Minecraft Settlement Generation AI - Project Plan

Blake Patterson, Michael Ward

February 26, 2022

Contents

1 Abstract

Minecraft is a sandbox video game, created by Mojang in 2009, where players explore and build in a procedurally generated 3D grid-like world with infinite terrain. The main game-play element of Minecraft consists of collecting various types of materials and using them to build tools and structures. The world is divided into 1x1x1 blocks which can vary in material, spawning location, and usability. Aside from the popularity of the base game, Minecraft has become well known for it's customization possibilities through a variety of open source application program interfaces. These application program interfaces allow players to change textures and color palettes, add new items, block types and enemies, and more.

The open source nature and in-game environment of Minecraft has also caught the attention of artificial intelligence researchers. The environment of Minecraft is ideal for research in AI because of the endless possibilities, from training an agent on simple tasks like searching for a specific object or material, to building complex structures or navigating obstacle courses. Since the environment is divided into a three-dimensional grid of equal sized cubes, it is also easy to measure and evaluate the performance of AI in Minecraft.

This project is focused on the application of AI for Procedural Content Generation (PCG) within Minecraft. PCG is defined as the algorithmic creation of game content with limited or indirect user input [7]. Content in the context of PCG can be described as most of what can be contained within a game including maps, rules, textures, items, quests, music, characters, and more [7]. Many popular games have made use of PCG including Rogue, Dwarf Fortress, Diablo, Spore, Civilization, Spelunky, as well as Minecraft [7]. The usage of PCG varies from game to game and can range from fully autonomous game design, to automating routine or common aspects of game design. One major critique of PCG in game design has to do with repetition and functionality; rule-based agents are likely to create good looking and functional content that looks similar, and search-based agents are likely to create more diverse content, but takes more time and resources to ensure that it is functional for the player [2].

Most instances of PCG in video games operate from a 'clean-state' where the generator does not have to consider interaction with preexisting in-game elements [2]. Exploring PCG within Minecraft opens up a new challenge within AI, in which the goal is to produce a functional and believable village settlement that adapts to different environments within a Minecraft map [5]. Instead of generating a village on a clean slate, this problem restricts the generator with the presence of preexisting game elements and focuses on adaptive generation of artifacts [2]. A map in Minecraft is made up of various biomes which contain different types of terrain, elevation gradients, fauna, and bodies of water. In order for a procedurally generated settlement to be functional and believable, it must be adaptive and able to build on top of and in response to elements that already exist in the Minecraft environment. The Generative Design in Minecraft (GDMC) AI settlement generation competition initially proposed this problem in 2018 [4]. GMDC has ran a yearly

open competition for researchers and students to submit their algorithm, which is scored by a panel in terms of the algorithms adaptability and functionality [1].

We propose to develop a Procedural Content Generation AI that is capable of generating a functional and believable Minecraft settlement, which is adaptive to varying environmental factors. Based on our first review of literature, it is apparent that developing multiple different algorithms to handle individual pieces of the problem has led to better outcomes in previous research. For example, iterations of the A* algorithm have been successful in creating road networks between houses within the settlements. For other parts of the problem such as terrain analysis and building generation, different approaches will need to be employed that require more research. In the tasks and timetable sections, we lay out our current expectations for what steps will be needed and the order of steps.

2 Tasks Planned

Assigned To	Task
Blake, Michael	Set up development environment
Blake, Michael	Test out HTML interface framework and learn how to use it
Blake, Michael	Conduct further research of possible algorithms/approaches
Blake, Michael	Finalize details of decided algorithm/approach
Michael	Code outline for Python script
Blake	Design settlement type(s) and implement the details in Python code
Michael	Implement terrain analyzer
Blake	Implement house generation
Michael	Implement farm generation
Blake	Implement field generation
Michael	Implement bridge generation
Blake	Implement food production generation
Michael	Implement tunnel generation
Blake	Implement road generation
Blake, Michael	Create project presentation

3 Estimated Timetable

	Tasks
Week 1	Set up development environment, Test out HTML interface framework and learn how to use it, Conduct further research of possible algorithms/approaches
Week 2	Finalize details of decided algorithm/approach, Code outline for Python script, Design settlement type(s) and implement the details in Python code
Week 3	Implement terrain analyzer, Implement house generation
Week 4	Implement farm generation, Implement field generation
Week 5	Implement bridge generation, Implement food production generation
Week 6	Implement tunnel generation, Implement road generation, Begin working on project presentation
Week 7	Finishing touches and final bug fixes on settlement generator, Finalize project presentation

4 References

[1] Marcus Fridh and Fredrik Sy. "Settlement Generation in Minecraft". In: (), p. 55.

[2] Michael Cerny Green, Christoph Salge, and Julian Togelius. "Organic Building Generation in Minecraft". In: *arXiv:1906.05094 [cs]* (June 11, 2019). arXiv: 1906.05094. URL: http://arxiv.org/abs/1906.05094 (visited on 02/24/2022).

[3] Jean-Baptiste Hervé and Christoph Salge. "Comparing PCG metrics with Human Evaluation in Minecraft Settlement Generation". In: *arXiv:2107.02457 [cs]* (July 6, 2021). arXiv: 2107.02457. URL: http://arxiv.org/abs/2107.02457 (visited on 02/24/2022).

[4] Christoph Salge et al. "Generative Design in Minecraft (GDMC), Settlement Generation Competition". In: *Proceedings of the 13th International Conference on the Foundations of Digital Games* (Aug. 7, 2018), pp. 1–10. DOI: 10.1145/3235765.3235814. arXiv: 1803.09853. URL: http://arxiv.org/abs/1803.09853 (visited on 02/24/2022).

[5] Christoph Salge et al. "Generative Design in Minecraft: Chronicle Challenge". In: *arXiv:1905.05888 [cs]* (May 14, 2019). arXiv: 1905.05888. URL: http://arxiv.org/abs/1905.05888 (visited on 02/24/2022).

[6] Christoph Salge et al. "The AI Settlement Generation Challenge in Minecraft: First Year Report". In: *KI - Künstliche Intelligenz* 34.1 (Mar. 2020), pp. 19–31. ISSN: 0933-1875, 1610-1987. DOI: 10.1007/s13218-020-00635-0. URL: http://link.springer.com/10.1007/s13218-020-00635-0 (visited on 02/24/2022).

[7] Noor Shaker, Julian Togelius, and Mark J Nelson. *Procedural content generation in games*. Springer, 2016.