Code Inspection Report

Un-Gerrymandered Software

Customer: Shawn Squire

DIAMOND DISTRICTING

Members: Corey Atkins, Matthew Hancher, Nahum Meherete, Joey Napolitano, Nirav Shah, Eric Yoo

Date: November 28, 2017

Un-Gerrymandered Software

Code Inspection Report

Table of Contents

1. Introduction ……...2

1.1 Purpose of This Document ……2

1.2 References ……..2

1.3 Coding and Commenting Conventions ……2 - 3

1.4 Defect Checklist ……3

2. Code Inspection Process ……..4

2.1 Description ……4

2.2 Impressions of the Process ……...4

2.3 Inspection Meetings ……5

3. Modules Inspected ……5 - 6

4. Defects ……..6 - 8

Appendix A – Agreement Between Customer and Contractor ……9 - 10

Appendix B – Peer Review Sign-off ……...11

Appendix C – Document Contributions
……11

Section 1: Introduction

Section 1.1: Purpose of this Document

The purpose of this document is to list any defects found within the code. The list may help aid improvements to the code as well as correct any defects that were found in the current version or save it for a later version of Un-Gerrymandered Software. Using the defects found in the prototype of Un-Gerrymandered Software, Diamond Districting can prevent defect multiplication from occurring at a later stage in the product. The customer should use this report to know what limitations exist in the Un-Gerrymandered Software.

Section 1.2: References

Diamond Districting. (2017, October 24). *Un-Gerrymandered Software System Design Document.*

Diamond Districting. (2017, October 19). *Un-Gerrymandered Software System Requirements Specification.*

Ingraham, Christopher. (2015, March 1). *This is the best explanation of gerrymandering you will ever see*. Retrieved from https://www.washingtonpost.com/news/wonk/wp/2015/03/01/this-is-the-best-explanation-of-gerrymandering-you-will-ever-see/?utm_term=.8c8e3fe95ad1

(2017, August 12). *Redistricting*. Retrieved from https://en.wikipedia.org/wiki/Redistricting#Gerrymandering

UMBC CSEE. (2015, Spring Semester). *C++ Coding Standards*. https://www.csee.umbc.edu/courses/undergraduate/202/spring15/projects/coding-standards.shtml.

Kevin Baas. (2012, November 12) *Auto-Redistrict*. http://autoredistrict.org/

The Qt Company. (2017) *Qt.* 5.6.x Offline Installers. https://www.qt.io/

Section 1.3: Coding and Commenting Conventions

We fully adopted our coding conventions from the UMBC Computer Science department's C++ Coding Standards. These standards are used for the CMSC 202 and CMSC 341 undergraduate courses. We chose this set of coding standards because all six of our project team members have taken and passed both CMSC 202 and CMSC 341, both of which are prerequisite requirements for this course. Therefore, these are the standards that all of our team members are most familiar with. Additionally, we feel that these standards are rigid

enough to make sure that all written source code is both readable and organized, but also flexible enough to keep functionality as a priority over aesthetics.

In particular, the standards include rules for filenames and extensions, file organization, class definitions, variable declarations, function declarations, and documentation. However, we had to slightly modify the conventions to accept files other than .cpp and .h files. Our project includes .ui files for the implementation of the graphical user interface. More information on these coding standards can be accessed through the reference provided in Section 1.2.

Section 1.4: Defect Checklist

Table 1 is a comprehensive tabular checklist of possible defects that our team used during the inspection process. For each of the 15 defects, the category is provided. The category may be one of the following: Logic Error, System Error, Runtime Error, Commenting, Aesthetics, Coding Conventions, Inconvenience, or External Defect.

Table 1. Defect Checklist

Defect	Category of Defect
Shapefiles are not as accurate as expected	External Defect
Tries for perfection, will not finish itself	Logic Error
Bodies of water are unrecognizable	Logic Error
Results are not the same each iteration	Logic Error
Time for execution is not manageable	Runtime Error
The system must be manually restarted to run again	System Error
No feature to rescale image with given input	Logic Error
Cannot update image in the same window, new window has to be created	Inconvenience
UI will only display images from the .ui file and not from the C++ file	Logic Error
District selection is currently fixed	Logic Error
Insufficient comments in header files	Commenting
Insufficient comments in implementation files	Commenting
User interface is not aesthetically-pleasing	Aesthetics
Variable names in some of the source code	Coding Conventions

files are not consistent	
File header comments are not present in any of the .cpp and .h files	Coding Conventions

Section 2: Code Inspection Process

Section 2.1: Description

To begin the code inspection process, we had scheduled a time to meet online or in person(preferably online) and decide which aspect of the code should be looked at for the given meeting. We then begin by running our code. After running the code, we analyzed our final results. We then analyzed the customer's requirements. After analyzing both, we were able to identify adjustments that needed to be made where our product deviated from our customer's requirements. For each defect found, we look over the defect and determine whether the defect is a logic error, a coding convention violation, a user friendliness issue, or if it is not correct to the customers standards.

Once the defects have been evaluated, we logged the defects in Section 1.4. Identifying the defects will aid us in developing future versions of the Un-Gerrymandered Software. Although our method for inspection was effective we feel as if it was not "ideal" in the sense that we as a group could not really do the inspection meetings in person. Having it done in person would make the meeting more efficient as everyone can relay their thoughts immediately and receive quick and authentic feedback. Our approach was not "ideal" because we did not do defect checking using SLOC as a metric, we used each specific module as a metric to detect defects.

Section 2.2: Impressions of the Process

We feel as if our method for inspecting the code was effective because we were able to schedule what modules needed to be checked within every meeting. This prevented us from having to deal with too many modules at once. Setting up modules to look at before each meeting allowed us to better search for any defects within our algorithm and GUI. A downside to our code inspection would be how we were only able to meet via internet. If we were able to meet in person we could better see the defects as a collective whole rather than testing by ourselves and making it more of a single act of detection.

From the GUI aspect, the best module in terms of minimal errors is the drop down menu. We are certain that the drop down module is error free and will work for any given test case. The worst module for GUI is that the districts within the code are hardcoded. Having the districts hardcoded does not allow for the code to be as dynamic as we had hoped for it to be, as well as it limits how the user can use the software to the best of its ability.

Section 2.3: Inspection Meetings

Table 2 is a list of code inspection meetings that our team has held thus far. It includes the date of the meeting, the location, the start time, the end time, who the participants were, the role of each participant, and the specific units that were covered in the meeting.

Table 2. Inspection Meetings

Date	**Location**	**Time start**	**Time end**	**Participants - Roles**	**Units covered**
11/27/2017	GroupMe (Virtual)	1:40PM	4:30PM	Nahum - Scribe Matt - Inspector Eric - Inspector Nirav - Moderator	Shapefiles not loading properly, Code has to be restarted manually
11/25/2017	GroupMe (Virtual)	5:00pm	8:00pm	Corey - Inspector Eric - Inspector Matt- Inspector Nahum - Scribe Nirav - Moderator	GUI code for state and district selection. Zoom function is logically incorrect.

Section 3: Modules Inspected

Table 3 lists the completed modules. For each module, a brief description and a projected date of completion is provided.

Table 3. Inspected Source Code Modules

Name	**Functionality**	**Where it fits into SDD design & architecture**
Drop Down Menu for State Selection	The user selects one of the three states (either Maryland, North Carolina, or Wyoming). In effect, a sub-menu appears for the number-of-districts selection.	This module fulfills the “State Selection” requirement that was noted in the SDD.
Drop Down Menu for District	The user selects the number	This module fulfills the

Selection	of districts desired for a selected state. The set of possible values depends on the state. In effect, a .png image of the selected state with the selected number of districts is opened up in a new window.	“District Selection” requirement that was noted in the SDD.
Zoom In & Zoom Out Buttons in Map Window	The user can zoom into the window by a certain amount of units by pressing the zoom in button, denoted by a “+” sign. Similarly, the user can zoom out of the window by pressing the zoom out button, denoted by a “-” sign.	This module fulfills the “Zoom In & Zoom Out Feature” requirement that was noted in the SDD.

Section 4: Defects

Table 4 shows the 15 open defects that we are currently trying to resolve. For each defect, the name, the category of the defect, the name of the module that the defect is a part of, and the description for the defect.

Table 4. Open Defects

Defect Name	Category	Name of Module	Description
Shapefiles are not as accurate as expected	External Defect	Algorithm	Shapefiles of states must be pre-formatted with population census data to run correctly. Otherwise, algorithm won’t recognize the file and go into infinite loop.
Tries for perfection, will not finish itself	Logic Error	Algorithm	Until the average population is perfect among all districts, it will continue to run even if impossible to make perfect.

Bodies of water are unrecognizable	Logic Error	Algorithm	Districts will be connected across water
Results are not the same each iteration	Logic Error	Algorithm	There are no guarantees that selecting the same number of districts will give you the same results every time
Time for execution is not manageable	Runtime Error	Algorithm	Timing out or non progression isn't detected in the algorithm and will go through an infinite loop.
The system must be manually restarted to run again	System Error	Algorithm	Upon completion of the redistricting algorithm, you must reload the shapefile to start again.
No feature to rescale image with given input	Logic Error	Zoom Function	Zoom feature only minimizes and maximizes the window, not image.
Cannot update image in the same window, new window has to be created	Inconvenience	District Selection	In order to select a new district, you must exit the current window in order to display new window.
UI will only display images from the .ui file and not from the C++ file	Logic Error	State Selection	If two different images were coded to be displayed in same window, the .ui file has precedence over .cpp file.
District selection is currently fixed	Logic Error	District Selection	The districts are preloaded screenshots. User can only select what's given.

Insufficient comments in header files	Commenting	Code	No comments on what each consist of.
Insufficient comments in implementation files	Commenting	Code	No comments describing the code or process.
User interface is not aesthetically-pleasing	Aesthetics	Code	User interface has hard coded districts and preloaded images.
Variable names in some of the source code files are not consistent	Coding Conventions	Code	Variable names in some of the source code files are not camel-cased.
File header comments are not present in any of the .cpp and .h files	Coding Conventions	Code	No comments on what each file is and what it's suppose to do.

Appendix A – Agreement Between Customer and Contractor

Client Agreement

Shawn Squire
1000 Hilltop Circle
Baltimore, MD 21250

The following represents an agreement between **Diamond Districting** (hereinafter referred to as “we”, “us”, or “Diamond Districting”) and **Shawn Squire** (hereinafter referred to as “you” or “Client”). The details of this agreement are as follows:

Professional Services. The Client hereby contracts with Diamond Districting to perform a visualization of non-gerrymandered states.

Description of Services. The following services will be provided:

· Files containing all source code to Un-Gerrymandered Software

Other Terms/Customer Comments

_____
_____
_____
_____
_____

Terms and Conditions

Limited Liability. We shall not be liable for any delay due to circumstances beyond our control to provide services, including acts of God, war, government regulations, disaster, or civil disorder.

Amendments. Any changes or modifications must be specifically placed in writing, attached, dated, signed, and approved by both parties.

Cancellation. Cancellation of services should be provided to **Diamond Districting** in writing to amend the current client agreement. In the event that the client cancels the contracted services outlined in this contract, the initial payment will be forfeited.

I have read and understand the terms of the entire agreement. I hereby agree to the terms of this agreement. We both agree to make the attached Terms and Conditions as part of this Agreement.

Client

Shawn Squire: ____Shawn Squire_____ **Date** ___11/28/2017_____

Diamond Districting (Team)

Corey Atkins: _____Corey Atkins_____ **Date** _11/28/2017_

Matthew Hancher: _____Matthew Hancher_____ **Date** _11/28/2017__

Nahum Meherete: _____Nahum Meherete_____ **Date** __11/28/2017__

Joey Napolitano: _____Joey Napolitano_____ **Date** _11/28/2017__

Nirav Shah: _____Nirav Shah_____ **Date** __11/28/2017_

Eric Yoo: _____Eric Yoo_____ **Date** __11/28/2017___

Appendix B - Peer Review Sign-off

All members of Diamond Districting have reviewed the document and agree on its content and format.

Corey Atkins: _____Corey Atkins_____ **Date** _11/28/2017____

Matthew Hancher: ____Matthew Hancher_____ **Date** _11/28/2017_____

Nahum Meherete: ___Nahum Meherete_____ **Date** _11/28/2017___

Joey Napolitano: ____Joey Napolitano_____ **Date** _11/28/2017_____

Nirav Shah: __Nirav Shah__ **Date** _11/28/2017___

Eric Yoo: _____Eric Yoo_____ **Date** _11/28/2017____

Comments:
__None_____

_____

_____

Appendix C – Document Contributions

Atkins, Corey - Introduction (16.67%)
Hancher, Matthew - Introduction (16.67%)
Meherete, Nahum - Code Inspection Process (16.67%)
Napolitano, Joey - Modules Inspected & Defects (16.67%)
Shah, Nirav - Modules Inspected & Defects (16.67%)
Yoo, Eric - Code Inspection Process (16.67%)