



Disaster Reactor

Version 1.0 approved 4/7/2016

Abstract

This document will help further the understanding of Disaster Reactor which is intended to help people around the world in need of aid. We hope that the readers of this document will understand how the system works so that the simulation can be used as efficiently as possible.

Prepared by: Amanda Alfaro, Khaleel Harris, Craig Jones, Sam Shrestha, Zachary Smith, Nealan Vettivelu

Contents

Revision History	3
1 Introduction	4
1.1 Purpose	4
1.2 Scope	4
1.3 Design Summary	4
1.4 Intended Audience	5
1.5 References	5
Conceptual System Architecture	6
1.6 Overview	6
1.7 Setup	6
1.7.1 Infrastructure	6
1.7.2 Damage, Value, Difficulty	7
1.7.3 Spawn Points	7
1.8 Simulation	7
1.9 Output	7
1.9.1 PDF Export	7
Technical System Architecture	8
1.10 Complete System Overview	8
1.11 Application Use Case	9
1.12 Renderer Subsystem	10
1.13 GUI Subsystem	10
1.14 Data Subsystem	10
1.15 Input Stage Subsystem	10
1.16 Simulation Stage Subsystem	10
1.17 Output Stage Subsystem	10
Rationales for Choices	11
1.18 Rationales for the Structure of the AI	11
1.19 Rationales for the Structure of the Main Application	11
Physical View	11
Data View	11
Work- Assignment View	12
1.20 Members Assigned to Program Framework	12

1.21	Members Assigned to the Input Stage.....	12
1.22	Members Assigned to the Simulation Stage.....	12
1.23	Members Assigned to the Output Stage.....	13
1.24	Members Assigned to Quality Assurance and Software Conformity.....	13
	Development View.....	14
1.25	Project Organization Overview	14
1.26	Source Code Organization.....	14
	Element Catalogue.....	19
1.27	Use Case Diagram Catalogue	19
	Development View Catalogue	20
	User Interface	20
1.28	Overview	20
1.29	Input Stage	21
1.29.1	Load Map Button	21
1.29.2	Inspect Mode Panel	22
1.29.3	Paint Mode Panel.....	23
1.29.4	Run Sim Button	23
1.30	Simulation Stage	24
1.30.1	Agents Toggle.....	24
1.30.2	Environment Toggle	24
1.30.3	Spawns Toggle.....	25
1.30.4	Trails Toggle	25
1.30.5	Stop Sim Button	26
	Other.....	26
1.31	Appendix A: Glossary	26

Revision History

Name	Date	Reason For Changes	Version
Nealan Vettivelu	4/5/2016	Updated Rationales, Work-Assignment View, Development View, Glossary	1.0
Nealan Vettivelu	4/6/2016	Updated Development View	1.0
Khaleel Harris	4/6/2016	Documented User Interface visualizations and descriptions	1.0
W. Craig Jones	4/7/2016	Expanded Document Structure	1.0
W. Craig Jones	4/7/2016	Expanded Work Assignment View	1.0
Sam Shrestha	4/7/2016	Updated Diagrams	1.0
W. Craig Jones	4/7/2016	Expanded Conceptual View	1.0
W. Craig Jones	4/7/2016	Documented Subsystems	1.0
Zachary Smith	4/7/2016	Updated Introduction and Complete System Overview	1.0
Amanda Alfaro	4/7/2016	Created Abstraction	1.0
Khaleel Harris	4/7/2016	Expanded UI Section 24. Input Stage	1.0
Amanda Alfaro	4/7/2016	Expanded UI Section 25. Simulation Stage	1.0

1 Introduction

1.1 Purpose

This document is intended to be used as an aid to help understand the underlying features of the Disaster Reactor software as stated in the Software Requirements Specification document. In addition to that, this document is meant to get all people involved in the use and development of the software on the same pace.

1.2 Scope

The goal of this Software Design document is to provide an overview of the software from a technical and marketable point of view. This will give developers a way to better understand how the various systems work and also give the marketers a way to better sell the product. The software is designed to give first responders an easy way of simulating disaster recovery.

1.3 Design Summary

Due to the nature of simulating real world events the range of devices that can run this software will be limited to higher end devices. In addition to this there are technical limitations that prevent some high end devices from running this software. The requirements for running the software are listed in the System Design Specification document. For the devices that can run the software, the goal is to make the software simple to use. This is because the software will be used in high stress situations and by taking some of the stress off of the user it enables them to make better decisions and also use the software better.

The design pattern for Disaster Reactor is pipe and filter. This is because just like in pipe and filter pattern, all this software is doing is taking inputs, making some calculations based on the data and presenting it to the user. In this case the input are map data, damaged area information and recovery information. With this information the software will simulate what will happen in the situation and report the findings back to the user.

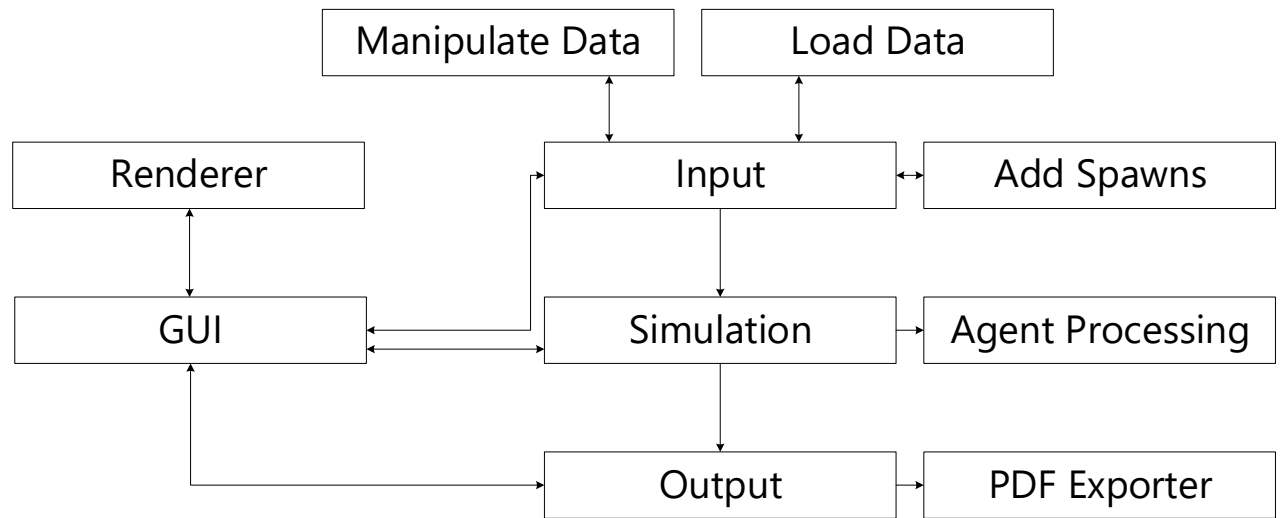
1.4 Intended Audience

The intended audience for this document is people who have an impact on the development of this software. Some of the people that have an impact on the software include developers, consultants, potential investors and distributors of the software. For people that do not have a direct say-so in the development of the software, section 2 of this document will be of interest since it provides a basic overview of the software. While people who are more directly involved in the development cycle of the software should view section 9 as it provides a more in-depth view of software systems. Both groups of people should view section 12 of this document as it shows a preview of what the software is intended to look like.

1.5 References

- Simple and Fast Multimedia Language
 - <http://www.sfml-dev.org/>
- OSM XML Documentation
 - http://wiki.openstreetmap.org/wiki/OSM_XML/

2 Conceptual System Architecture



2.1 Overview

Disaster recovery is an essential service provided by governments and aid groups around the world. The delay between a disaster occurring and a response is critical, and every second can mean lives lost. Some disasters have pre-made plans that can be quickly implemented, but that's not always the case. Those situations that require the rapid development of a recovery plan, and Disaster Reactor will facilitate that goal.

2.2 Setup

The user must first provide input data to Disaster Reactor before the simulation can begin.

2.2.1 Infrastructure

The first input required by the user is Infrastructure Data. This is provided in the form of an Open Street Map file (.osm), which supplies information about roads to the simulation. Roads provide significant benefits to the mobility of recovery agents. It also gives reference points for when the user is asked to supply the next set of information.

2.2.2 Damage, Value, Difficulty

Next, the user supplies data about areas of damage, value, and difficulty. Areas of damage refer to asset damage that needs repair or areas that are afflicted by the disaster and need aid. Areas of value are locations that have population or assets. Areas of difficulty are locations that are difficult to traverse, such as swamps. Simulation agents will attempt to provide aid to areas that have value and are damaged, while avoiding areas of difficulty if possible. This data is painted onto the map using a mouse, and changes values on the disaster display area.

2.2.3 Spawn Points

Finally, the user will supply spawn locations. These are locations that the simulation agents will appear from, and should represent locations that have supplies or the infrastructure to quickly deploy recovery efforts.

2.3 Simulation

The user will then begin the simulation. The simulation will generate thousands of tiny simulation agents. The simulation will allow each agent to perform some actions every frame, and each agent tries to perform the most aid using those actions. The simulation is intended to be a bulk statistical model representing "aid" flows. The user may end the simulation at any time, and will then be able to see pathways that most effectively supplied aid.

2.4 Output

Once the user has ended the simulation, they will be able to view the distribution of agents across the map, and the tracks that the agents took to get there. This should provide a first approximation of a recovery plan, which can then be further developed into an implementable recovery plan.

2.4.1 PDF Export

The user will have the option to export several different data views to a pdf file for distribution or printing.

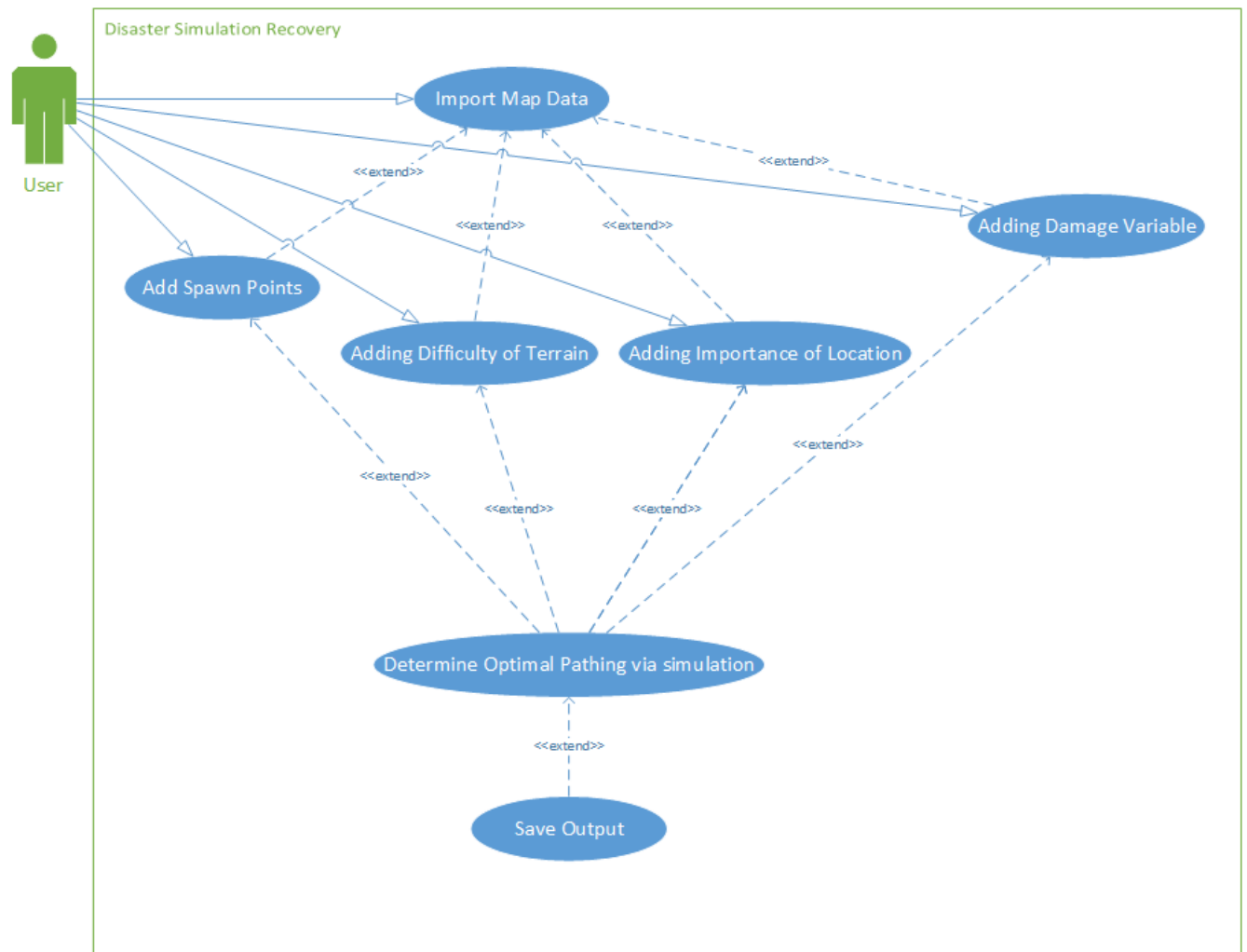
3 Technical System Architecture

Disaster Reactor		
Input	Simulation	Output
GUI	Renderer	Data
Windows OS		

3.1 Complete System Overview

The goal of Disaster Reactor is to provide an easy to use system to help aid first responders in the wake of an natural disaster. This is done by dividing the software into three major components. The first component is the loading of the map data for the effected area. The map data is in the form of OSM. The second component is the editing of the map. The user is able paint on damaged area and select where AGENTS will be stationed. The third component is displaying the outputs from the simulation. The map will take advantage of SFML and show the actions of the AGENTS and what paths they take.

3.2 Application Use Case



From the user's point of view, there are five different actions which can be accomplished. These are adding new spawn points, adding difficulty of terrain, adding importance of location, adding damage variable, or importing map data. The simulation output will automatically be saved by the program.

3.3 Renderer Subsystem

The program will use a common rendering subsystem build using SFML. This is a low level subsystem will control the actual interface between the program and graphics stack. This will facilitate a consistent graphical experience throughout the program. It will be responsible by rendering the GUI elements and the Simulation map pane.

3.4 GUI Subsystem

The GUI Subsystem integrates with the Renderer subsystem to provide a common and consistent user interface for each stage. It will also keep track of user input from the mouse and keyboard.

3.5 Data Subsystem

The data subsystem represents the pipes in the pipe and filter system. It is passed from stage to stage, holding the changes each stage makes. The Data Subsystem is implemented as a dedicated class that is created by the Input stage and modified by the other stages.

3.6 Input Stage Subsystem

The Input Stage Subsystem will integrate with the Renderer, GUI, and Data subsystems in order to facilitate user input. The Input stage will populate the Data subsystem.

3.7 Simulation Stage Subsystem

The Simulation Stage processes the Data Subsystem and simulates many tiny aid agents until the user is satisfied with the results. The simulation tries to maximize the aid potential of each agent per iteration.

3.8 Output Stage Subsystem

The Output Stage will take all of the data that was collected in the Simulation Stage and display it for the user. The stage will allow the user to apply filters on what subsets of the data the user wants to see, such as final agent locations, agent aid distribution, or agent motion.

4 Rationales for Choices

4.1 Rationales for the Structure of the AI

The artificial intelligence has been constructed in this way with the primary concern being speed as well as low memory usage due to the time critical nature of the application, and complexity of the program. To solve the first issue of speed, we made the AI run in iterations, allowing for discrete intervals. This meant that the simulation was able to be stopped at any point, so that the user would be able to utilise the results immediately.

To utilise as little memory as possible the AI has been coded into a three tiered system. This allowed for agents to get information at a much higher compression due to the tiers summarising information at different scales. The first tier can be considered to be the strategic layer in which macro decisions would be made as to where aid was generally needed. The intermediary tier is called the sub-commander which allowed for a more micro focus on which general direction an agent should be moving in. The final tier can be considered the individual agents which is the immediate decision that should be taken by an agent. The simulation is intended as a statistical model for recovery rather than suggesting the movements of individuals. Those decisions must be decided by the end user of the application.

4.2 Rationales for the Structure of the Main Application

When designing the main application for Disaster Reactor the main goal was to make the design as easy to read and navigate as possible. For this reason, we chose larger buttons as well as contrasting colours to ensure that the user was able to identify quickly what needed to be done.

5 Physical View

We are not utilising any hardware in this project. This section is reserved for future iterations of the design which may potentially include the use of hardware.

6 Data View

We are not implementing a database or data model in this project. This section is reserved for future iterations of the design which may potentially include a database or data model.

7 Work- Assignment View

7.1 Members Assigned to Program Framework

The Program Framework will be developed by:

- W. Craig Jones
 - Craig will ensure that the program framework is developed in a fashion that will facilitate the design laid out by this document.

7.2 Members Assigned to the Input Stage

The Input and Output stage will be developed by:

- W. Craig Jones
 - Craig will ensure that the Input Stage properly interface with the Program Framework, as well as the rendering of the Input Stage.
- Zachary Smith
 - Zachary will focus the painting portion of the Input Stage and GUI.
- Khaleel Harris
 - Khaleel will focus on the file importing portion of the Input Stage and GUI.
- Sam Shrestha
 - Sam will ensure that the Input Stage fulfils the needs of our target audience.
- Amanda Alfaro
 - Amanda will ensure that the Input Stage meets our user experience expectations.

7.3 Members Assigned to the Simulation Stage

The Simulation implementation has been tasked to the following software developers:

- Nealan Vettivelu
 - Nealan will focus on the implementation of the artificial intelligence used by the simulation.
- W. Craig Jones
 - Craig will focus on the rendering of the Simulation Stage and graphical filters that can be applied to the simulation.

7.4 Members Assigned to the Output Stage

- W. Craig Jones
 - Craig will ensure that the Output Stage properly interface with the Program Framework as well as the rendering of the document.
- Zachary Smith
 - Zachary will focus the printing of the output results, as well as the GUI.
- Khaleel Harris
 - Khaleel will focus on exporting of output results, as well as the GUI.
- Sam Shrestha
 - Sam will ensure that the Output Stage fulfils the needs of our target audience.
- Amanda Alfaro
 - Amanda will ensure that the Output Stage meets our user experience expectations.

7.5 Members Assigned to Quality Assurance and Software Conformity

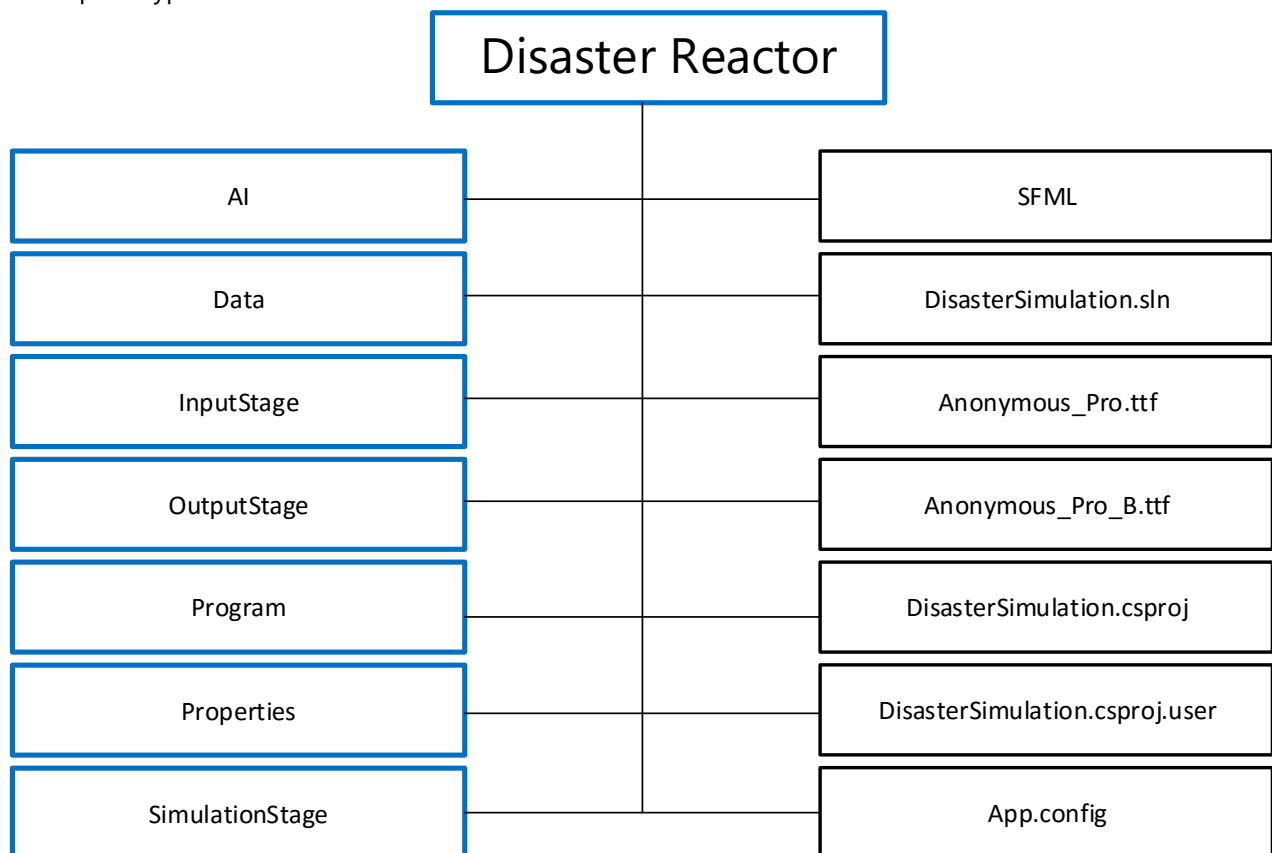
- Amanda Alfaro
 - Amanda will ensure that the software meets the requirements set out by the document.
- Sam Shrestha
 - Sam will ensure that the software meets our quality expectations.

8 Development View

8.1 Project Organization Overview

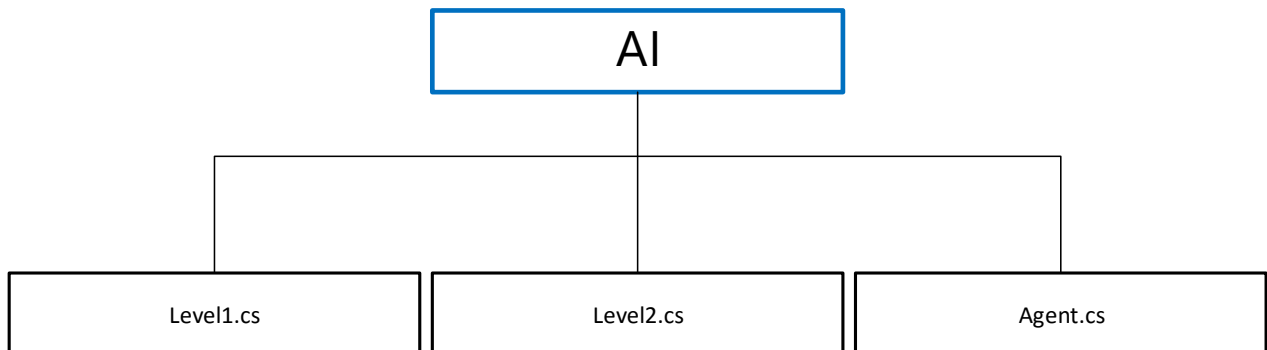
Disaster Reactor has been built within Visual Studio 2015 Community using both C# and Simple and Fast Multimedia Library (SFML). The directory structure can be seen in image below.

NOTE: Within this section, an blue box denotes a directory while a black box denotes a file within the directory. Also Disaster Simulation is still used for some file names as it was the legacy name during the initial prototype.

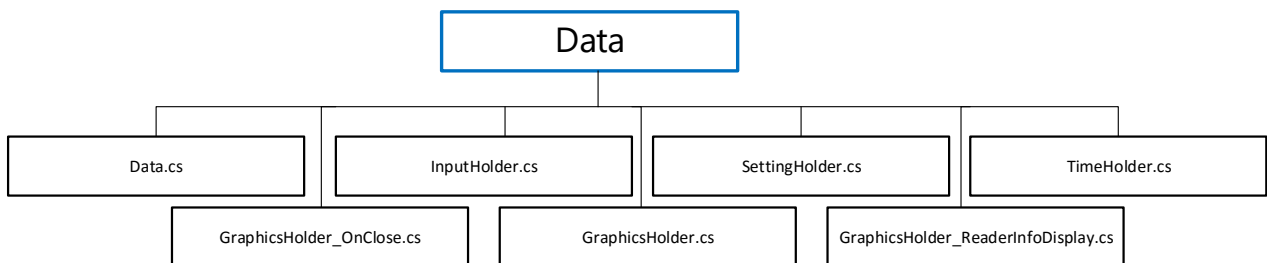


8.2 Source Code Organization

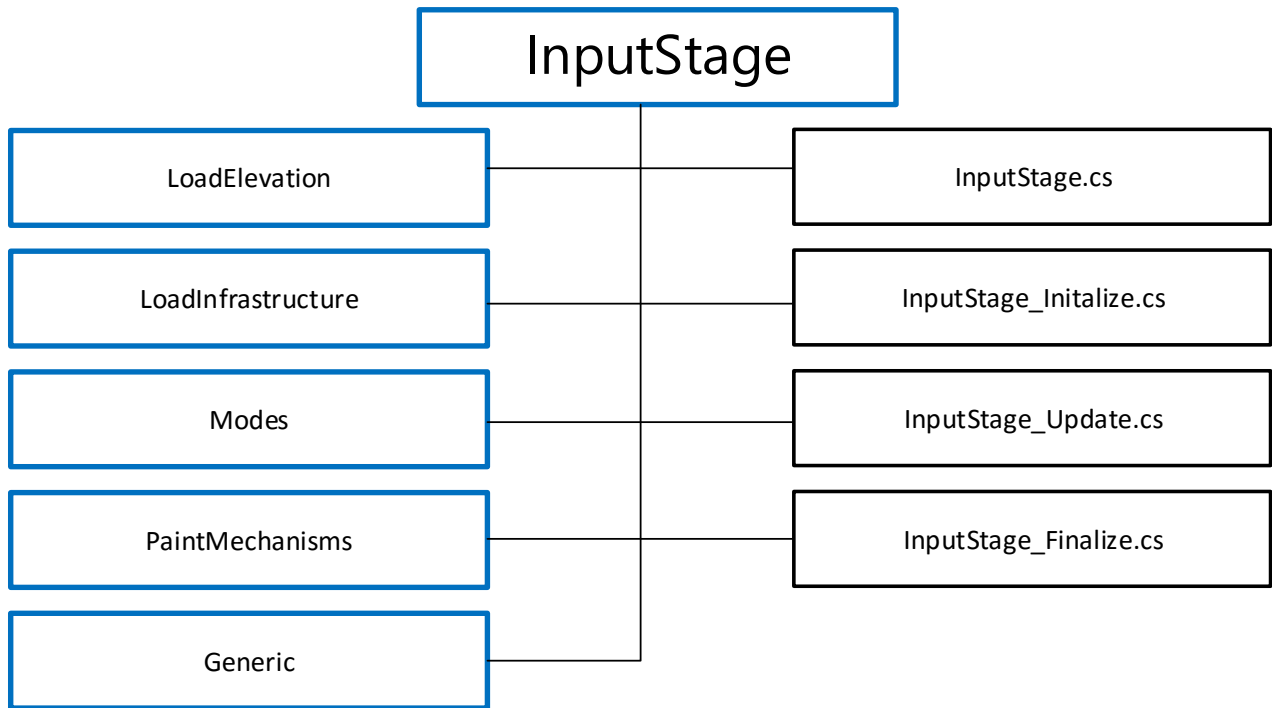
There will be six main source directories within the main project directory. These directories will contain code relating to a particular component or logical distinction. The five directories are as follows:



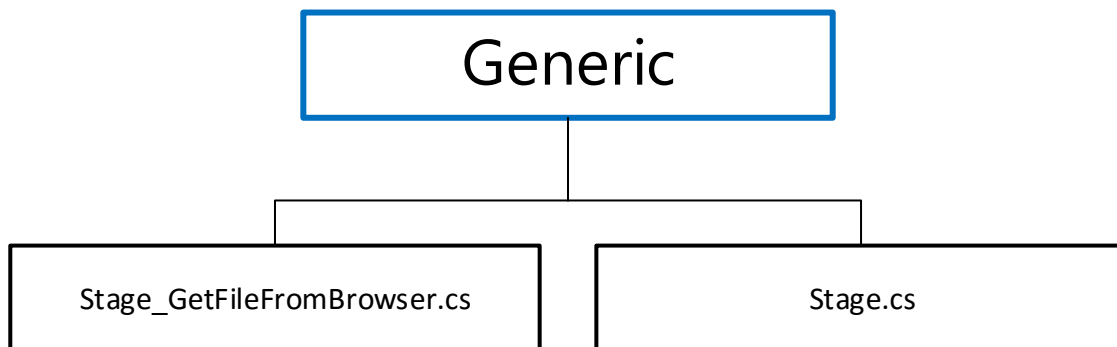
- Artificial Intelligence – Contains source files on how the agents will behave given the input data.



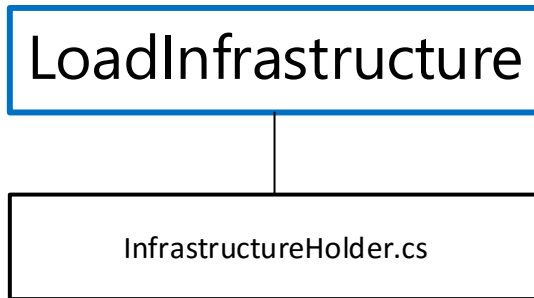
- Data – Contains source files pertaining to the overall user settings of the program as well as the data passed from stage to stage.



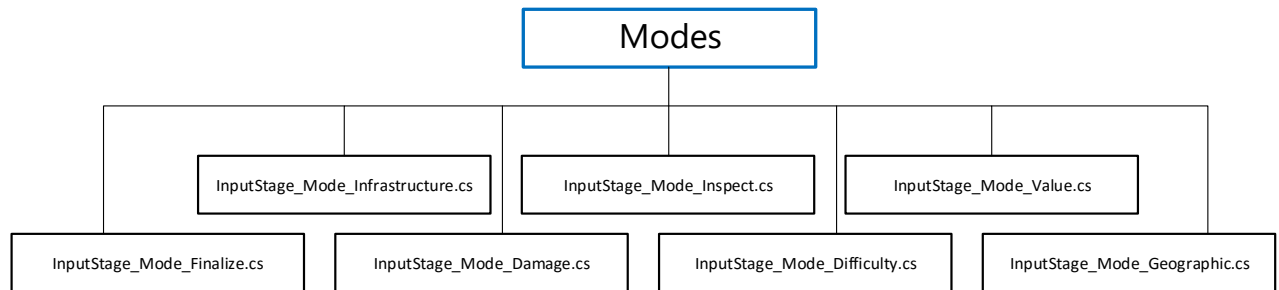
- InputStage - . Contains source files regarding the loading of input files, as well as initialisation of all major data structures and population of the Data object.



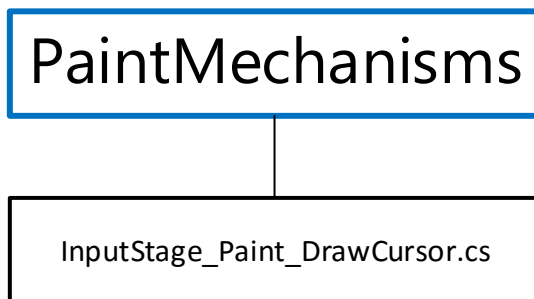
- Generic – Contains source files for the base class of the stages



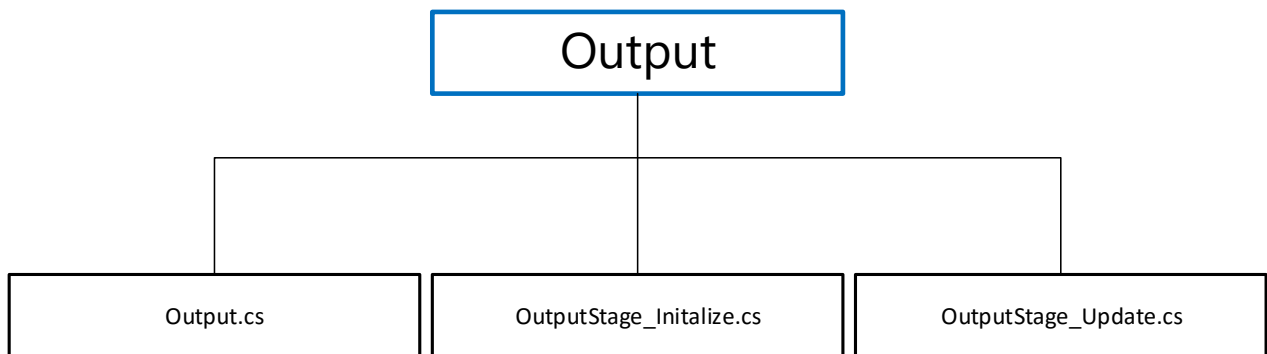
- LoadInfrastructure – Contains source files regarding road data.



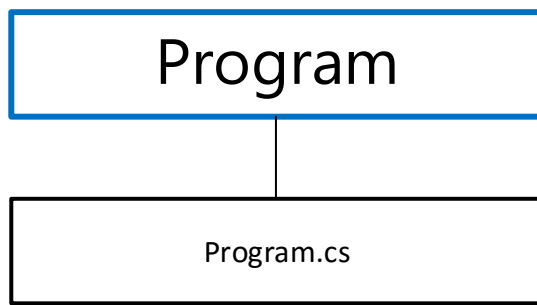
- Modes – Contains source files regarding the different modes in the input stage



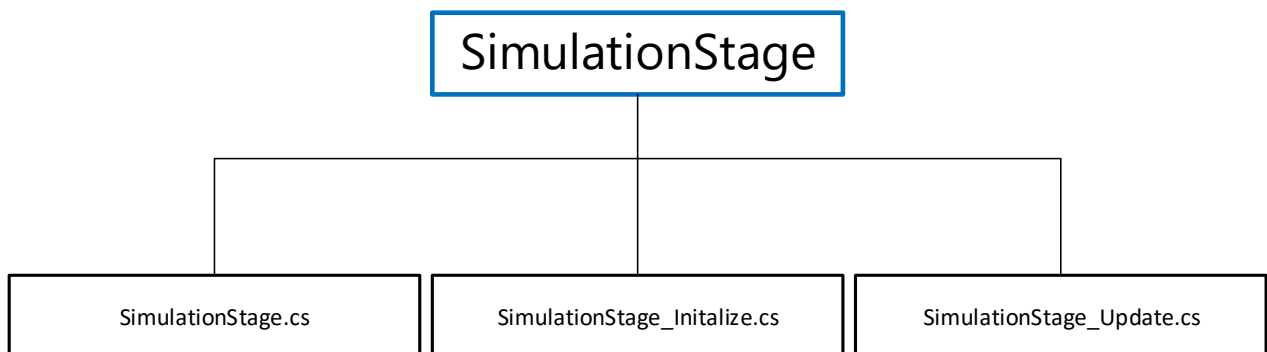
- PaintMechanisms – Contains source files regarding the cursor.



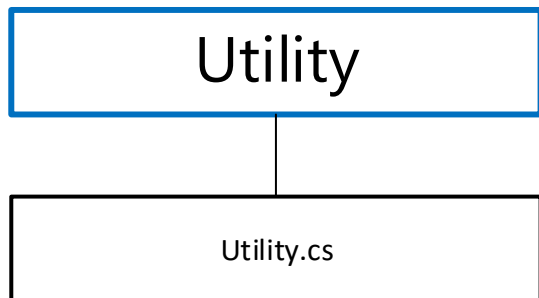
- Output Stage- Contains source files pertaining to the output of the simulation results including printing out the results as a pdf.



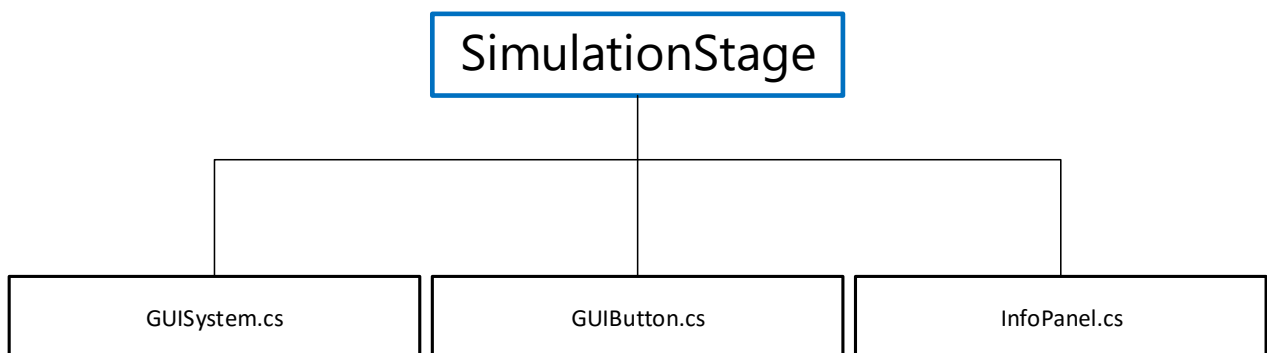
- Program – Contains source code files primarily surrounding the rendering of the interface and flow between stages



- SimulationStage – Contains source files regarding the simulation part of the program




- Utility – Contains source files regarding miscellaneous functions that are used by several different processes



- Graphics – Contains source files regarding the **GUI** system

9 Element Catalogue

9.1 Use Case Diagram Catalogue

Use Case Diagram	
Symbol	Description
	Actor
Oval	Use Case
Rectangle	Subsystem
Arrow	Association

10 Development View Catalogue

Development View	
Symbol	Description
Blue Rectangle	Directory
Black Rectangle	File
Black Line	Contains Relationship

11 User Interface

11.1 Overview

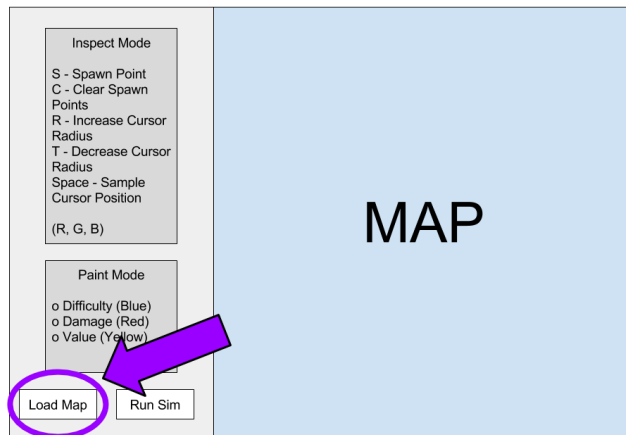


Graphical User Interface

These two images are sketches of the Graphical User Interface for the two stages (Input Stage and Simulation Stage) of our Disaster Reactor program.

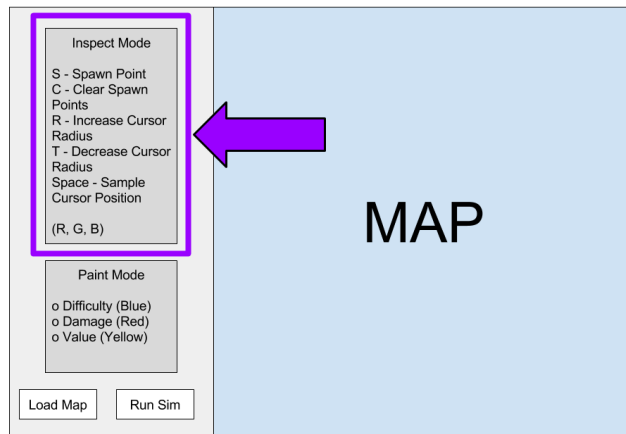
11.2 Input Stage

11.2.1 Load Map Button



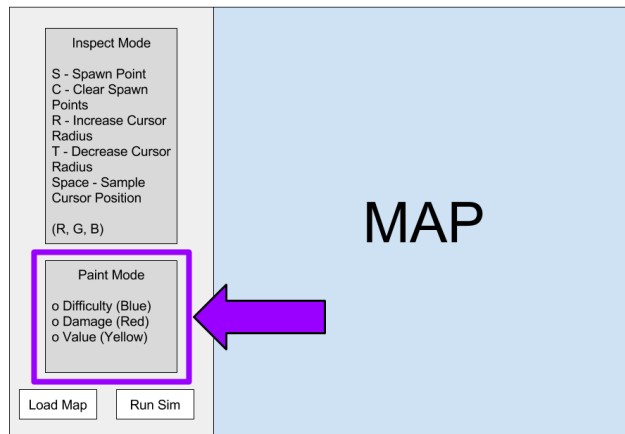
- This button allows the user to load an .osm file into the program to populate the simulation environment.
- We chose to use a button for this function instead of other methods (e.g. hotkeys) because we felt that, in this case, a button would be better in providing a clean and simple interface.

11.2.2 Inspect Mode Panel



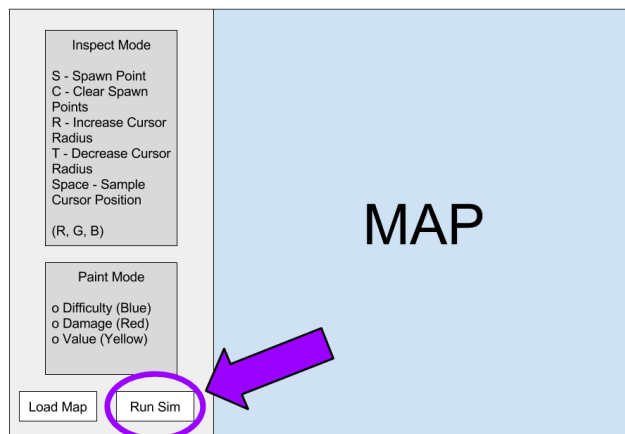
- *Disabled until a valid map has been loaded and while Paint Mode Panel is active.*
- This panel allows the user to:
 - Choose and delete spawn points.
 - Increase or decrease cursor radius.
 - Sample properties at cursor position.
- We chose to use hotkeys to activate different functions while in Inspect Mode because this allows the user to focus on the map and change modes simultaneously without the need to divert their attention to the panel.

11.2.3 Paint Mode Panel



- *Disabled until a valid map has been loaded and while Inspect Mode Panel is active.*
- This panel allows the user to paint values that correspond to three different attributes - Difficulty, Damage and Value - onto the map.
- The values of these attributes influence how the spawned agents interact with their surroundings.
- We chose to use radio buttons for the Paint Mode panel because they are a clean and simple way to show the user that only one variable may be painted at a time.

11.2.4 Run Sim Button



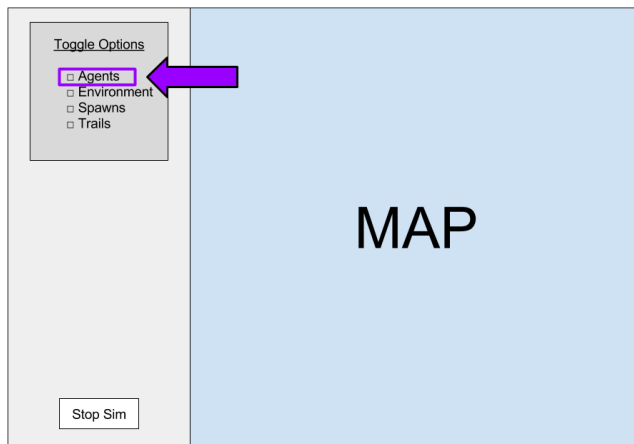
- *Disabled until a valid map has been loaded.*
- If the user has loaded a valid map into the program, this button switches the program into Simulation Stage which begins the simulation.

11.3 Simulation Stage

Design of Simulation Stage:

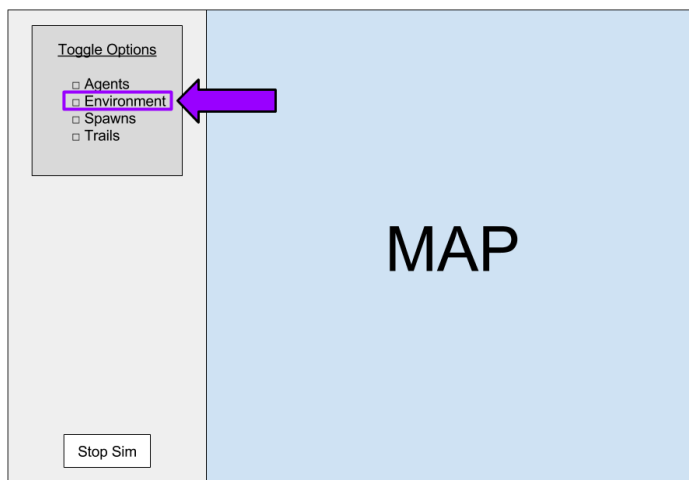
For the simulation stage we chose to use toggle boxes because they provide an efficient means to switch between modes rather than using buttons or other methods.

11.3.1 Agents Toggle



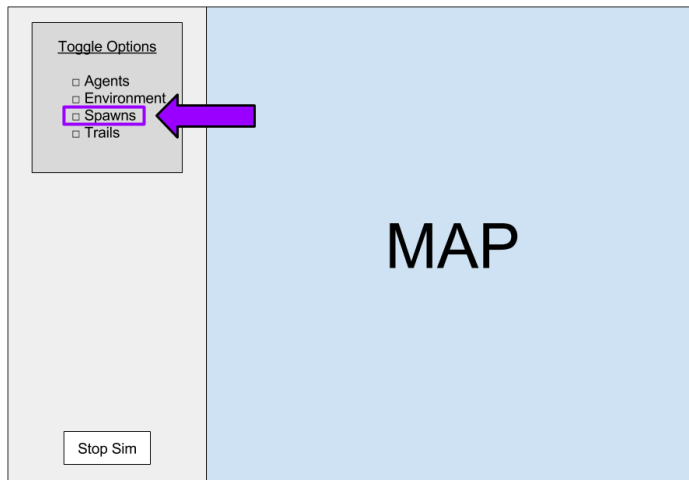
- Controls whether or not the user can see the active agents.

11.3.2 Environment Toggle



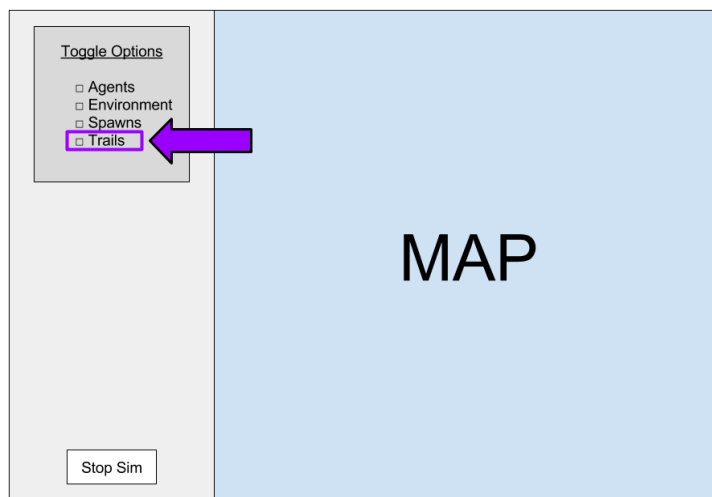
- Controls whether or not the user can see the map.

11.3.3 Spawns Toggle



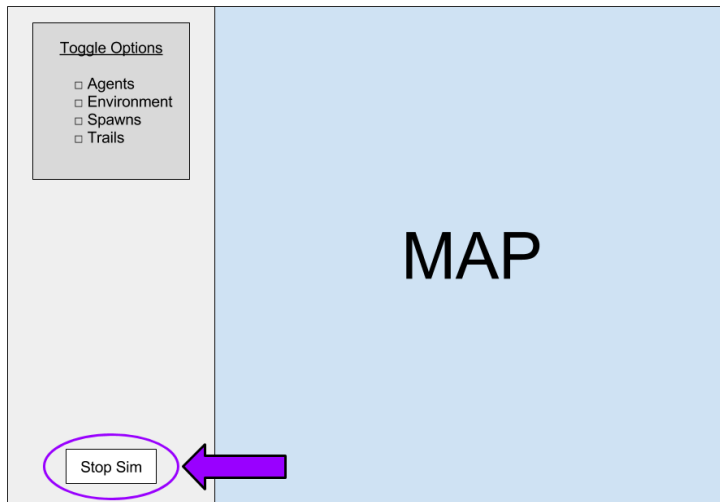
- Controls whether or not the user can see what locations agents have spawned from.

11.3.4 Trails Toggle



- Controls whether or not the user can see the movement of agents.

11.3.5 Stop Sim Button



- Allows user to stop the simulation.
- We chose to use a button for this function instead of other methods (e.g. hotkeys) because we felt that buttons would be better for the clean and simple interface we want to achieve.

12 Other

Details that are not specifically stated in this document are left to the concern of the software development team. Details are subject to change and these changes will be shown this section of the document.

12.1 Appendix A: Glossary

Agent – A single unit of workers that can be used to help in aid.

Decision – The most beneficial option for an agent to take when considering damage, aid and value in a certain region.

Graphical User Interface (GUI) -- Includes graphical elements such as windows, icons, and buttons.

Open Source Map (OSM) - An open source map format that follows a similar structure to XML.

Simple and Fast Multimedia Library (SFML) - An application programming interface designed to aid with multimedia components.