

CS 320 Course Project Final Report

for

Medival Maps

Prepared by

Rabid Cabbage Collective

|  |  |  |
| --- | --- | --- |
| Patrick Wellington | <student #> | [Patrick.wellington@wsu.edu](mailto:Patrick.wellington@wsu.edu) |
| Lucas Gubala | 11543921 | [Lucas.gubala@wsu.edu](mailto:Lucas.gubala@wsu.edu) |
| Kevin Ngo | 11518172 | [Kevin.ngo2@wsu.edu](mailto:Kevin.ngo2@wsu.edu) |
|  |  |  |

|  |  |
| --- | --- |
|  |  |
| Date: | 12/13/2018 |
|  |  |
|  |  |
|  |  |

Contents

Contents ii

1 Introduction 1

1.1 Project Overview 1

1.2 Definitions, Acronyms and Abbreviations 1

1.3 References and Acknowledgments 1

2 Design 2

2.1 System Modeling 2

2.2 Interface Design 2

3 Implementation 3

3.1 Development Environment 3

3.2 Task Distribution 3

3.3 Challenges 3

4 Testing 4

4.1 Testing Plan 4

4.2 Tests for Functional Requirements 4

4.3 Tests for Non-functional Requirements 4

4.4 Hardware and Software Requirements 4

5 Analysis 5

6 Conclusion 6

Appendix A - Group Log 7

# 

# Introduction

*<TO DO: Please provide a brief introduction to your project.>*

## Project Overview

Medival Maps is a map generation tool designed for use with table top board games such as Dungeons and Dragons. MM randomly generates city like grids, that are populated with buildings and characters that bring the town to life. The player can choose what size of town to generate, as well as the fantasy races that will inhabit this town.

Medival Maps are not meant to recreate a real city, rather provide a simple layout that a Dungeon Master (the person responsible for leading the table top game), can rely on to be geographically sound. Instead of creating arbitrary distances between buildings, and coming up with what sort of buildings are in the town on the spot, DM’s can autogenerate the town.

## Definitions, Acronyms and Abbreviations

DnD: Dungeons and Dragons, a table top role playing game.

DM: Dungeon Master, the person who runs the table top role playing game. Responsible for creating the world.

MM: Shorthand for Medieval Maps.

NPC: Non-Player-Character, a villager in the settlement.

## References and Acknowledgments

Wizards RPG Team. Dungeon Master’s Guide. Renton, WA; Wizards of the Coast. 2014.

# Design

## System Modeling

< Update your UML diagrams in milestone 2, to reflect the real implementation of this software.

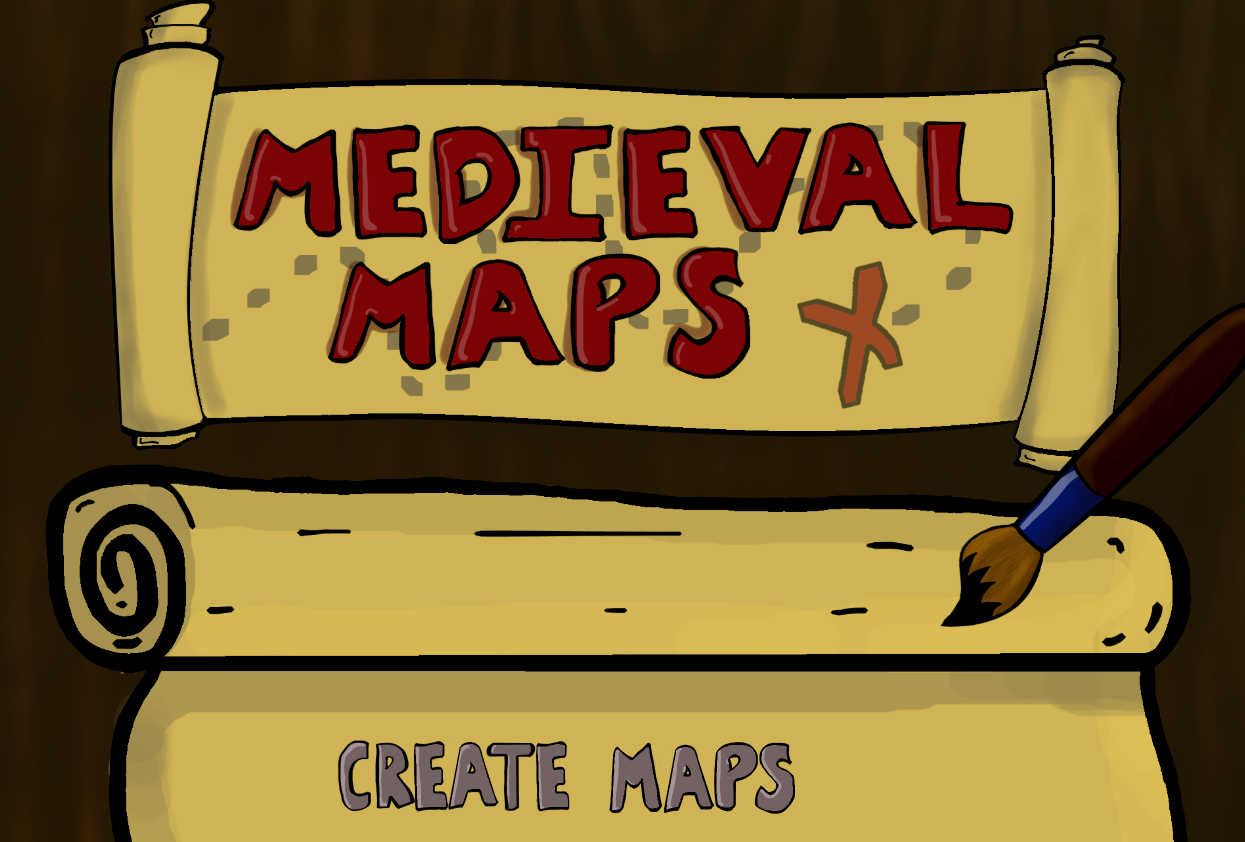
TO DO: Provide an updated version of the UML diagrams, including use case diagrams, sequence (or state) diagrams, activities diagrams, and class diagrams. If you don’t have an updated version, just mention: “our implementation strictly follows the design document (milestone 2)”. >

## Interface Design

<Provide several screenshots to illustrate your interface design.

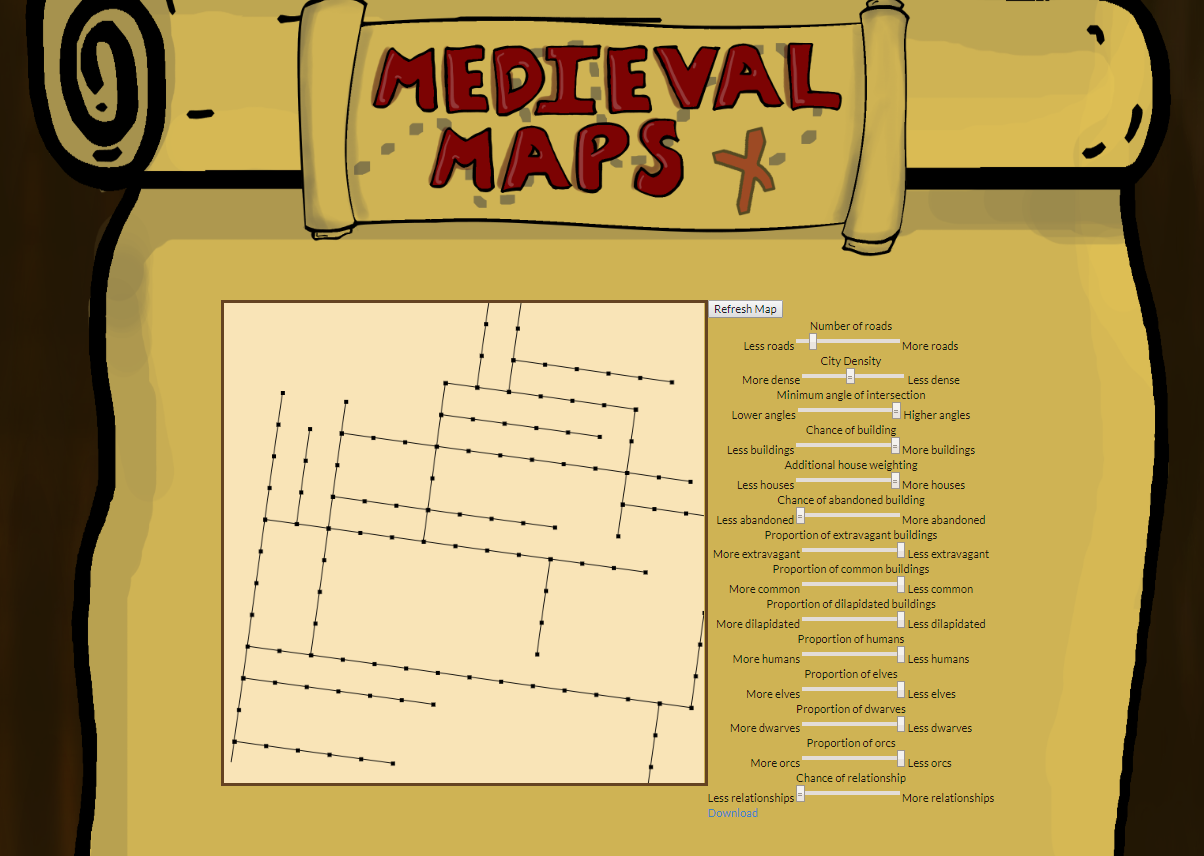
TO DO:

For each subsystem, pick one or two representative screenshots and paste here.>

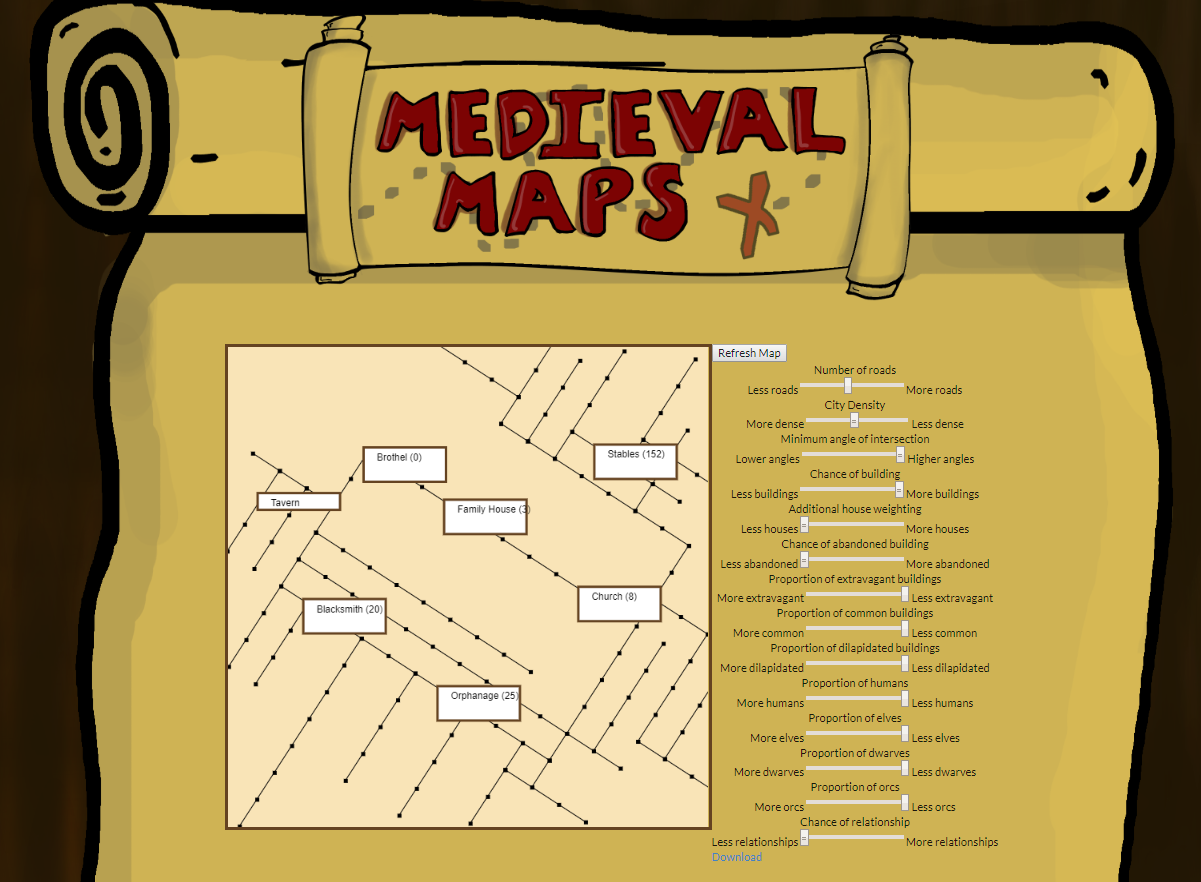


Home Page:

Home Page 2:



Map Generation:



# Implementation

## Development Environment

While developing this website, we mainly used IntelliJ, and JSFiddle for Javascript. We also used jQuery, HTML, CSS, and Meteor.

## Task Distribution

Patrick Wellington: Programming related to map generation and user input management. Also worked on CSS and Meteor integration.

Kevin Ngo: Programming related to NPC generation, and fantasy race distribution.

Lucas Gubala: Meteor implementation, CSS, algorithm work. Website art and layout. .txt documents for NPC generation.

## Challenges

**Curved roads:**

We initially wanted the maps to be generated with curved roads. While we were able to create an algorithm that could draw these curves, we had a hard time determining a method for making sure that the overlapping regions of the roads made logical sense. After attempting to fix this problem for some time, we decided that the map implementation would only use straight roads for simplicity and coherence.

**Map Drawing:**

Initially, we set out to create a map generator that would create somewhat realisitc looking maps. What we found in the process of trying to create these maps, was that there were many complications. After a significant time trouble shooting the program together, we came to the conclusion that the map utility could have a different use for the DM.

Due to the complications with drawing, curved roads, and working on the canvas in Javascript, we ended up with a different style of map then we set out to create. The map allows for a quick directional reference for players and DM’s, and can provide context for what might otherwise might be too large or daunting of a city to keep track of in game.

**Multiple User Settings:**

Due to time constraints, this was a feature that we decided to cut. Initially we intened to have a template system that would bank certain values, and allow for users to click on presets to populate the map sliders with those values. With the flexibility in the sliders that we wanted to provide to the user, we simply set the beginning presets to create maps that would serve as a good jumping off point for creation.

**Customizable buildings:**

While we wanted to have the ability to directly manipulate buildings on the user end, we ended up creating a functionality that allows for the building tags to be placed on the map when clicked on. This allows users to choose the buildings they want to highlight when exporting the map, and pick and choose what parts of the generation are being displayed.

# Testing

## <*This section is a summary of your testing report>*

## Testing Plan

Testing of NPC Generation:

* Test functionality that fill first NPC generated per building with specific job, tailored to that building eg., (Butcher generated in the Butcher Shop).
* Testing that NPC generation creates logical relationships between NPCs.
* Check that the building type is a valid listed building that has been generated and assigned a node on the map.

Map Generation:

* Test sliders to check for infinite loops during generation.
* Test orthogonality slider to see if maps generated are complicit with our standards.

## Tests for Functional Requirements

<Describe your test results for the functional requirements.

TODO: Provide a list of use cases or functions you have tested, as well as the testing results (whether or not the system passed the tests).>

Test Cases:

NPC Generator:

1. Case where NPCgen is passed an invalid building.
   1. Returned: Error and abort.
2. Case where NPCgen uses different fantasy race weighting.
   1. Expected proportions of fantasy races generated.
3. Case where optional parameter for first NPC generated in not present.
   1. Generates occupations randomly, without building specific occupation.

Map Generator:

1. Cases where density is at its max and min.
   1. Map successfully generated.
2. Case where user exports map with tags added.
   1. Map is exported with chosen tag visible.

## Tests for Non-functional Requirements

Our only non functional requirement that we wanted to attain, was that maps will generate in under 45 seconds.

The testing of this involved testing numerous combinations of slider inputs, and checking the total generation time for each. Our generation time for these maps uniformly was under 45 seconds, meeting our goal.

## Hardware and Software Requirements

Meeting these test cases required there to be error checking built into the software for NPC generation. These check made sure that any outliers would be handled by the sytem.

# Analysis

<In this Section you need to analyze the effort that has been put on this project.

TODO: Describe how many hours (approximately) each team member spent on the project, for each milestone, which milestone takes the most effort and why. >

Patrick Wellington: 46 hours

Kevin Ngo: 44 hours

Lucas Gubala: 45 hours

\*Majority of the work days were spent together, collaborating on all parts of the project.

Milestone 3 was the milestone that took the most effort. This makes sense, as it was the portion of the project where the vast majority of implementation was completed. If we had budgeted our time more effeciently, we could have mitigated some of the crunch time that we experienced near the end of the semester.

# Conclusion

<Conclude the document with what you have learned through working on the project.>

Appendix A - Group Log

< Describe how frequently the group meembers meet during the semester, and how effective the communication is. This is optional for one-person projects.>