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 computer games, CGI effects, feature films, print media, or Role-playing games 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 relied on procedural asset generation to create over 18 quintillion planets each of which has a unique ecosystem composed of flora and fauna. Such a scale is, of course, beyond the reach of any manually created system. Figure 1 is a screenshot of a planetary terrain region of "No Man's Sky."

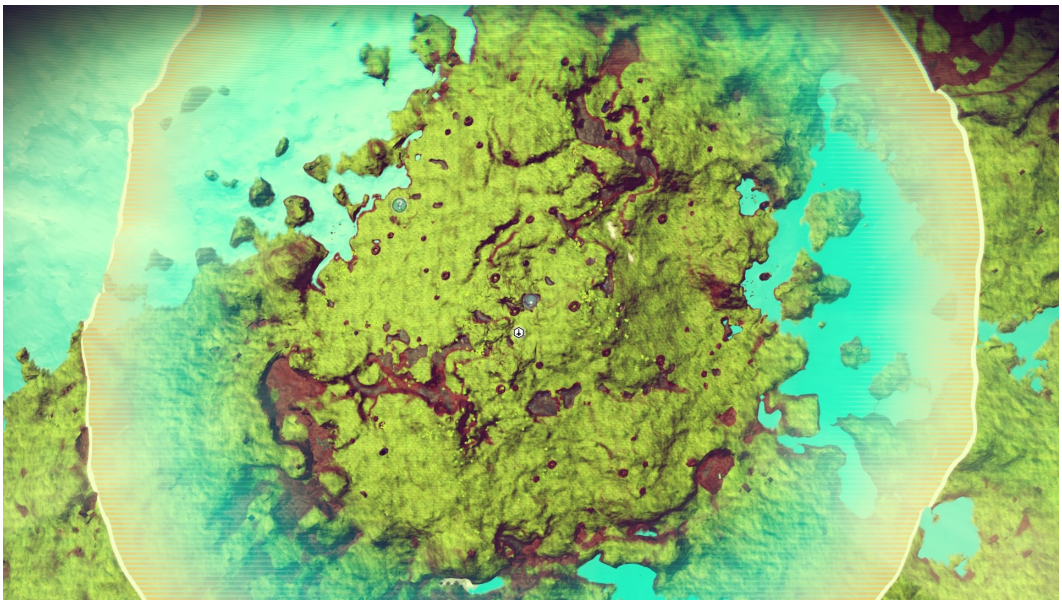

Figure 1. A screenshot of a planetary terrain region of "No Man's Sky"

1.2. Similar Systems

1.2.1. Minecraft

"Minecraft" is a computer game that can produce massive worlds that are chock-full of little details, like elaborate cliff faces and waterfalls. Moreover, 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."

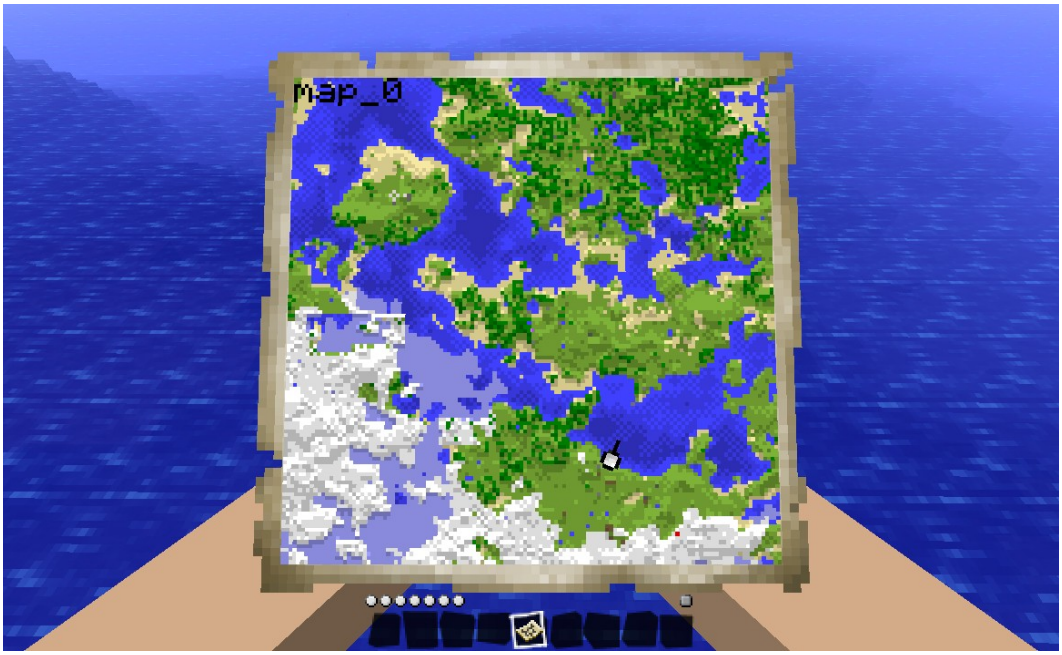

Figure 2. A screenshot of "Minecraft"

1.2.2. Medieval Fantasy City Generator(MFCG)

The "Medieval Fantasy City Generator(MFCG)" is a web application. This application generates a random medieval city layout of a requested size: small, medium or large, which is made up of different types of regions, and the generation method is rather arbitrary. Furthermore, many elements are provided for the 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. It allows the user to edit the map to modify some unsatisfying places using the 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 the "png" or "svg" format by using the export feature if he is satisfied with the map. Figure 3 is a screenshot of MFCG.

1.2.3. Azgaar's Fantasy Map Generator(FMG)

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 a pseudo-medieval 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 most prominent feature is that the user can choose the type of map he likes, which provides the

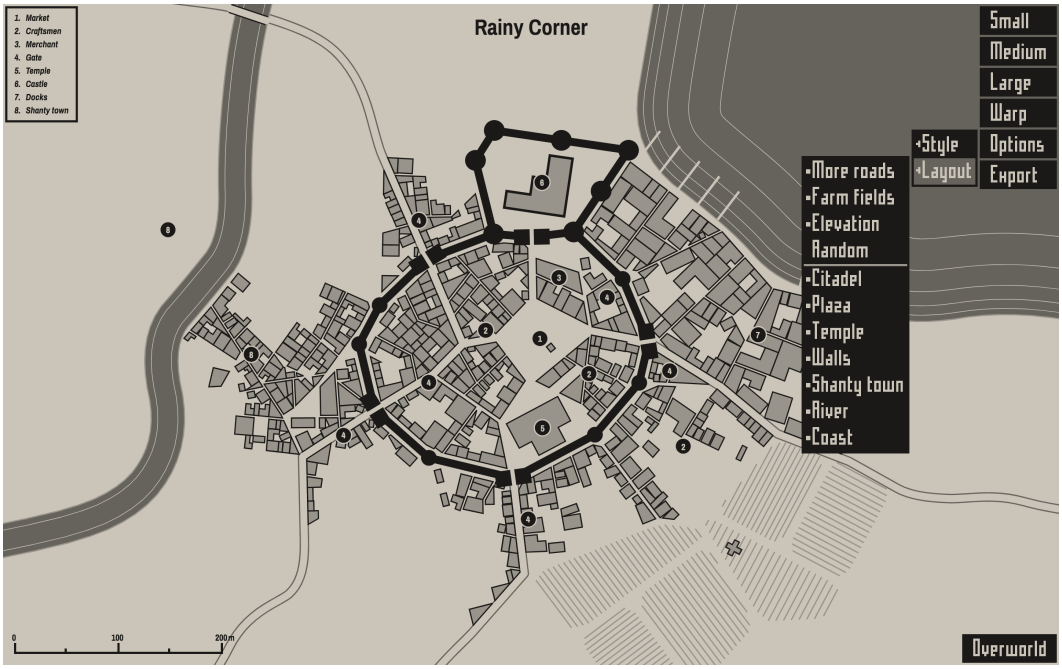

Figure 3. A screenshot of the Medieval Fantasy City Generator

following 5 map types: political, cultural, height, biomes and pure landmass. It also supports user-defined map type, and the user can add additional layers on existing map layers: rivers, temperature, and population. Also, it allows the user to annotate and edit the map using various such editors: layout, style, template, scale, countries, or cultural. It also supports exporting the map in the "png" or "svg" format, but unlike the previous 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.

1.3. Project Goal

The goal of this project is to automatically generate city maps for use in Role-playing games or world building narratives. 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. Our system will have the essential features, such as: allowing the user to annotate or edit, supporting the user to export the map as an image in the "png" or "svg" format, allowing the user to save it to the database and restore it from the server. To achieve these goals, we provide the following operations and obey these rules:

- The user should use the given tool to draw the map
- The user manually creates water, lands, and walls

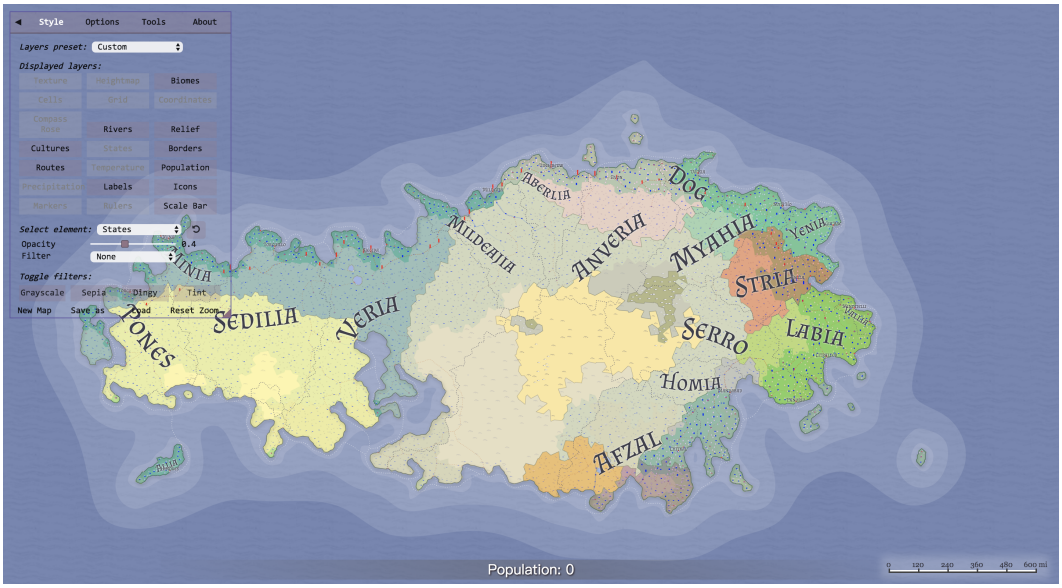

Figure 4. A screenshot of the Azgaar's Fantasy Map Generator

- The map procedurally generates districts, buildings, streets, and street names

2. Requirements

2.1. Functional Requirements

Initially, almost all the requirements were given by Dr. Kenny Hunt, in an introductory meeting. There were daily meetings after that, in which Dr. Kenny Hunt would add additional requirements each time. He would also tune up any previous requirements that were given. The functional requirements focus on developing a web application. There are two roles for this system: admin and user. As a result, the following functional requirements were established for the project:

- Both admin and user can:
  - Log In: with unique email and password
  - Log Out: end the session
  - Edit Personal Profile: delete the account or change email, first name, last name, and password
- Admin is able to:
  - View All Users: on the "dashboard" page
  - Enable or Disable Users: allow the user to log in or not
  - Search for Users: use any combination of email, first name, and last name; on the "dashboard" page
- User is able to:
  - Sign Up: with unique email, first name, last name, long password, and confirmed password
  - View All Own Maps: on the "dashboard" page
  - View All Visible Maps: on the "community" page
  - Search for Maps: use any combination of the map name, created date, edited date, the owner's email, the owner's first name, and the owner's last name; on the "community" page
  - Download Maps: of either own or other visible; in "png" or "svg" format
  - Make Maps Public or Private: allow the map displaying on the "community" page or not
  - Create Maps: with a name; on the "dashboard" page
  - Delete Maps: on the "dashboard" page
  - Edit Maps:

* When the user opens the map editor page, it shows an empty map with random map seeds by default, and then the user can make maps by accessing the menu on the right
* In the menu, the user can input the number of map seeds to create a new empty map, or remain unchanged
* The user can select different types of layers for the map: elevation, affluence, desirability, district, and building, which can superimpose 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 decreasing 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 districts and buildings are procedurally generated
* The user is allowed to change the type of the current district by right-clicking and selecting a new type form the context menu
* The map provides 13 different types of districts: rich, medium, poor, empty, plaza, park, farm, water, harbor, university, religious, castle and military
* The user can zoom in and zoom out the map by pressing on the alt key and using the mouse wheel
* The user can use a specialized button to resize the map on the right top
* The user can save the map or download it by clicking the button on the right bottom

Figure 5 is the use case diagram that describes the functionalities to be implemented by this web application.

2.2. Non-Functional Requirements

There are too many non-functional requirements of a system: response time, availability, usability, security (authentication, authorization, integrity, privacy, etc.) and so on. Preventing writing a wish list, we decided to focus on the following requirements:

- Security:
  - Input Validation:
    - Add validations at both client and server sides
    - All methods should always validate all parameters in the server-side
  - Password Encryption:
    - Shall always be encrypted
    - Not even the system administrators shall see clear text passwords
    - Applying a hashing algorithm to passwords
  - Prohibiting Cross Site Scripting (XSS):
    - Never insert data anywhere in a ¡script¿ element
    - Never insert data in an HTML comment
    - Never insert data in an attribute name
    - Never insert data in an attribute value
    - Never insert data in a tag name
    - Never insert data anywhere in CSS
  - Secure Session Cookie:
    - Setting the "Secure" cookie attribute and instruct web browsers to send the cookie only over encrypted, e.g., HTTPS, links.
    - Setting the "HttpOnly" attribute true and prohibiting the JavaScript code from reading the cookie via the DOM "document.cookie" JavaScript object.
    - Always change the session id after login
- Performance:
  - High performance of rendering map:
    - Comparing and selecting the fastest way to render maps, e.g., canvas, SVG

2.3. Selection of Life Cycle Model

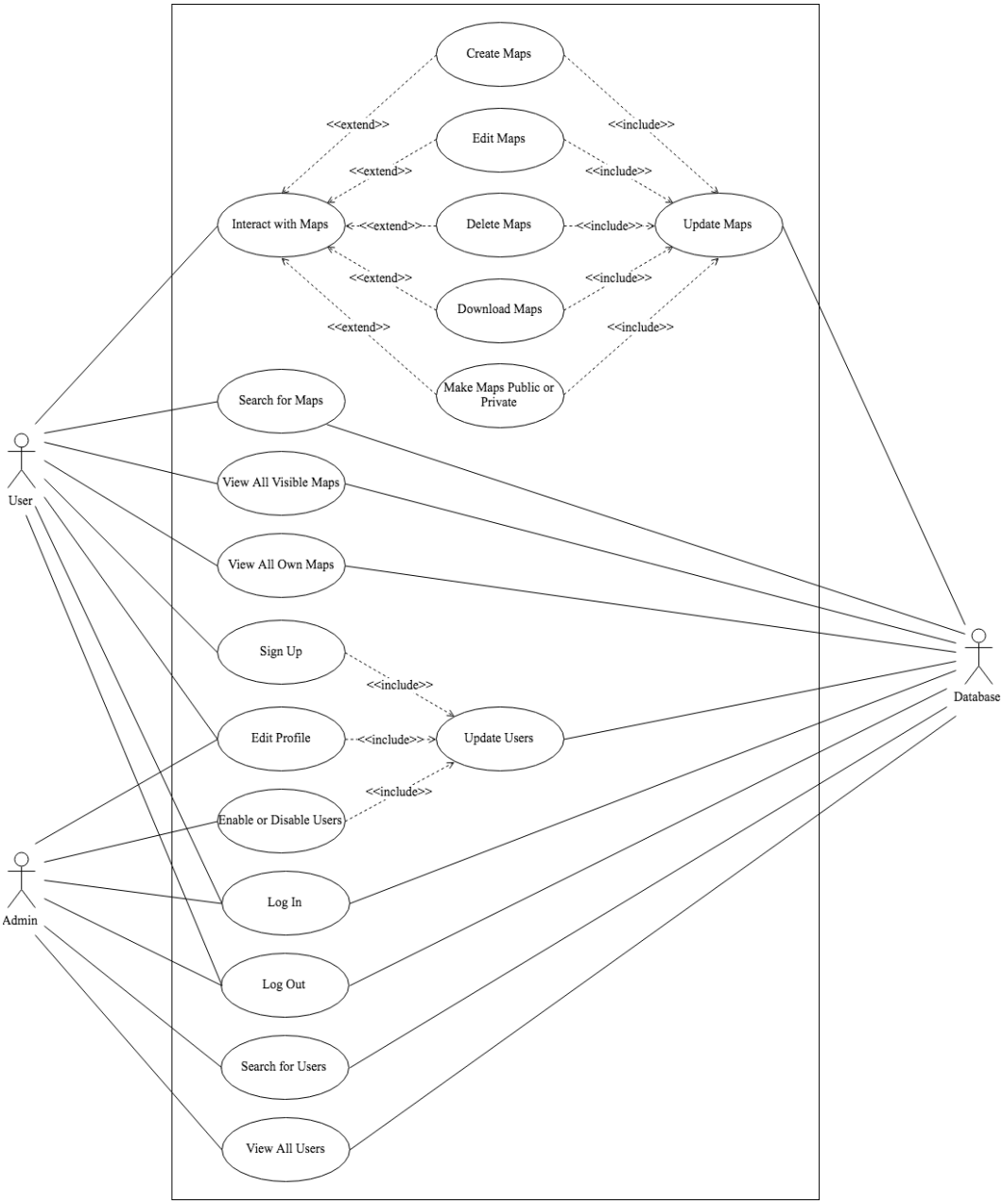

Figure 5. Use Case Diagram

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.

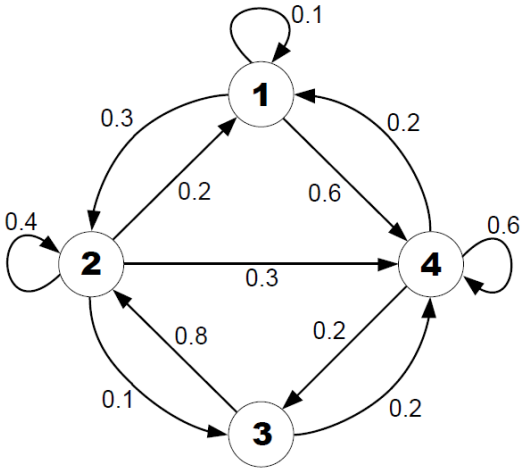

Figure 6. State Chart Diagram

3.3. Point 2

This gives Point 2

4. Bibliography

5. Appendices