Metropolist

A Manuscript
Submitted to
the Department of Computer Science
and the Faculty of the
University of Wisconsin–La Crosse
La Crosse, Wisconsin

by

Shizheng Yang

in Partial Fulfillment of the
Requirements for the Degree of

Master of Software Engineering

May, 2019

Metropolist

By Shizheng Yang

We recommend acceptance of this manuscript in partial fulfillment of this candidate's requirements for the degree of Master of Software Engineering in Computer Science. The candidate has completed the oral examination requirement of the capstone project for the degree.

Prof. Kasi Periyasamy Examination Committee Chairperson	Date
Prof. Steven Senger Examination Committee Member	Date
Prof. Kenny Hunt Examination Committee Member	Date

Abstract

Yang, Shizheng, “Metropolist,” Master of Software Engineering, May 2019, (Kenny Hunt, Ph.D.).

This manuscript describes the development of a web-based map generator, which allows user to create randomly generated fantasy maps of cities and additionally allows user to annotate and edit city element at a fine granularity.

Acknowledgements

I would like to express my sincere appreciation to my project advisor Dr. Kenny Hunt for his invaluable guidance and untiring support. I would also like to express my thanks to the Department of Computer Science at the University of Wisconsin–La Crosse for providing the learning materials and computing environment for my project.

Table of Contents

List of Tables

List of Figures

Glossary

LaTeX

LaTeX is a document markup language and document preparation systems for the TeX typesetting program.

1. Introduction

1.1. Background

Generating content for large computer games or CGI effects in feature films is a significant bottleneck in terms of effort and resources. A typical game contains many thousands of audio files, images, textures and 3D models. Procedurally generated content provides a cost-effective alternative to the manual creation of models, textures, images and sound assets and can expand playability beyond what is otherwise possible. The video game "No Man's Sky", for example, was released in 2016 and relies on procedural asset generation to create over 18 quintillion planets each of which has a unique ecosystem composed of flora and fauna. Figure 1 is a screenshot of "No Man's Sky". Such a scale is, of course, beyond the reach of any manually created system.

Figure 1. A screenshot of "No Man's Sky"

Well-known techniques such as fractals, L-systems, Perlin noise, and others have been used to procedurally generate plants, terrain, and cityscapes. At this point, we have to introduce a well-known game: "Minecraft", which can produce massive worlds that are chock-full of little details, like elaborate cliff faces and waterfalls. And it relies on procedural generation, which automatically creates environments and objects that are at once random, but guided by rules that maintain a consistent logic. Mountains are always rocky and sprinkled with snow, for example, while the low lands are typically full of grass and trees. Figure 2 is a screenshot of "Minecraft".

1.2. Similar Systems

The "Medieval Fantasy City Generator(MFCG)" is a web application. This application generates a random medieval city layout of a requested size: small,

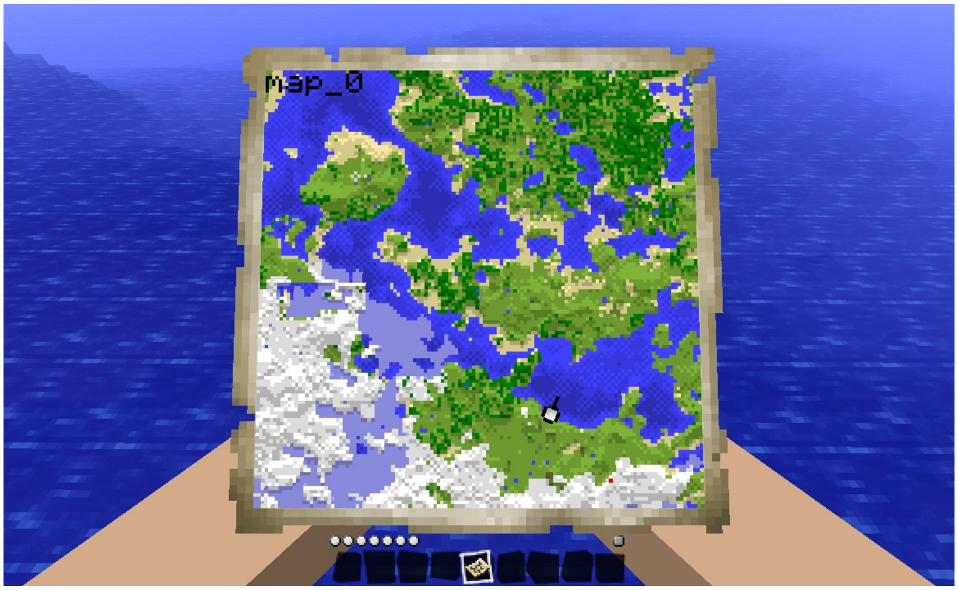


Figure 2. A screenshot of ”Minecraft”

medium or large, which is made up of different types of regions, and the generation method is rather arbitrary. Besides, many elements are provided for user to add to the city, such as farm fields, citadel, plaza, temple, river, coast and so on. Because of the premise of medieval fantasy, the map always includes the walls and castle, but the user can decide whether to display them. What’s more, it allows the user to edit the map in order to modify some unsatisfying places using warp tool. The author also mentioned that the goal of the application is to produce a nice looking map, not an accurate model of a city. Finally, the user can save the map as an image in ”png” or ”svg” format by using the export feature if he is satisfied with the map. Figure 3 is a screenshot of MFCG.

The ”Azgaar’s Fantasy Map Generator(FMG)” is another similar system, which is a large scale system. The size of the map made by FMG varies from western Europe to the world continent, not just an island, but a random fantasy map represents pseudomedieval world. Just like the real world, the map generated by FMG has the feature that the continent is always surrounded by the ocean and will never touch the border of the map. Although it is a random map, it is still based on real-world rules. While its biggest feature is that the user can choose the type of map he likes, which provides the following 5 types: political map, cultural map, height map, biomes map and pure landmass. In addition, it also supports user-defined map type, the user can add additional layers on existing map layer, such as: rivers layer, temperature layer, and population layer. What’s more, it also allows the user to annotate and edit the map using layout editor, style editor, template editor, scale editor, countries editor, cultural editor or namebase editor. Last but not least, it supports exporting the map in ”png” or ”svg” format, but unlike the previous

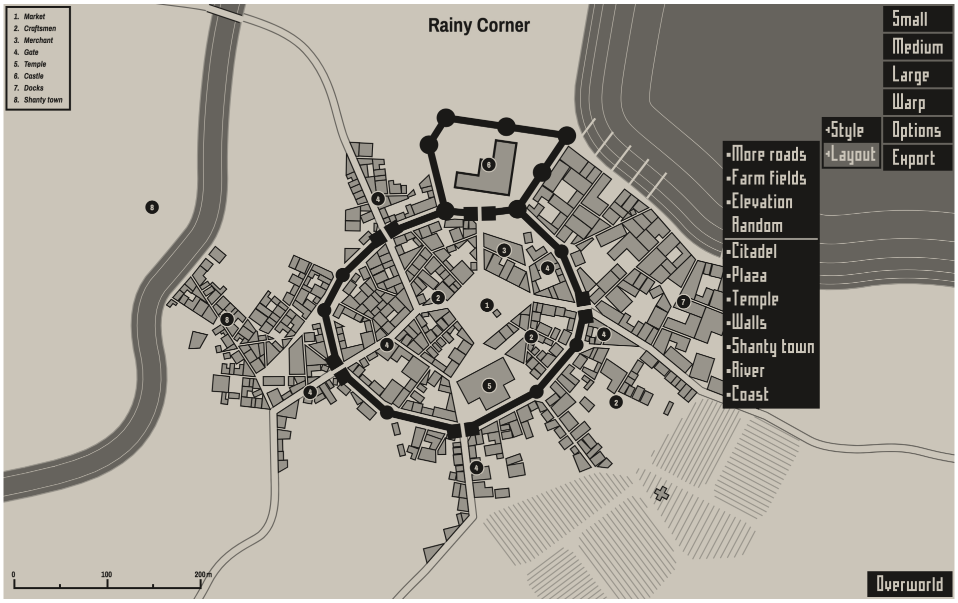
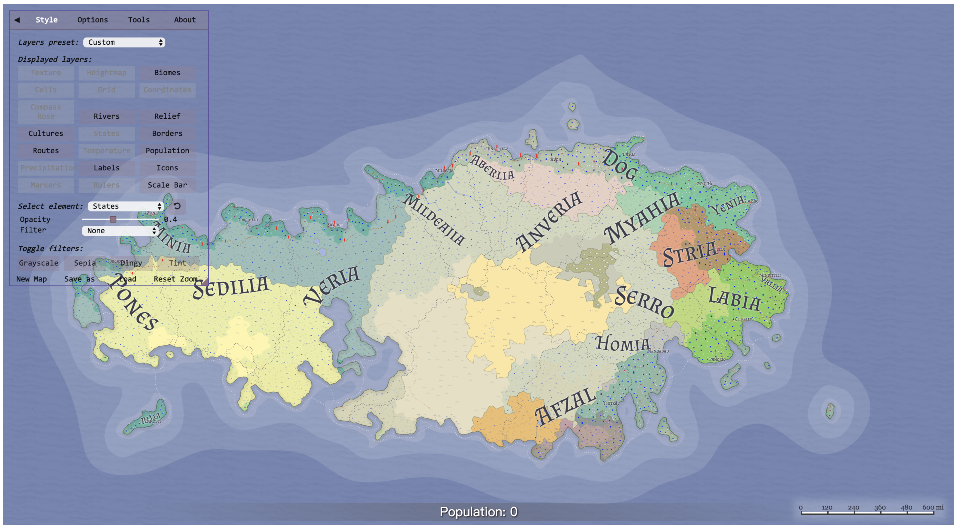


Figure 3. A screenshot of Medieval Fantasy City Generator

application, if the user wants to come back to edit the map in the future, he can save it in the ”map” format. Figure 4 is a screenshot of FMG.


Layers preset: Custom

Select element: States

Opacity 0.4

Filter None

Population: 0

Figure 4. A screenshot of Azgaar’s Fantasy Map Generator

1.3. Project Goal

This project seeks to develop a software system that allows the user to generate maps of medieval cities for use in fantasy games or for purely artistic purposes. While the ultimate goal of this project is to procedurally replicate maps of a quality similar to the best cartographic hand-created maps by expert artists, we expect to obtain a modest approximation to the desired level of quality. Besides, our system will have the basic features such as: procedurally and randomly generated, allowing the user to annotate or edit, supporting the user to export map as an image in ”png” or ”svg” format, allowing the user to save it to the database and restore it from the server. In order to achieve these goals, we will implement the following operations and rules:

- When the user open the map editor, it shows user an empty map with random map seeds by default, then the user can starting making map by accessing the menu on the right
- In the menu, the user can enter the number of map seeds to create a new empty map, or remain unchanged
- The user can choose the type of layout for the map: elevation, affluence, desirability, district and building, which can be superimposed on each other
- The user can choose to display street names or not
- The user can manipulate the ”increase/decrease toggle”, ”increment sliding” and ”waterline sliding” to edit the map: increasing or descreasing the value of elevation and affluence; building or removing the wall; making map cell as water or city
- If the waterline is changed, the continent will be changed accordingly.
- The user can choose to display contour lines or not
- After selecting one or more than one layers under ”edit” mode, the user can change the size of the ”soft brush” by using the mouse wheel, which determines the area where the map will be edited
- The user edits the map by clicking and dragging the ”soft brush”
- The user can zoom in and zoom out the map by pressing on the alt key and using the mouse wheel, of course, the map can be resized by a specialised button
- The user can save map or download it by clicking the button on the right bottom

2. Requirements

2.1. Overview

This gives a brief overview of this section.

2.2. Point 1

This subsection gives a great deal of precise description supporting point 1. For example, Figure 5 explains in great detail a state chart.

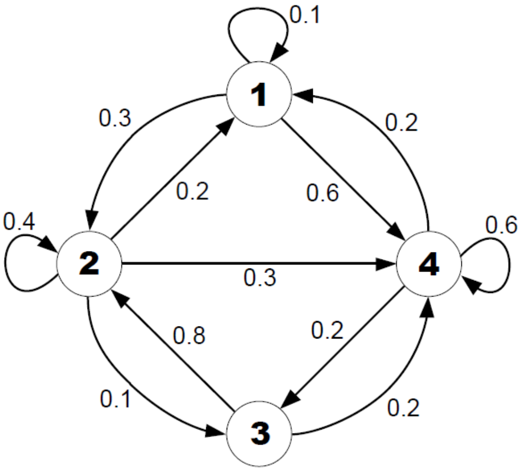


Figure 5. State Chart Diagram

2.3. Point 2

This gives Point 2

3. Design and Implementation

3.1. Overview

This gives a brief overview of this section.

3.2. Point 1

This subsection gives a great deal of precise description supporting point 1. For example, Figure 6 explains in great detail a state chart.

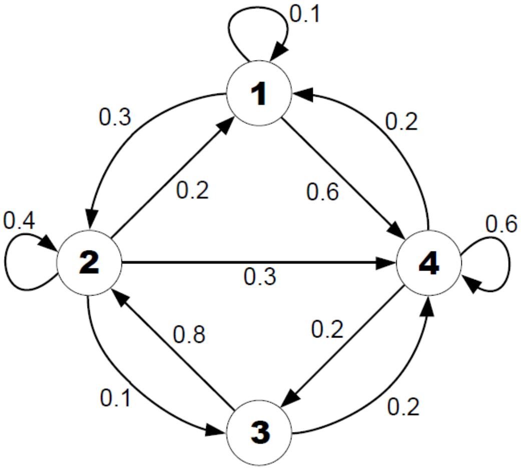


Figure 6. State Chart Diagram

3.3. Point 2

This gives Point 2

4. Bibliography

5. Appendices