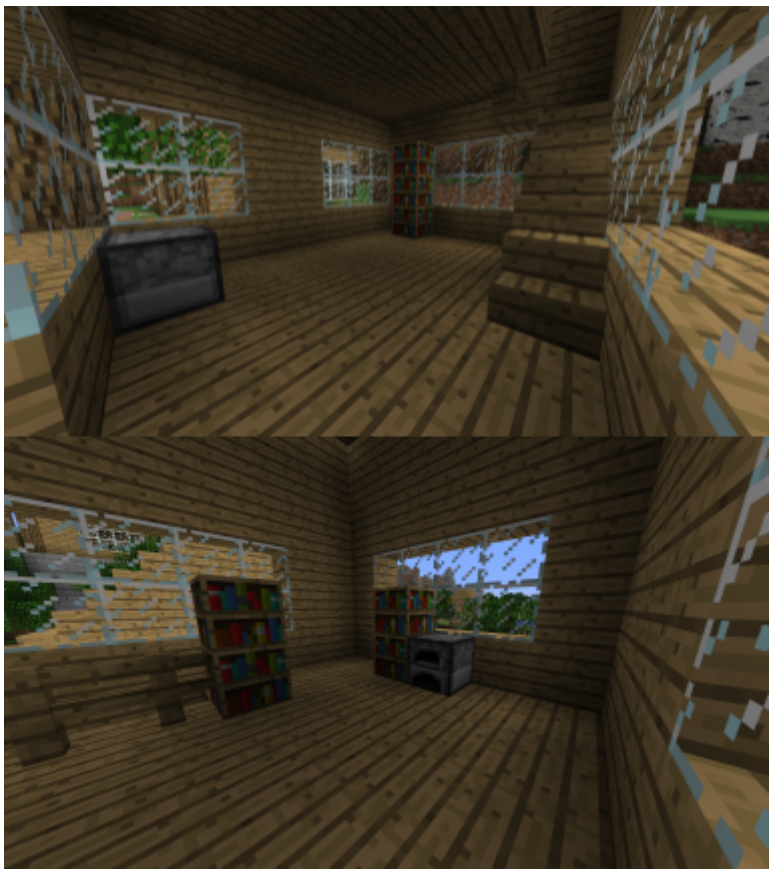


Figure 27
Indoors view of building. Only the furnaces, bookshelves and fences are visible however chests and beds are placed but fainting to render in-game.

The generator generally has an execution time of 10 minutes with a 256x256x256 block selection. Larger selections increase the time taken exponentially.

5.2. User Study

The user study was conducted on nine participants. Each participant was given instructions on how to conduct the test and a randomized order which the generators should be explored. This prevents the scoring to be affected by participants getting inattentive towards the end of the survey. Before exploring the maps which were altered by the generators, they explored the maps without any alterations to see the unaffected terrain. Participants conducted the study on their own computer without supervision. Participants were encouraged to spend between 5-10 minutes per map to get a complete feeling for it

before continuing. This was however not controlled in the survey. After exploring each generator the participant answers a questionnaire, scoring the generators in adaptability, functionality, narrative and aesthetics and optionally adding a comment on each of the generators. See Figure 28 and Figure 29 for overviews of maps used in the survey.

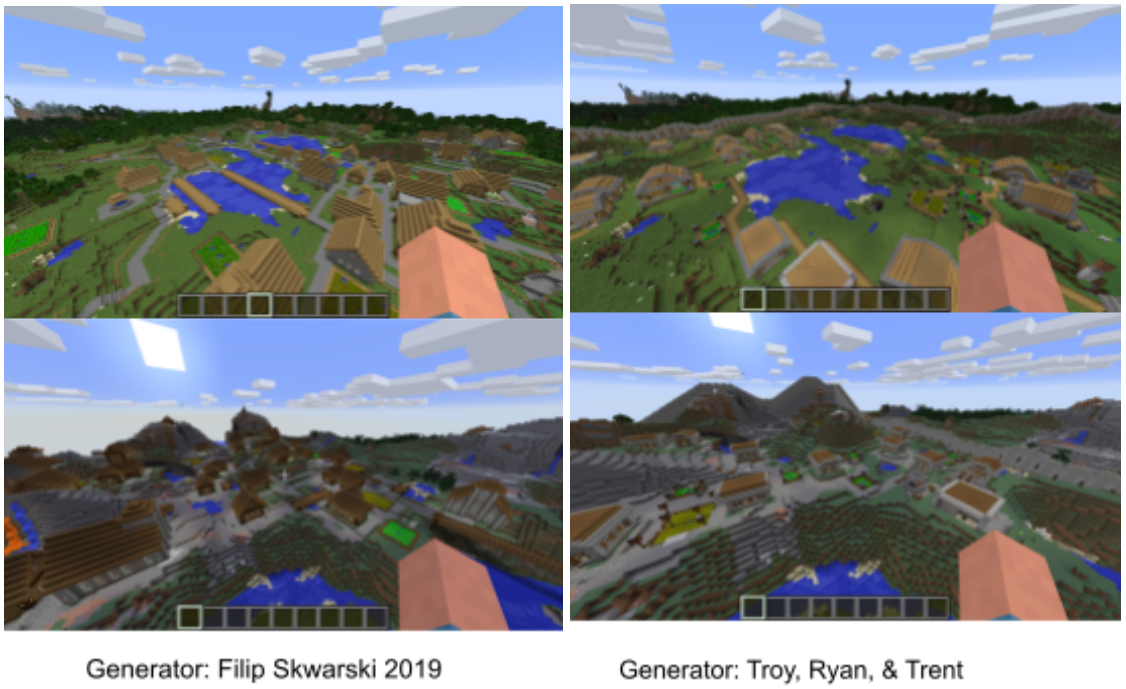


Figure 28

Images of the generators used in the user study on the maps provided to the participants. Shows Skwarski's and TRT's generators.

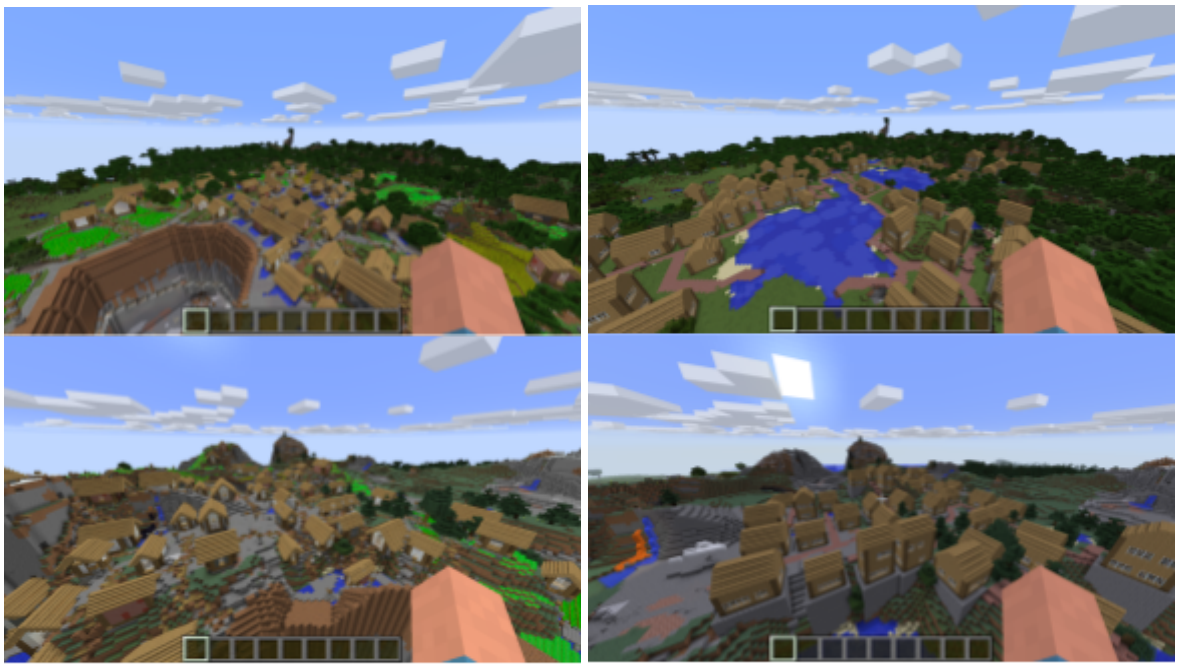


Figure 29
Images of the generators used in the user study on the maps provided to the participants. Shows Mason's and this thesis' generator.

In total score the artefact places 4th place with a score of 3,1 points. David Mason's places 3rd with 3,9 points. Filip Skwarski's generator got 2nd place with 4,8 points. Finally Troy, Ryan & Trent's gets 1st place with 4,9 points. See Figure 30.

The artefact scores comparable to David Mason in adaptability at a score of 3,1 points. In functionally the artifact scores 3,0 points; 0,2 points more than David Manson. In narrative the artefact scores the lowest at 2,2 points, closest are Filip Skwarski at 3,7 points. The artifact gets 4,1 points in aesthetics, a bit behind the other generators that are clustered together around 5,1 points. See Figure 31.

The complete data from the questionnaire is available in Appendix B. The maps used in the study are available to download [27].

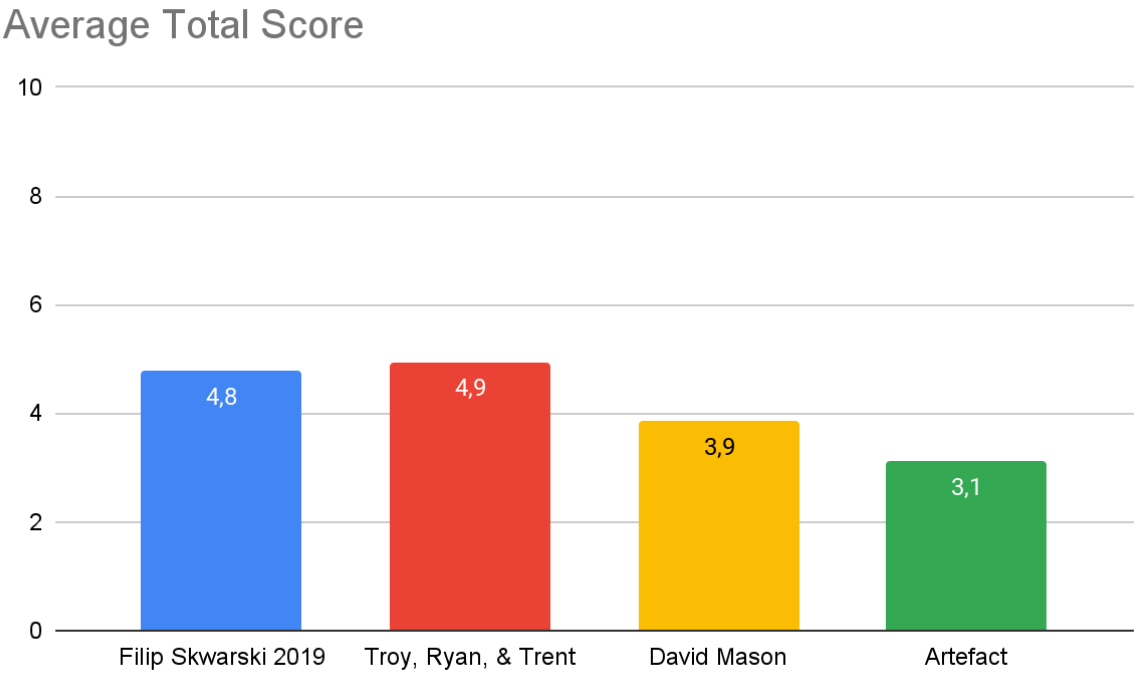


Figure 30

Average Total Score for the generators in the user study

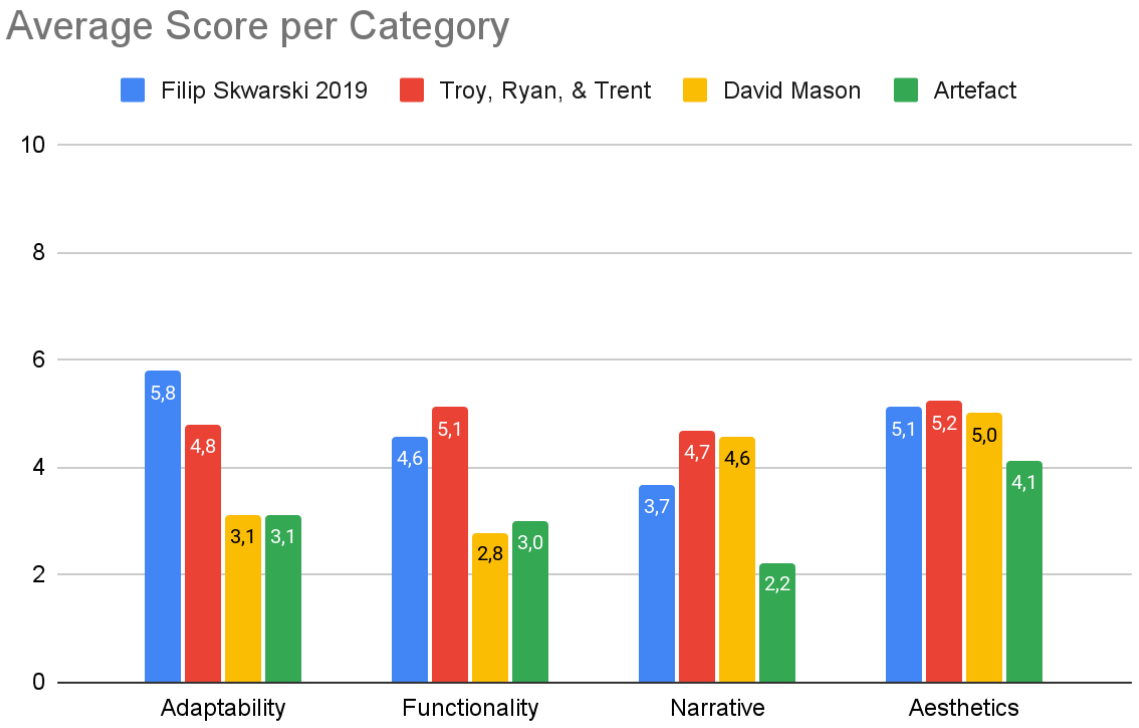


Figure 31

Average Score per aspect for the generators in the user study

5.3. GDMC Competition

The artefact was also submitted to the annual GDMC-competition[28]. Under the name MAU_AE_JF the generator placed 17 of 20 contestants. Getting a final score of 3,07 points. The scoring process for the competition and this survey are similar and comparable. The artifact was not entered in the Chronicle challenge. In the result presentation made by Salge [29], he mentioned that there was an increase of use of agent based solutions. The generator was run on two different maps, one map had a landmark volcano, the other one was a hybrid map, where the generator had to take into account pre existing buildings created by another generator.

Entry	Name	Rank	Adaptability	Functionality	Narrative	Aesthetics	Final score
10	MAU_AE_JF	17/20	3.647	3.059	2.412	3.176	3.073

Table 1
This work's Ranking and scores in the GDMC-competition 2021

Entry	Name	Adaptability	Functionality	Narrative	Aesthetics	Total
10	MAU_AE_JF	10th	14th	19th	17th	17th

Table 2
This work's Ranking for the individual criterias in the GDMC-competition 2021

The comments from the judges give both praise and criticism. Overall the road generation got good comments, saying that it is clever and making interesting paths, multiple judges compliment how the road follows the river, see Figure 32. The buildings offer some variety but are rather plain looking, also the material used for the building does not fit for the biome, see Figure 33.

The interior is lacking but there is potential for improvement. The placement of the houses was appreciated, and only one comment mentioned that the house carved its way into the terrain.

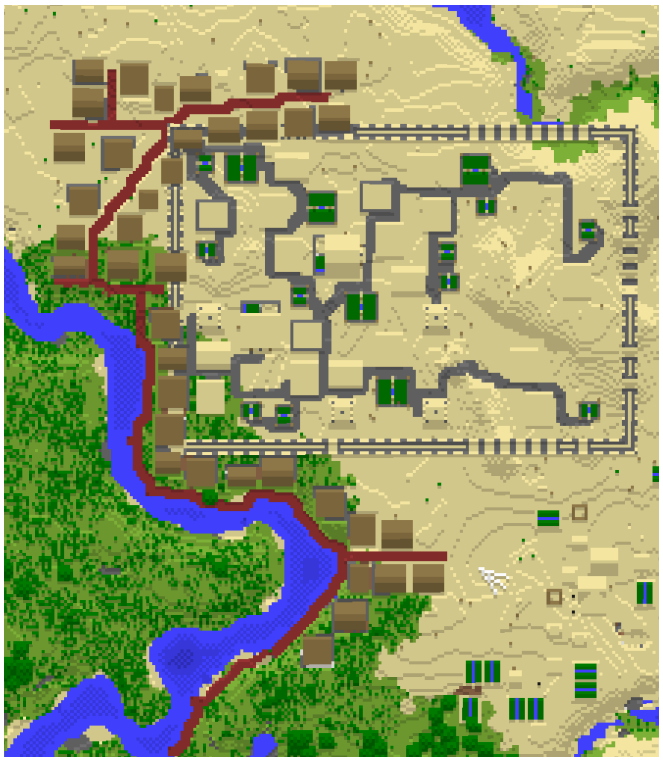


Figure 32
Overview of Hybrid map from GDMC-competition

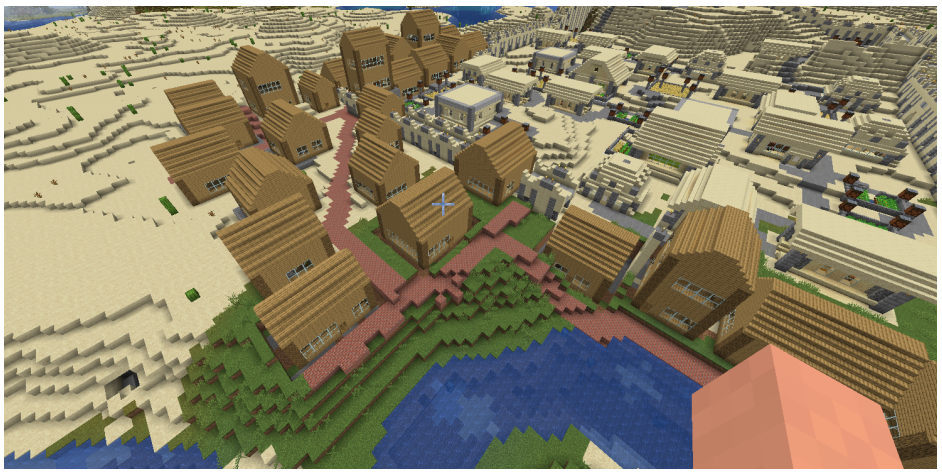


Figure 33
Artefact in GDMC hybrid map.

For detailed scoring and comments from the competition see Appendix C.

6. Discussion

The discussion header contains analysis of how the generator and evaluation accomplishes the goal of addressing the research question.

6.1. Discussing the Final Generator

The artefact created during this thesis is a filter for the MCedit framework written in Python 2.7. The filter when applied in the framework is capable of generating a settlement with its own road network and buildings in it.

The roads are created with a multi-agent system with specialized agent types. These created a mostly believable road network where the path of least resistance was favored. The road network was fully traversable by the player character and it managed to utilize much of the given area, unless completely prevented by cliffs and rivers, while keeping enough space for houses.

The system still contains several flaws. One of these is the generation of different elements happens in a single pass such as first all roads are done then the houses are added. This consequently, removes any ability to adapt to its own constructions. Making the road agents invalidate desirable building placements which would benefit the settlement. Another problem that also was noticeable is that tree leaves located above the road were not removed occasionally resulting in hovering trees. Notice in Figure 20 where the red road is obscured by the green, however this is not something that is entirely negative, watching Figure 24 where a tree is located close by the road and the leaves hanging over the road suits the environment.

The houses themselves are all built on plots generated and all follow the same formula of maximizing the size of the building in the size of the plot. To introduce some variety in the buildings a lot of parts were randomized, which was done as an easy way out programming

wise, not without consequences as some buildings show problems mainly with the furniture agent. Having a more active agent that would design the house and interior based on needs and available resources is something that was stiven for but was scrapped because of time constraints.

In comparison to the other MAS generator found in the official competition 2020 created by Brightmoore, his generator has agents capable of creating complex houses and has a system for tracking and printing the agents history, giving it an edge in both narrative and aesthetic. Similarities between his and this study's MAS is they both use agents to roam the terrain and evaluate if they are able to build at the different positions during the generator's execution.

6.2. Design Choices

During the creation of the artifact several decisions were made regarding implementation. These create foundations for the implementation which determine several aspects of the resulting artifact. The choices here were at the time the most suitable to reach the wanted performance of the generator.

The use of Manhattan distances as a metric in the evaluation function was made because it simulates the actual distance traversed in a city. It was also chosen for its lower requirement regarding computing power to calculate, compared to euclidean distance.

The pathfinding algorithm chosen for the agents is Dijkstra's algorithm because many use cases did not have a specified goal node when it began. Rather, the goal is the closest node out of a set and it was not possible to know the best target beforehand. This condition disqualified the use of the A-star algorithm since it functions by prioritizing the nodes closest to a pre-specified target, something which is not known in this implementation.

To be able to compete in the GDMC-competition, functionalities of the generator had to be scrapped to manage the time frame. Problems were encountered with getting the MCEdit framework to function and using the python language was a new experience which took time to understand and learn how to use. This led to a lower performing generator than initially estimated.

6.3. Data Discussion

For this thesis there are two independent comparisons for this generator. The comparisons also ran simultaneously which inhibited the opportunity to improve the artifact with the feedback prior to any evaluation. Consequently the artifact is the same in the two comparisons. The first comparison is made during the survey for this thesis. Here a selection of well placed entries in previous years of the competition and this generator were compared. The second comparison is the GDMC 2021 competition where the generator was entered. In both comparisons the other entries are unknown with regards to work hours and expertise and thus consequently does not provide a stable comparison foundation.

In both comparisons this generator was out competed when accounting for all the aspects. From the survey the data in Figure 31 suggest several aspects are affecting this, these are, in order of severity, narrative, adaptability and functionality. Then in Table 2 the low rankings are the narrative and aesthetics aspects. This leads to two hypotheses: there is improvement to be done for all aspects and the narrative aspect is the one which detracts the most from the generator's overall performance.

Starting with the narrative, this aspect was the least prioritized when developing this generator and consequently does not have any specific features related to it. The generator’s narrative score would be improved by adding a feature which would generate the settlement in steps and account for time when creating new parts and simulating existing ones.

Aesthetics is accounted for in the generator by using the same materials and general shape for each element in the settlement. This aspect can be improved by creating different kinds of buildings and structures while keeping the same style and also creating more beautiful buildings. The generator manages the basics of this aspect by being uniform in the constructions throughout the settlement.

The functionality was a high priority for this generator. The features tied to it relates to the furnishing and the accessibility of the roads. The largest omission relating to this aspect is lighting, which is non-existent. This problem would be solved by adding lighting agents which would be tasked to place light posts regularly along the roads and extending the furnishing agent to place lights inside the buildings.

Adaptability was another high priority. The main feature for this aspect is the road's avoidance of water and steep inclines, see Figure 34 how the road follows the riverbank. To improve this aspect's score more consideration should be made to the surrounding materials which would determine the building material palette dynamically.

The issues discussed do not necessarily represent a failure of the multi-agent methodology. Rather, many of these issues could be solved within the framework and using agents such as the lighting and building type by expanding the system. They were not in this implementation because these features were not a priority compared to the features implemented when developing the artifact. More suggestions for improvements are described in the future works section.

In the comparisons, the layout of the settlement is appreciated by the comments. Since much of the work done focused on this aspect it is also an expected outcome. The focus was on how agents can be used in settlement generation. The uses explored were road system generation, plot generation and furniture placement. These features were consistently praised in the evaluations. Which indicates that the use of agents is good for these applications. And

by extending the functionality of the existing agents and developing new ones would improve the generator even more.

Figure 34
Wide shot over river from Hybrid map from GDMC-competition

6.4. Survey Analysis

The survey was designed to as closely as possible mimic the judging process of the GDMC-competition to utilize it as another measurement. While this did get the intended effect and the result from both are similar it also gave rise to several other problems.

The first of these is that the survey was unfocused and consequently did not manage to indicate how well the agent performed, only that it managed to do something. This also led to the instructions available to be interpreted differently which shows with the wildly different individual scores in Appendix B. There was also no requirement of having any experience of the game to be a participant of the study, which can explain why there is a big range of scoring, in the case of Narrative on David Mason's generator it scored one 0 points and one 8 points. An 8 points difference. When the participants are the most coherent there is a 5 points difference. This indicates that scoring in this setting is subjective. Participants without knowledge of the game might miss ingame factors that would otherwise affect the scoring, for instance that in Minecraft, light sources keeps monsters away from spawning

keeping the settlement safe, while a more experienced participant would notice this straight away and therefore might give the generator a lower score.

Another problem of the survey was that it took a long time to complete which could have been the reason for the few responses collected, which in turn makes the results highly affected by outlier responses. The time could have been reduced by designing a survey which only measures the specific parts the generator performs with such as land usage and road layout.

7. Conclusions

This study wants to answer how a MAS style settlement generator could be implemented. The artefact produced takes inspiration from Lechner [6] city modeling and implements its own version of road network agents. The road network agents act as a foundation for the settlement generator as all subsequent processes are dependent on it. Lechner et al. [6] has additional interesting aspects in their paper that were not possible to implement for this study; more of these are mentioned in the future work chapter.

The artefact shows flaws in its implementations when comparing with other generators but still manages to create interesting and creative ways of solving problems not seen in other generators. Thus this style of a MAS is suitable to generate settlements in minecraft.

When using the same evaluation method as the official GDMC-competition, the solutions created for the artefact fell behind most of the ones used in comparisons, especially in the narrative aspect. The point difference between the paper's user study and the official GDMC-competition is a 0,03 point difference. Though no statistical conclusion can be made from this, it's an interesting occurrence.

The creative opportunities a MAS provides is limited by the implementation and ingenuity of the system. To increase its scoring in the different categories, possible improvements are presented in the future work chapter. When compared to the top placing entries of the competition this implementation does not reach them but is also not completely outclassed.

8. Future Work

While working on the generator and writing up on it plenty of ideas on how to improve this came to light. These improvements relate to both the actual implementation of the generator and the design of the user study and are thus split into two sections.

8.1 Generator

The final generator functioned well however many features envisioned for it were cut due to time and expertise constraints. After completing the generator some performance improvements were identified as well.

8.1.1 Performance Improvements

The first type of improvements concerns optimizing the generator's runtime. The first is optimizing the pathfinding algorithm to create local graphs which are vastly smaller than the full-sized graph used currently. This optimization should allow the generator to greatly reduce the running time and consequently function over a larger area. Another option is by using a direction map [31]. Because a direction map is created only once during initialization and during runtime it only updates the tiles affected by a change.

Another performance problem stems from the Python language. Since it is an interpreted language it is not as suited to computationally heavy programs. The GDMC competition provides other alternatives to submitting a generator. The alternative used here

was a filter file for MCEdit but the others are either a HTTP server or a program capable of modifying minecraft files directly. This third option can be developed in any language and thus use a more suitable language such as C++ or C#.

8.1.2 Additional Features

As mentioned in the text, iterative generation would benefit the artifact because then the final settlement could be generated over several steps and simulate the passage of time. This could also be coupled with an improved survey of the given terrain to guide development towards specific goals. For instance, after an initial pass where the core of the settlement is created the next pass could identify a forest nearby and as such try to develop towards it and generate forestry related buildings.

Future generators could implement more agent types such as bridge agent, stair agent, serpentine road agent, speciality house agent, zoning agent, and more. This would greatly increase expresitivity to the generator to be able to construct different building types such as warehouses, docks, smithies, town centres but also gardens, parks, plazas and farms.

A block palette is a common feature of other generators that this generator does not have. The palette would benefit the generator's adaptability and aesthetic scores and would make a good addition to any future generator.

One interesting improvement is to expand on the usage of zones described in [6], where zones could be classified as either residential or commercial areas. In Minecraft additional dimensions can be added to this. For instance, residential areas can have different wealth levels affecting the materials and appearance of the building. And for commercial areas this could mean differences in agricultural, mining, governance, social or shop buildings. With possibility of unique features.

Focusing and improving the survey of the landscape for selecting the starting position and taking stock of available resources would allow the settlement to specialize. The kinds of

specializations could focus one one or a few types such as a military outpost, forestry settlement or possible fishing village.

8.2 Survey

The survey was modeled wholesale after the judging process of the GDMC competition because that would make the results of the survey and the competition comparable. While this is the case it did limit the kind of analysis available to be performed since it encompases the whole of a settlement and not a specific aspect.

Another addition could be considering the participants' experience in minecraft since this value could greatly affect how any construction in the game is perceived.

An alternative survey would focus on the macro scale of the settlement generation. First select a few generators on some relevant criteria. Then each generator would output multiple settlements, in the range of 10 or more. Finally the participants are asked to rank the different settlements from overview images. The result from this is meant to show how settlements adapt their layout to the terrain.

References

[1] Noor Shaker, Julian Togelius, and Mark J. Nelson (2016). *Procedural Content Generation in Games: A Textbook and an Overview of Current Research*. Springer. ISBN 978-3-319-42714-0.

[2] Amato, A. (2017). Procedural content generation in the game industry. In *Game Dynamics* (pp. 15-25). Springer, Cham.

[3] Salge, C., Green, M. C., Canaan, R., & Togelius, J. (2018, August). Generative design in minecraft (GDMC) settlement generation competition. In *Proceedings of the 13th International Conference on the Foundations of Digital Games* (pp. 1-10).

[4] "The GDMC competition", *GDMC*, 2021. [Online]. Available: http://gendesignmc.engineering.nyu.edu/results. [Accessed: 04- Mar- 2021].

[5] Fridh, Marcus, and Fredrik Sy. "Settlement Generation in Minecraft." (2020). (MAU Bachelor's Thesis)

[6] Lechner, Thomas, Ben Watson, and Uri Wilensky. "Procedural city modeling." *In 1st Midwestern Graphics Conference*. 2003.

[7] A.I. Design. 1980. *Rogue*. Game [DOS]. (1984). Epyx, Inc, San Francisco, US.

[8] Ian Bell et al. 1984. *Elite*. Game[BBC Micro], (Sep 1984). Acornsoft Limited, Cambridge, England.

[9] Beneš, Jan, Alexander Wilkie, and Jaroslav Křivánek. "Procedural modelling of urban road networks." *Computer Graphics Forum*. Vol. 33. No. 6. 2014.

[10] Blizzard Entertainment Inc. 1996. *Diablo*. Game [PC], (31 Dec 1996). Blizzard Entertainment Inc., Irvine, US.

[11] Doran, Jonathon, and Ian Parberry. "Controlled procedural terrain generation using software agents." *IEEE Transactions on Computational Intelligence and AI in Games* 2.2 (2010): 111-119.

[12] Parish, Yoav IH, and Pascal Müller. "Procedural modeling of cities." *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*. 2001.

[13] Mojang. 2011. *Minecraft*. Game [PC], (18 Nov 2011). Mojang Synergies AB, Stockholm, SWE. Last played:2021-03-15

[14] Sun, Jing, et al. "Template-based generation of road networks for virtual city modeling." *Proceedings of the ACM symposium on Virtual reality software and technology*. 2002.

[15] Williams, Benjamin, and Christopher J. Headleand. "A time-line approach for the generation of simulated settlements." *2017 International Conference on Cyberworlds (CW)*. IEEE, 2017.

[16] "DARPA Challenge," *Home | DRC Finals*. [Online]. Available: https://archive.darpa.mil/roboticschallenge/. [Accessed: 15-Mar-2021].

[17] "ROBOTCHALLENGE," *RobotChallenge*. [Online]. Available: http://www.robotchallenge.org.cn/. [Accessed: 15-Mar-2021].

[18] "GVGAI," *AI and Games*. [Online]. Available: https://aingames.cn/. [Accessed: 15-Mar-2021].

[19] Kelly, George, and Hugh McCabe. "A survey of procedural techniques for city generation." *ITB Journal* 14.3 (2006): 342-351.

[20] *MCEdit*. [Online]. Available: http://www.mcedit.net/. [Accessed: 15-Mar-2021].

[21] A. Dorri, S. S. Kanhere and R. Jurdak, "Multi-Agent Systems: A Survey," in *IEEE Access*, vol. 6, pp. 28573-28593, 2018, doi: 10.1109/ACCESS.2018.2831228.

[22] "GDMC competition results," *2020 Settlement Generation Competition - Generative Design in Minecraft*. [Online]. Available:

https://gendesignmc.wikidot.com/wiki:2020-settlement-generation-competition. [Accessed: 18-Mar-2021].

[23] K. Peffers, T. Tuunanen, M. A. Rothenberger, and S. Chatterjee, “A Design Science Research Methodology for Information Systems Research,” *Journal of Management Information Systems*, vol. 24, no. 3, pp. 45–77, 2007.

[24] CD Project Red. 2011. *The Witcher 3*. Game [PC], (19 May 2015). CD Project, Warsaw, POL. Last played: 2019

[25] R. Sedgewick and K. Wayne, *Algorithms, 4th Edition.* Addison-Wesley, 2011, p. I–XII, 1-955.

[26] Museth, Ken. "Hierarchical digital differential analyzer for efficient ray-marching in openvdb." *ACM SIGGRAPH 2014 Talks*. 2014. 1-1.

[27] Maps used in survey: https://drive.google.com/file/d/1ONEtTeQDrdYr3Gm4HwB1HF0MZ7Y3tywb/view?usp=sharing

[28] GDMC Result 2021: https://www.dropbox.com/sh/mr6s5a2zb5w669j/AAAqtun6UR-Jz_8S5fYwZgGJa?dl=0 2021-08-08

[29] Christoph Salge. "2021 Winners of the AI Settlement Generation Challenge in Minecraft (GDMC)" *YouTube*, (06 Aug 2021). Available: https://www.youtube.com/watch?v=uYUIZUGPNX8. [Accessed: 18-Aug-2021].

[30] Adrian Brightmoore (2020) https://github.com/abrightmoore/GDMC2020_ChronicleChallenge [Accessed: 2021-08-22]

[31] Jansen, Renee, and Nathan Sturtevant. "A new approach to cooperative pathfinding." *Proceedings of the 7th international joint conference on Autonomous agents and multiagent systems-Volume 3*. 2008.

[32] SpeedTree. (2021). Interactive Data Visualization, Inc. [Online] Available: https://store.speedtree.com/ [Accessed: 10- Aug- 2021]

[33] "Settlement Generation Challenge 2019 - Filip Skwarski 2019", *Gendesignmc.wikidot.com*, 2021. [Online]. Available: https://gendesignmc.wikidot.com/wiki:2019-settlement-generation-results#toc6. [Accessed: 2021-10-10]

[34] "Settlement Generation Challenge 2020 - Troy, Ryan, & Trent", *Gendesignmc.wikidot.com*, 2021. [Online]. Available: https://gendesignmc.wikidot.com/wiki:2020-settlement-generation-competition#toc8 . [Accessed: 2021-10-10]

[35] "Settlement Generation Challenge 2020 - David Mason", *Gendesignmc.wikidot.com*, 2021. [Online]. Available: https://gendesignmc.wikidot.com/wiki:2020-settlement-generation-competition#toc1 . [Accessed: 2021-10-10]

[36] "MultiAgentFilter_AE_JF the generator created for this thesis", *https://github.com*. [Online]. Available: https://github.com/AlbinEsko/MultiAgentFilter_AE_JF. [Accessed: 2021-10-12]

Appendix A - Competition Criteria

Adaptability

- Do the structures in the settlement adapt to the terrain?
- Do the structure in the environment reflect the environment, i.e. usage of available material, adaptation to the biome?
- Does the settlement take advantage of terrain features or compensate for problems with the terrain?
- Are the settlements different in reaction to the different initial maps?
- Are there any other ways in how the settlement adapts to the given maps?

Functionality

- Does the settlement provide protection from danger?
- Does it keep mobs from spawning?
- Does it keep mobs out?
- Protection from other environmental dangers?
- Is the settlement accessible to a player avatar in survival mode?
- Can you walk to everywhere?
- Does the settlement provide faster modes of transport?
- How easy is it to find your way around?
- Does the settlement provide the player with additional affordances?
- Does the settlement makes ressources easy to obtain?
- Is there an easy way to get food?
- Does the settlement provide functionality to the villagers?
- Does the settlement reflect the embodiment of the player avatar?

- Is it appropriately scaled?

Narrative

- Is the settlement evoking an interesting story?
- After looking at the settlement, could you give a short description of what this settlement is about that sets it apart from other settlements?
- Is it clear what the function of the settlement is?
- Does this function make sense in regards to the terrain and environment it is in? I.e. is the logging camp in a forest, the harbour town at the sea?
- Is the functionality of the settlement supporting this narrative function? I.e. does the fortified frontier settlement have functioning walls, is the farming village equipped with functioning fields?
- Does the final settlement give any indication of how the settlement developed?
- Is it possible to look at the settlement and imagine in what order things where build, or what stages the development of the settlement took?
- Is there an indication of the history of the settlement evident in the structure?
- Are the any convincing and consistent allusion to human cultures or specific points in history that the settlement is modelled after?
- Does the settlement have a culture - either fictional or historical, that is evident from the settlement?
- Do you know things about this culture just by looking at the settlement?

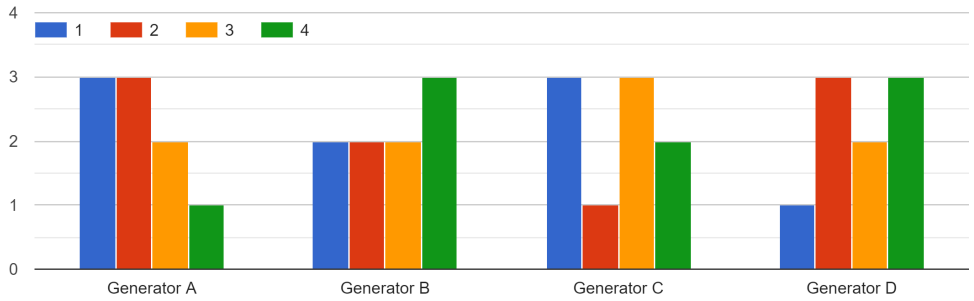
Aesthetics

- Does the settlement look good?

- Is there a consistent look to the settlement? Does it appear that all structures belong to the same settlement?
- Is there an appropriate level of variation in the existing structures?
- Are there any jarring features that make the settlement look unbelievable?

Appendix B - Raw Data from survey

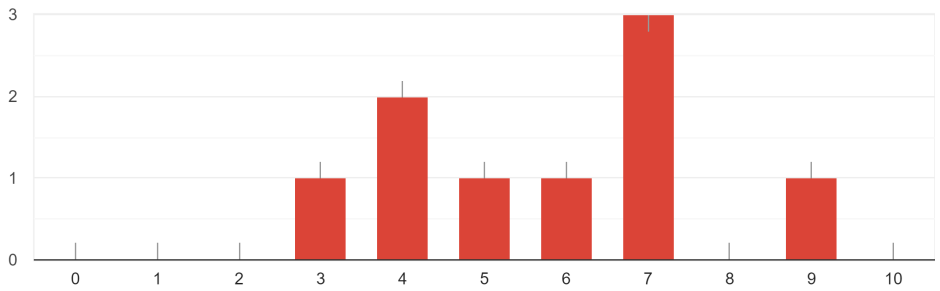
In which order did you explore the generators?



Generator A - Filip Skwarski 2019

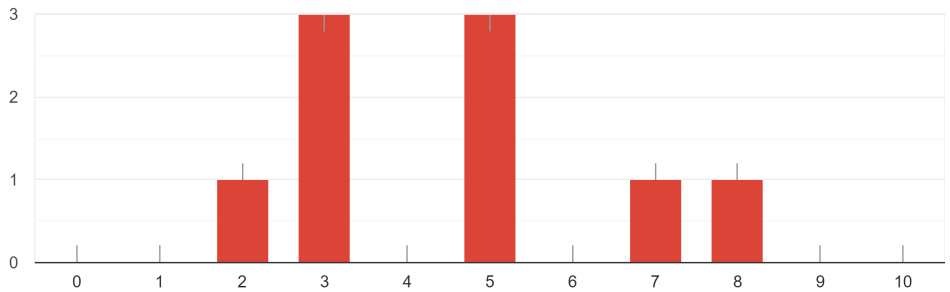
Generator A - Adaptability

9 svar



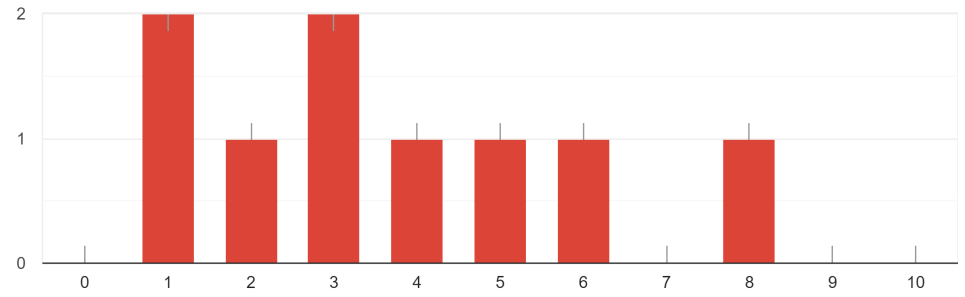
Generator A - Functionality

9 svar



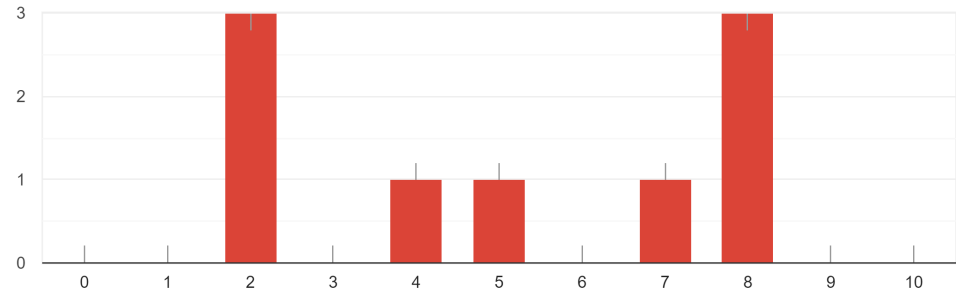
Generator A - Narrative

9 svar



Generator A - Aesthetics

9 svar



Comments:

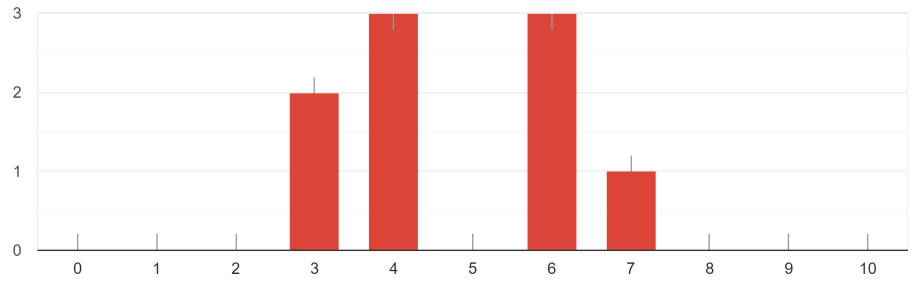
- Disconnected structures (Doors facing the wrong way, or simply unreachable), excessive amounts of bridges (often meaningless)
- The structures look good on their own and there is some variation in shape while retaining consistency, but there are also jarring features (like the aforementioned bridges and wierd roads)
- Too many paths; nice framing of landscape; no interiors
- It looks great! It's nice to walk around the village to explore.
- There was a big hole in front of one of the houses in Generator B Map A. I had to jump over the hole to get inside the house. Otherwise it looks great.

- Not to many houses but different types and you could see that some houses had different purposes like the churches for example.
- Probably the best looking settlement of the lot

Generator B - Troy, Ryan, & Trent

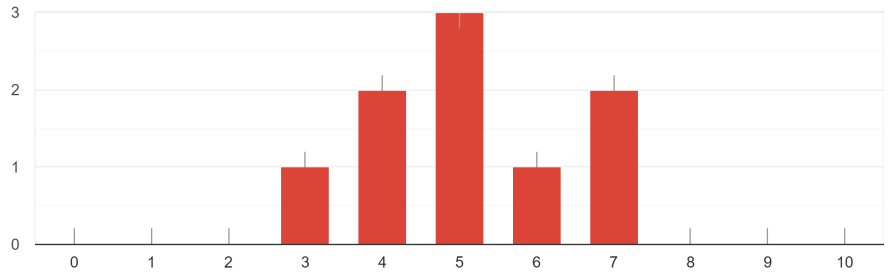
Generator B - Adaptability

9 svar



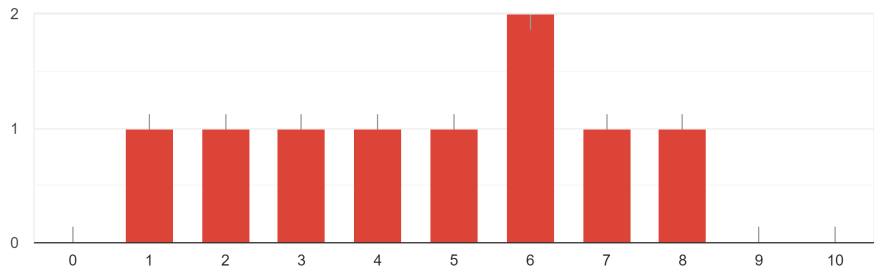
Generator B - Functionality

9 svar



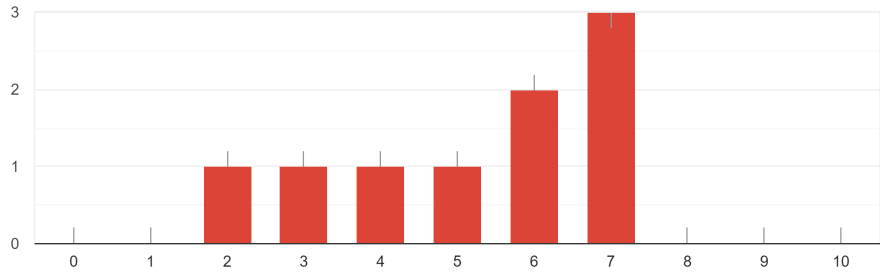
Generator B - Narrative

9 svar



Generator B - Aesthetics

9 svar



Comments:

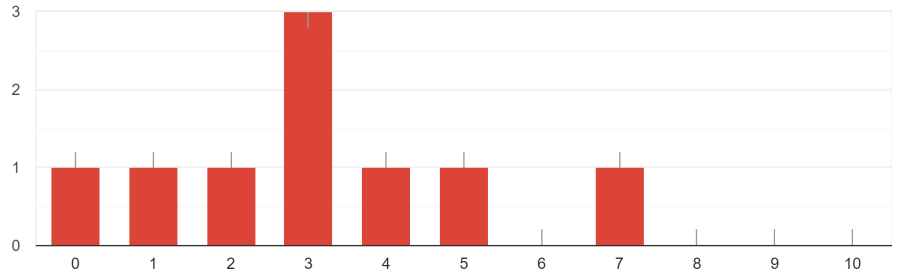
- The gates aren't connected to roads and there is no way onto the walls. the town could do with some extra roads (currently more or less everything lies on the same road, with no alternatvie roads giving the inhabitants "shortcuts" present) / briges (there are none at all) to allow for more efficient travel. Probably due to not generating bridges, some parts of the walled-in land becomes empty as there's no way to get there without passing over water. Rather than trying to compensate for problems with the terrain, the generator seemingly simply skipped anything that seemed problematic (Once again, this causes "empty" areas).
- I'd like to see some more variation in the homes, and some extra building types, like a horse stable / animal pen / fighting pit or some other structure you would expect to find in a hunter/military settlement.

- Accessibble doors, good looking, functional. The wood type adapted well to the environment.
- My fav generator.
- Very little difference between maps; enjoyed the random loot; water, food and gathering places
- Some of the houses are far apart from each other.
- Very medieval with a lot of protection for the village that gave the village more of a narrative. The houses also had beds that seems useful and it had incorporated a certain design and style for the houses.
- Like the aesthetics of the settlement, though in my opinion it fitted in better with the mountain environment.

Generator C - David Mason

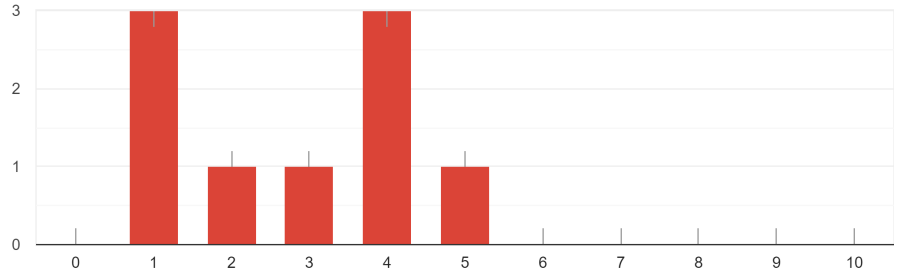
Generator C - Adaptability

9 svar



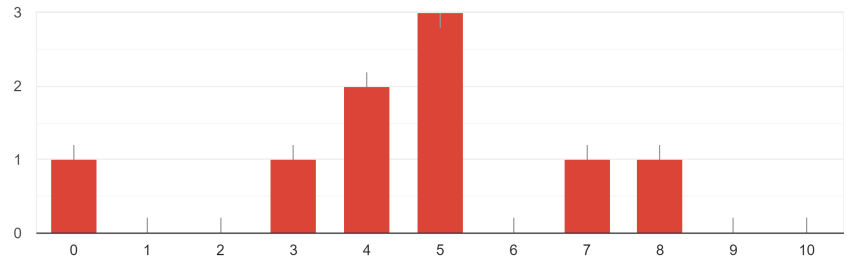
Generator C - Functionality

9 svar



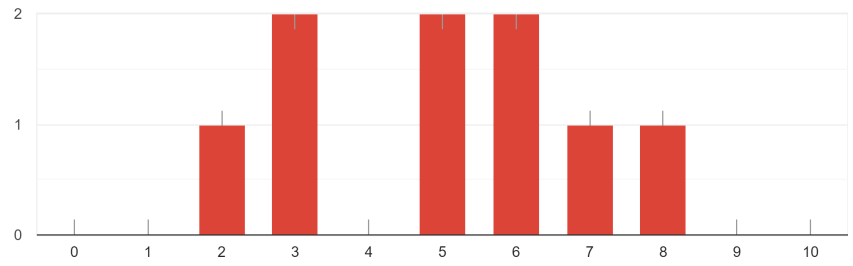
Generator C - Narrative

9 svar



Generator C - Aesthetics

9 svar



Comments:

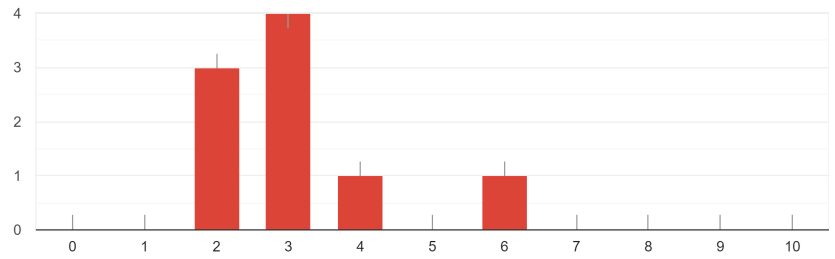
- unusable streets due to the incline being 2 blocks or way more at times. Buildings placed in fields, breaking them, there were also buildings place partly underground in caves or in water (unreachable door). Furnances blocking the stairs, "flying" torches and ladders. streets destroying the trunks of trees but leaving the rest flying. ...STONE roads over water?
- So, many jarring features, but the structures themselves and as a group looks ok with a high level of variation.
- The mining pit was a cool idea.
- Little consideration of the terrain beneath; organic and realistic usage of surface; pleasant shapes in paths and farmland

- On Map C a lot of the houses are messed up. Some houses are underground and others have floating doors. But there is a cool mine in the middle of the map.
- the mine big plus, the houses under ground where somewhat difficult to access. second map (mountain was the coolest)
- Definitely a mining settlement

Generator D - Artefact

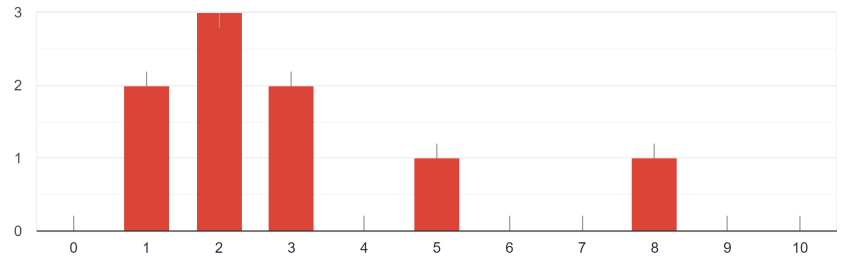
Generator D - Adaptability

9 svar



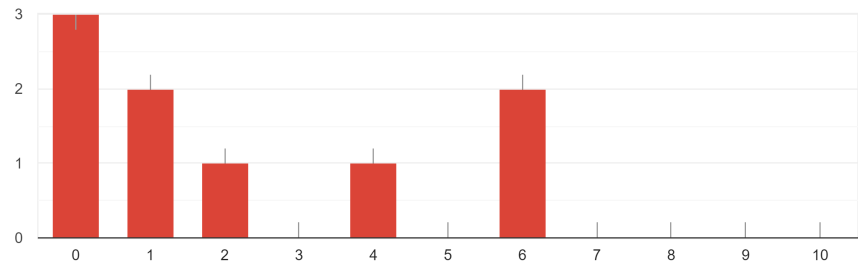
Generator D - Functionality

9 svar



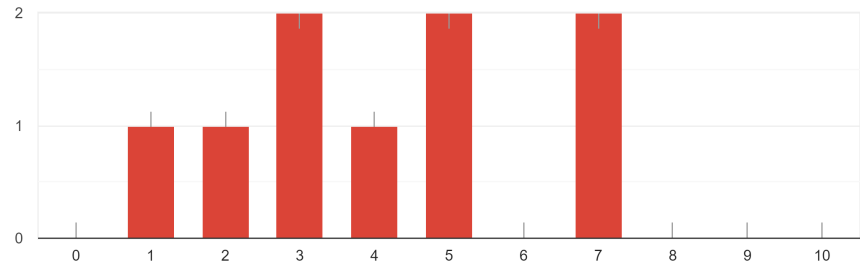
Generator D - Narrative

9 svar



Generator D - Aesthetics

9 svar



Comments:

- Although the scores are low i really liked the layout overall for this generator on map A. There was a good balance of house-sizes and roads. Would be great for a renovation project.
- No torches, even in houses (no safety)? houses are very boring/similar. no bridges at all. No food at all. oak houses in a spruce forest.
- High accessibility though!
- Chaotic paths; no infrastructure (aside paths); very generic construction
- On Generator A Map D there are a lot of floating trees close to some of the houses. And some trees are blocking the paths.
- more generic houses, but very clean. This generator did not do anything to unexpected but in a way that made the village more believable. This generator was very clean but

left you with a feeling that the village was under construction, while the first generator made a mine that made more of a purpose for the village it was creating.

- This was a neat settlement. Reminded me of pokemon villages for some reason.

Appendix C - Data from GDMC Competition 2021

Entry	Name	Rank	Adaptability	Functionality	Narrative	Aesthetics	Final score
10	MAU_AE_JF	17	3.647	3.059	2.412	3.176	3.073

Excerpt of the GDMC 2021 competition rankings for this generator

Entry	Team Name	Rank	Adaptability score	Functionality score	Narrative score	Aesthetics score	Final score
15	Tsukuba Team	1	4,684829721	5,717956656	5,602786378	5,373374613	5,344736842
7	aith (solo-team)	2	4,292387543	4,51384083	5,268512111	5,55017301	4,906228374
6	Nils Gawlik	3	3,508668731	4,718266254	4,989164087	6,371826625	4,896981424
3	William, Selina, Ho Kiu, Rhys	4	3,911764706	5,588235294	4,294117647	5,705882353	4,875
16	CharlieMDRz	5	4,929411765	4,429411765	4,617647059	4,688235294	4,666176471
5	Cristopher Yates	6	3,794117647	4,235294118	4,941176471	5,323529412	4,573529412
2	Porter Hickey	7	3,558823529	4,388235294	4,029411765	6,282352941	4,564705882
14	Beautiful_Ka	8	2,823529412	4,705882353	4	5,352941176	4,220588235

	askoders						
20	ICE_JIT_ft_WFC	9	3,641795666	3,681114551	4,312693498	5,116099071	4,187925697
9	LIGC	10	4,9	3,958823529	3,352941176	3,988235294	4,05
1	Brianna & Kyle	11	3,647058824	4,035294118	3,058823529	4,917647059	3,914705882
8	The World Foundry	12	1,660808596	3,224676744	4,447769077	4,30651976	3,409943544
11	Lisa	13	2,910034602	3,366782007	3,456747405	3,373702422	3,276816609
12	Rexos and Xeono X	14	2,647058824	2,764705882	3,117647059	4,376470588	3,226470588
13	Rainbow Parliament	15	1,058823529	2,647058824	3,470588235	5,323529412	3,125
18	Terje Schjelderup	16	4,320433437	2,498452012	2,634674923	2,989164087	3,110681115
10	**MAU_AE_JF**	**17**	**3,647058824**	**3,058823529**	**2,411764706**	**3,176470588**	**3,073529412**
4	Keywarn	18	4,070588235	2,294117647	2	2,970588235	2,833823529
19	Hunternif	19	3,176470588	1,176470588	2,705882353	4,094117647	2,788235294
17	Leiden University MGAI team 1	20	1,470588235	2,205882353	3,058823529	4,352941176	2,772058824

The GDMC 2021 competition rankings.

Comments:

- Really nice job on making the road follow the river, I like that. I also appreciate that you tried to generate random furniture, you could have done more with that. The stone basin on the houses is good too! The generator needs a bit more work with variation, it looks very similar. Maybe less houses, or some sort of text to indicate what the settlement is about.
- For detailed information regarding your scoring, please contact [redacted]
- Pathfinding seems quite clever, following shorelines, avoiding cliffs, and avoiding the walled city.
- Procedural houses, placed around road network. Light furnishing (bookshelves). Some in lava falls. Intersection through hills
- Basic bulldings with some interiors, could do with a bit more variety in exterior design. Quite good pathing, I liked the ending layout!
- To be honest this generator is rather plain- the buildings need more variation in appearance and function, as well as size. There is a good start to furnishing, though it needs more work. The roads and pathfinding are well done, though I'd like to see them have some lights.
- Lots of wood houses in desert. Brick walkways tend to deal with inclines awkwardly. Consistent Walkways. Decent diversity in building style. Doesn't look too unlike settlements generated by vanilla minecraft. Useful buildings. Wood houses are very plain
- The settlement doesn't quite respect existing structures on the Hybrid map, but doesn't destroy them either and it seems to split the settlement into different parts, which adds to the story it tells. Buildings are quite similar, nicely connected with roads and with some detail inside.
- Buildings and roads are terrain-adapted, but only few types of buildings with no facilities.
- These are really nice buildings in this generation, and their layout and the combination between the layout and the roads works really well. They are nicely clustered while also giving a sense of the organic growth of this settlement, especially around the river. The houses are nicely spread out but do not attempt to climb up the volcano / mountain walls, which is a really strong part of this generator. The road around the river is okay, but perhaps the weakest part of this at the macro scale since it's very obvious how this part works; although I see the logic in using the river as a kind of "anchor" for the rest of the settlement, I think this could be a little more varied. The houses themselves are sufficiently similar to evoke a coherent aesthetic while also being sufficiently different to not get boring; the door and window locations and numbers vary nicely, and the interiors have some good stuff in them, though these sorts of buildings

are really just crying out for more interior variation and also a second floor! One of the houses really "carved out" its space from the mountain, and in the context it looks "ok" though I think that element of the generator would create more problematic outcomes on other sorts of maps. One was also close to the lava and promptly set on fire! I'm not a huge fan of the two houses that are really "buried" inside the mountain though, the walls of the mountain are far too smooth - but aside from these issues, I really like this generator, especially its macro-scale shape and the roads. Good stuff!

- It’s cool that the path follows the river, the roads also look nice.
- What's good: Mostly adapts well to terrain, in particular roads. Decent looking buildings. Areas to improve: More variation, although there is clearly procedural generation of different shapes of buildings, all buildings look very similar both in form and color. More interior. Indication of a larger narrative than just a collection of homes.