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. Such a scale is, of course, beyond the reach of any manually created system.

Well known techniques such as fractals, L-systems, Perlin noise and others have been used to procedurally generate plants, terrain and cityscapes. This project seeks to develop a software system that allows users 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.

1.2. Similar Sites

The Medieval Fantasy City Generator(MFCG) is an amazing similar site. This application generates a random medieval city layout of a requested size: small, medium or large, which made up of different types of districts, and the generation method is rather arbitrary. Besides, the city can be given many elements, such as farm fields, citadel, plaza, temple, river, coast and so on. Since the background of this city generator is medieval, city is always inseparable from the walls and castle and everything is based on the user's choice. Additionally, it allows user to distort the map in order to modify some unsatisfying places by using warp tool. The author also mentioned that goal of application is to produce a nice looking map, not an accurate model of a city. Finally, user can save map as image in png or svg format by using export feature if he is satisfied with the map. Figure 1 is a screenshot of MFCG.

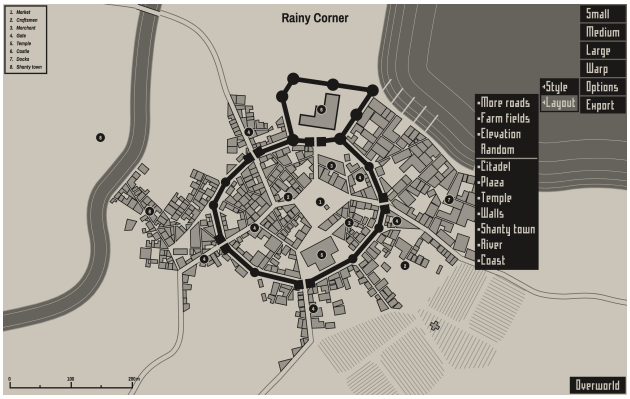

Figure 1. A screenshot of Medieval Fantasy City Generator

1.3. Project Goal

Our project goal is

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 2 explains in great detail a state chart.

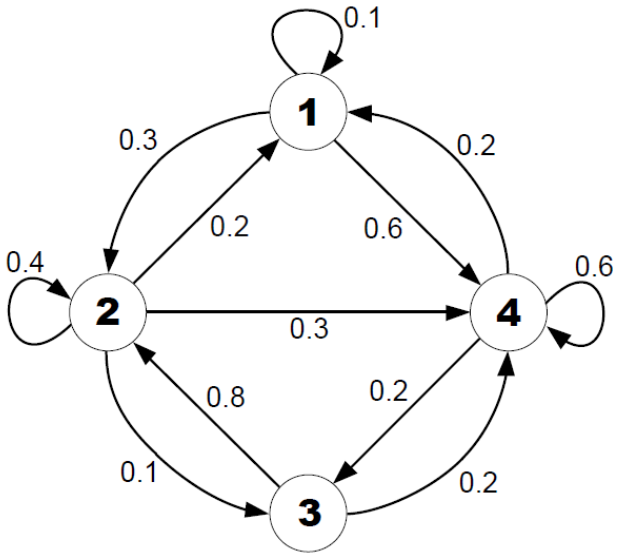

Figure 2. State Chart Diagram

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 3 explains in great detail a state chart.

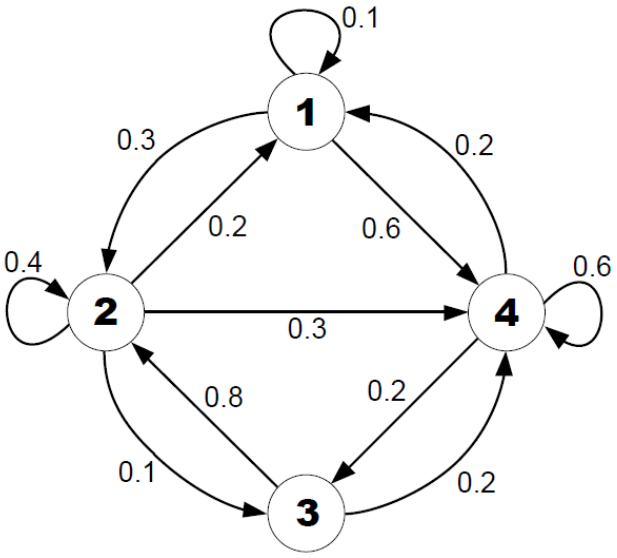

Figure 3. State Chart Diagram

3.3. Point 2

This gives Point 2

4. Bibliography

5. Appendices