The University of Sheffield

City College

Final Year Project

Simulation of Crowd Emergency Evacuation Using Emotional Agents

---

This report is submitted in partial fulfillment of the requirement for the degree of Bachelor of Science in Computer Science by

Argyrios Liatsis

February 2019

Approved

Prof. Petros Kefalas

Declaration

"All sentences or passages quoted in this dissertation from other people's work have been specifically acknowledged by clear cross-referencing to author, work and page(s). I understand that failure to do so amounts to plagiarism and will be considered grounds to failure in this dissertation and the degree examination as a whole"

Name: Argyrios Liatsis

*Signature*____________________

*Date*_______________________

Abstract

Evacuation of hazardous areas is a serious concern for coordinators, event hosts and authorities. In order to facilitate the creation and implementation of possible solutions to this problem, this project tackles it by making use of Artificial Intelligence techniques, specifically Multi-agent systems to generate an evacuation simulation model. The simulation tool selected for this purpose is NetLogo. This project will provide aid in better understanding of the behavior of humans in distress under hazardous conditions.

Table of Contents

Table of Figures

1. Introduction

Ever since people started congregating in enclosed spaces, there existed the need to swiftly evacuate them in cases where the activities deviated from the planned ones, such as emergencies [1]. To elaborate, various hazardous events threatening the well-being of humans in enclosed spaces is a serious threat that exists today and the means to mitigate the damage nowadays are more plentiful [2]. The means that this project focuses on is creating an agent-based model that simulates the evacuation of personnel and/or civilians in enclosed spaces and how their emotional state affects the individuals in their proximity. Creating the tools themselves to solve such a problem is a long-term, strenuous and challenging tasks, however by utilizing the existing tools the desired simulation that comprises our system becomes a more flexible and agile task. On the following subsection, the aim, as well as the definite objectives of the project are stated and explained.

1.1 Aim

The aim of this project is to utilize software simulation tools in order to achieve creating a multi-agent system, comprised of intelligent software agents which interact with each other. To elaborate on that, the agents should be able to express the ability to "infect" other agents that are in close proximity to them (within the boundaries of the system) with contagious emotions.

Consequently, the aforementioned agents should possess the ability to mimic emotions that are expressed by actual human beings in the "real" world. Such a system could be used by specialists in other fields to study the behavior of humans in alerted states of mind and apply those findings in various ways, such as designing the indoors of said congregation rooms; theaters, cinemas, lecture halls, etc.

Ultimately, the aim is to develop a system that fully simulates the conditions that ensue in a hazardous area populated with individuals of varying backgrounds, personalities, moods and emotional states.

1.2 Objectives

The objectives that must be completed in order to achieve the aim of this project are presented as bullet points below:

1. Understand how emotions, mood and personality of human beings affect their expressed behavior at moments when their lives are endangered (panic, awareness etc.)

2. Understand how Multi-agent Systems work, to a level adequate for the completion of the project

3. Achieve a moderate-to-high level of expertise with the NetLogo simulation platform

4. Apply the knowledge gained from objective No.1 in a multi-agent evacuation simulation model by making use of the NetLogo simulation tool.

5. Provide a simulation demonstration of the final software implementation showcasing the desired features

The objectives that shall assist the aim of the project are adequately straightforward: In order to formulate the desired system, a set of behaviors is required to be developed.

The aforementioned behaviors is the main component that creates the interactions between the simulated individuals of the system. Detailed information is presented in Chapter 4 about the system itself. The system should have the capability of reflecting the way emotional states such as panic would spread in case some unexpected, life-threatening scenario occurs in confined spaces, such as night clubs, auditoriums, theaters, etc.

2. Literature Review & Background

This section of the project aims to break down and differentiate the cognitive science aspects of this project and ultimately define, explain and present the information and techniques requires for the completion of this project.

2.1 Multi-Agent Systems

There exist various definitions regarding what a Multi-agent System is (hereby referred to as MAS), however the definition that is explained in [4] is better suited in this case. However, in order to identify what a MAS is, the definition of what Artificial Intelligence has to be provided first.

Artificial intelligence (shortened as AI henceforth), as a term, was coined by John McCarthy circa 1956 [5]. It is the field of computer science associated with developing intelligent, human-based information technology systems. AI is steadily growing to be a fundamental component in the technology industry, providing solutions for complex problems, such as, but not limited to:

- Pattern recognition, e.g. market trend detection and weather prediction
- Speech recognition, e.g. providing support for human-computer interaction (HCI)
- Intelligent control, e.g. providing intelligent and adaptive control over intricate processes [5]
- Machine learning, e.g. multilingual question classification (What is the greatest city in Germany?)(Was ist die größte Stadt Deutschlands?) [17]
- Computer vision, e.g. real-time face tracking for security purposes [17]

MAS is an archetype that derives from a number of disciplines, namely distributed artificial intelligence (DAI) and study of groups of living organisms by utilizing artificial life [5].

To elaborate further, MAS is the emerging subfield of AI that its goal is to provide principles for constructing complex systems involving multiple agents and coordination independent agent's behavior by utilizing elaborate mechanisms [22].

2.2 Examples

The possibilities and opportunities for manufacturing MAS are endless; virtually anything in the observable universe has the potential to be transformed into a MAS, for example how the nodes of a Dyson Sphere communicate with each other and/or propagate the absorbed energy from a solar system's star [7].

The system could work as follows: A leading drone with higher capabilities than its counterparts approaches the target star and sets up a virtual network based on pre-existing schematics. Following up, a swarm of lesser drones would start coordinating with the rest in a communication network in order to complete the formation set up by the leading drone, while executing any anti-collision protocol(s) that they are possibly integrated with.

Another promising example of a MAS could be a school of fish and their behavior as one unit. To be more precise, the breeding sequence that red handfish (*Thymichthys politus*) follow in order to ensure their species longevity [8]. This species of fish is relatively new and yet to be studied thoroughly, however it has been observed that they require a relatively rare plant to lay their eggs on to reproduce [8]. A MAS representation could be the joint collaboration of red handfish in search of the aforementioned breeding grounds.

2.3 Agents

There exist no general consensus regarding the definition of an agent, yet in [20] an agent is supposed to be any entity that can be viewed as perceiving its environment through sensors and acting upon that environment based on built-in mechanisms and/or actuators. For instance, a human agent has ears, eyes and other sensors that perceive their environment and legs, reflexes and so on that act as their actuators[20]. Essentially, an agent's action chosen at a particular point in time can depend on their entire percept sequence observed up to the present, but not on any non-perceived input[20]. An agent's percept sequence is the total history of everything the agent has ever perceived. Figure #1 visually represents these fundamentals.

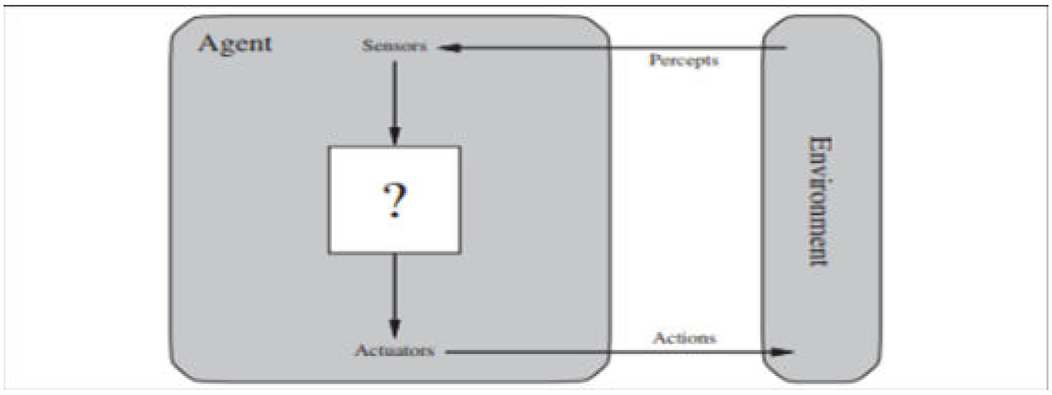


Figure #1. A basic agent representation [20]

Basically, the described agent, as well as the figure presented above give us the definition of an agent on its simplest form, that of the **reflex agent**.

2.4 Goal-Based Agents

Agents that have knowledge regarding the current state of the environment are not always efficient enough[20] For instance, a train can either use a rails junction or continue on the established path. The correct decision depends on the location the train plans to travel to. Goal-based agents are the main type of intelligent agents that exist and are the closest representation of autonomous, decision-making being, such as humans or animals. They consider various scenarios before acting based on their set behavior, as well as data drawn by their environment to achieve their set goals. Even though they may seemingly lack in efficiency compared to their reflex-agent counterparts, they are more flexible. For example, a train's destination can be changed on the go, and the train's driver can

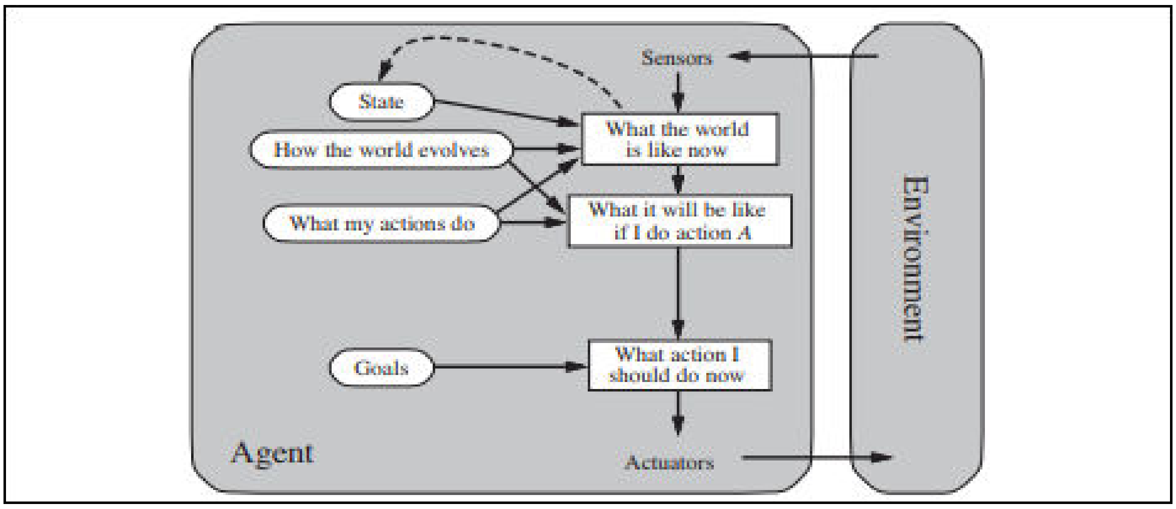

Sensors
State
How the world evolves
What the world is like now
What my actions do
What it will be like if I do action A
Goals
What action I should do now
Agent
Actuators
Environment

Figure #2. Visualization of a Goal-based agent's functions[20]

More types of agents exist, however they are out of the scope of this project, thus they are not referenced in this project.

2.5 Emotional Agents

Derived from the interdisciplinary field of cognitive science, namely psychology, this subsection focuses on emotional integration into intelligent agents.

There are various factors that directly affect agents that comprise a MAS, and therefore individuals in real life. These include, but are not limited to, factors such as mood, personality, appraisal, emotion contagion, etc. [13][15]. In a sense, from a computer science perspective, modeling agents and transforming them into emotional agents adds a layer of heterogeneity[22].

In order to continue further, brief descriptions of what defines an agent as an "emotional" agents are provided below:

Personality. An individual's personality can be derived by the distinct values of the Big Five[23]basic factors that affect personality (Openness, Consciousness, Extraversion, Neurotism, Agreeableness). For the purpose of the system, the personality characteristic is a set value in the initialization of the agent and does not change by any means.

Moods

Moods are often erroneously interpreted as emotions, however they differ in the sense that they are weaker, more diffuse affective conditions that last over long periods of time. Moods and their effect on individuals can play a major role in the outcome of an evacuation event. Figures #3 and #4 depict how particular hours of the day and particular days of the week affect the moods of individuals, which are based on research done by [19]:

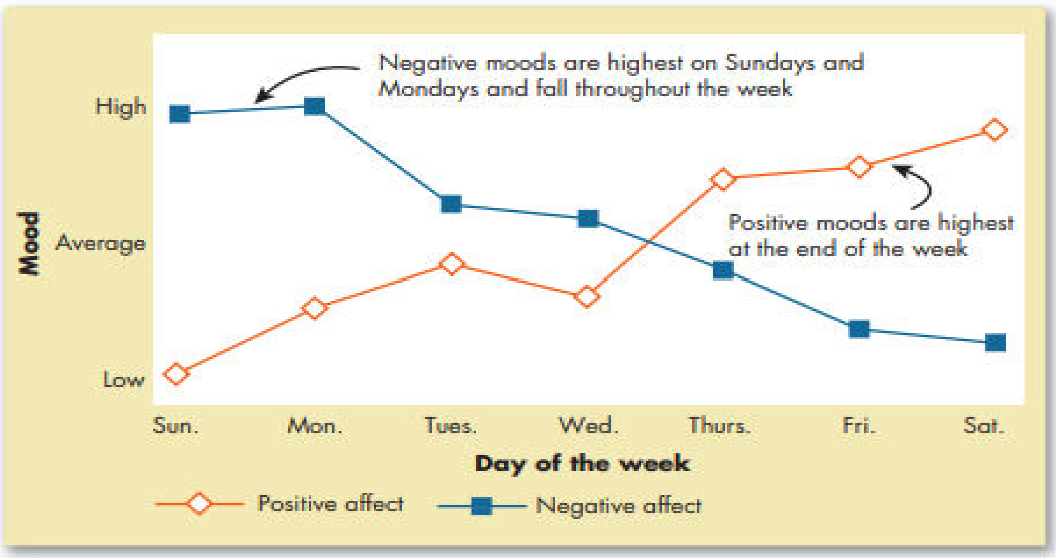


Figure #3. Mood variance during the week [19]

As it is evident in figure #3, the average mood positivity rises as the week reaches its end, and suffers a huge decline at the beginning of a new week. The opposite holds for negative mood.

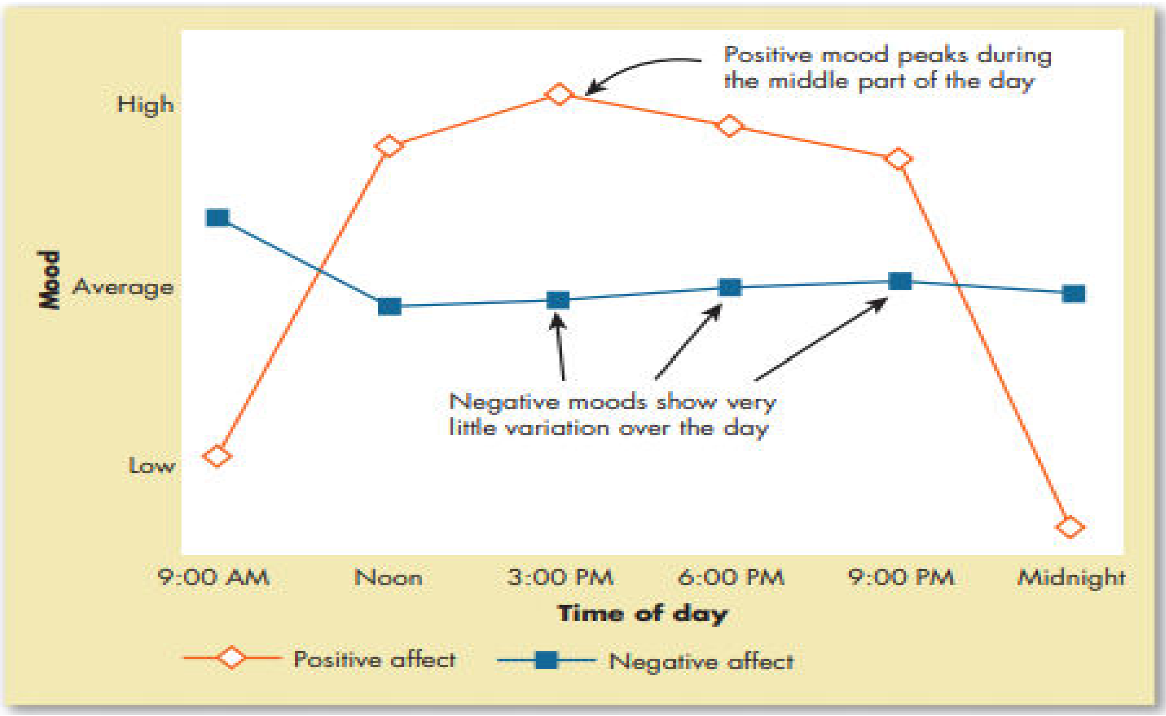


Figure #4. Mood variance during the day [19]

As it is shown in figure #4, positive mood receives a great boost throughout the day, reaching peak values and maintaining them for approximately nine hours before declining rapidly. However, negative mood does not show the same variance.

2.6 Emotions

Emotions, by definition, are intense, short-term bursts of affective reactions to specific environmental stimuli [9].Mimicking and/or inducing emotions in intelligent agents has proven to be a challenging task over the course of time due to the high complexity such a task entails.

- *valence.* Represents the (x) axis of how enjoyable it is for an agent to experience a particular state

- *arousal.* Represents the (y) axis of the magnitude of the experience an agent is undergoing

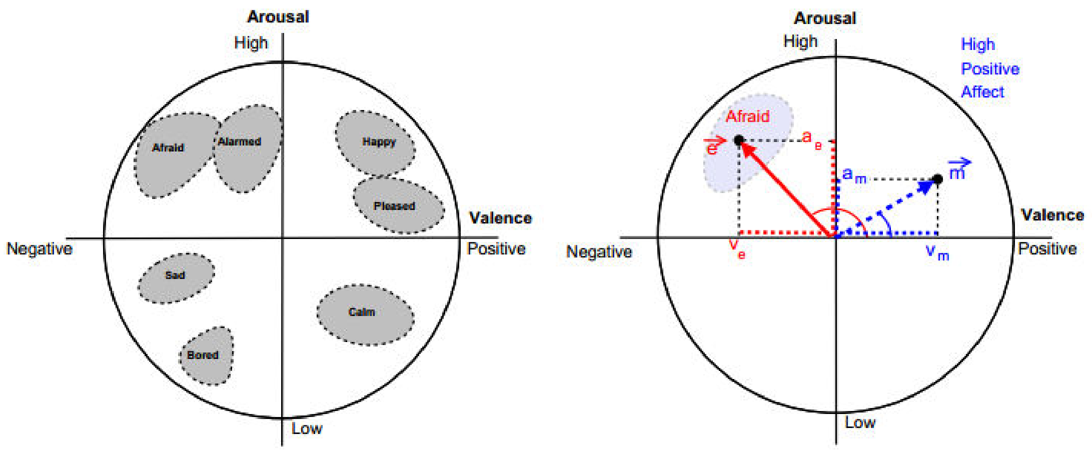


Figure #5. Circumplex of affect representation [9]

The circumplex of affect is a widely accepted model of defining long-term mood of individuals, as well as short term emotional states. The greater the *valence* value(positivity), the less likely an agent is to affect others in their proximity, while the lower value(negativity), the more likely it is for an agent to be affected and have their emotional state altered [13][15].

Likewise, the greater the value of *arousal*(hyperactivity), the greater the likelihood of the agent to act/react more vividly, as opposed to lower value of arousal(hypoactivity) that promotes lethargic actions, or no actions whatsoever in more extreme cases.

2.7 Emotional Contagion in Agents

Emotional contagion is the impact i. Furthermore, emotional contagion is a form of perception since the agents in question perceive emotional stimuli. When two agents of varying state interact, the one with the lower value for "fear" emotion instantly inherits the neighbor's higher value, changing its state and ultimately achieving **emotional contagion**. A schematic representation follows below:

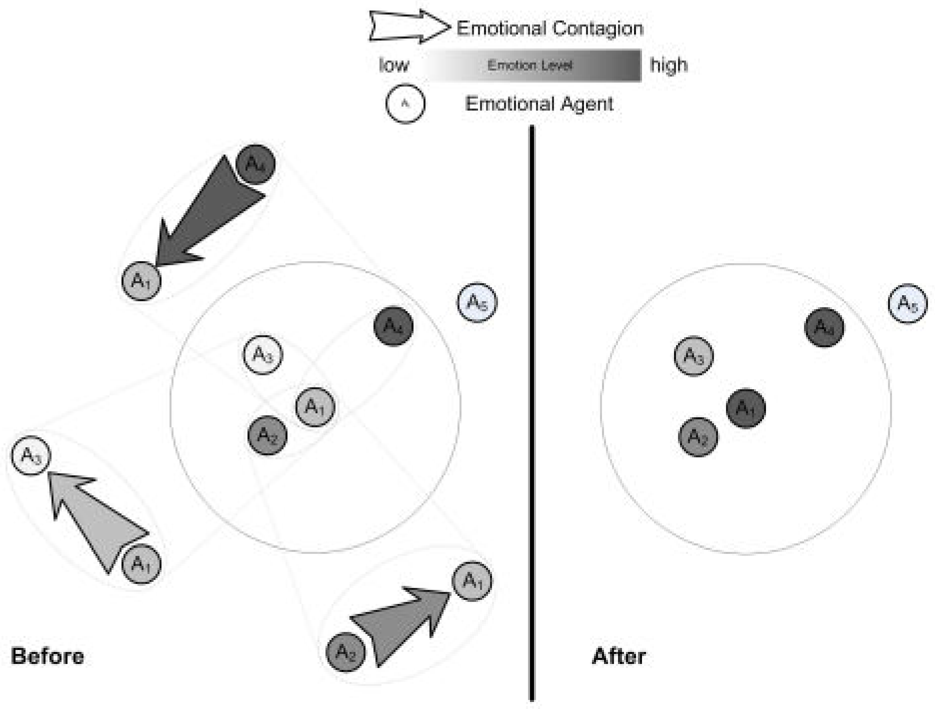


Figure #6. Emotional Contagion Representation [16]

Agents A2, A3 and A4 possess different values of the same emotion. Taking into consideration the fact that the darker the shades, of grey, the greater the value of the corresponding emotion, agent A1 shall eventually inherit the level of emotion of Agent A4

In addition to the above, another important factor regarding emotional contagion is the number of agents existing in a particular scenario, as well as the existence (or lack thereof) of authoritative figures. In the example of figure #7, it is apparent that the greater the number of authoritative figures, the lesser the number of agents that succumb to emotional states with negative valence values [16].

In the experiment derived from [16], the propagated emotion in this scenario is fear, and it is represented in various escalating levels: Calm, Alarm, Fear, Terror, Panic, Hysteria.

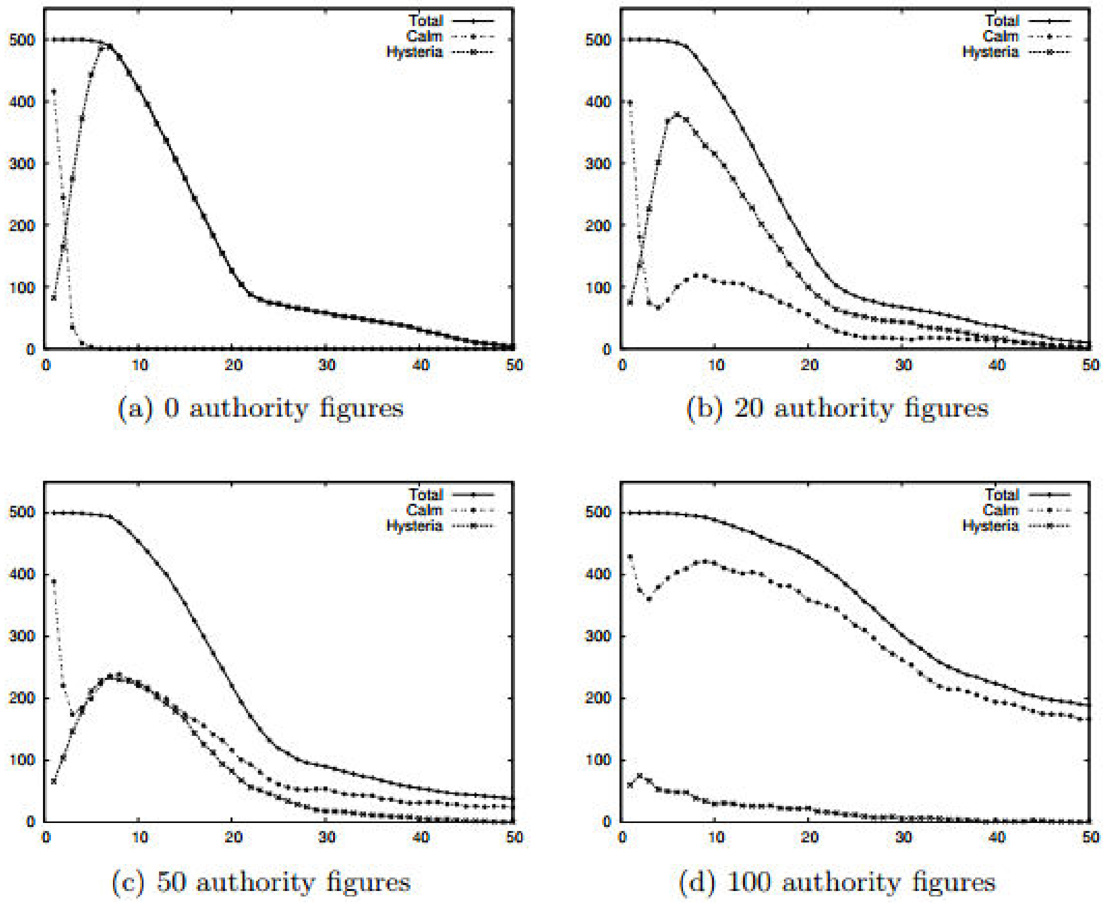


Figure #7. Simulation results of four test scenarios of "Evacuation_SXM_Emotions" Netlogo model developed by Dr. Sakellariou [16][29]

The experiment above is showcased to provide an approximation on how the final state of statistical output the desired system of this project should provide.

By the end of this chapter the first objective has been accomplished. Furthermore, transforming the principles explained in the chapter above into a software system proved to be challenging, although not extremely difficult. By taking into account the facts presented on figure #6, emotions are transmitted in contact with each other agent, thus providing assistance at figuring out a way in which to conceive a system that mimics emotional contagion.

3. Methodology & Project Management

Projects require delicate handling in the software development process, and in order to ensure that a set schedule is followed across the semester, a series of techniques regarding project management is introduced. The first one is "Project Simplification and Iterations", which describes the project break down process followed to ensure that the chosen software development methodology is followed. The following and last one is "Risk Management", which presents the potential risks and probabilities these risks might occur during the course of the project.

3.1 Software Development Method

Selecting a software development approach did not prove to be a challenging task due to the simplicity of the project's requirements, the nature of the project itself, and last, but not least, the instructions received by the project's supervisor. Initially, the Iterative Development process was considered due to its flexible, segmented, rapid nature. Each iteration deliverable is essentially a miniscule waterfall model cycle, complete with analysis/design/development cycles [3][10].

The advantages of such an approach are increased flexibility compared to the old-and-tested, compact Waterfall, and the ability to selectively create parts of a project fit for specific individuals. Moreover, it offers adaptability in the case of requirements changing during the course of the project, regardless of the magnitude [3][10].

However, after reconsidering and reflecting on the problem of development process, the iterative approach proved to be inadequate in this case. In this project, it was deemed more appropriate to utilize the incremental method. The reason behind this is the addition of further behavioral patterns in the already pre-existing "emotionless" agents.

The incremental approach is a solid, robust solution. To elaborate further on it, however, there is a requirement to decide on a specific model. The model that was taken into consideration was the Rapid Application Development model (RAD), as it is depicted in Figure #8.

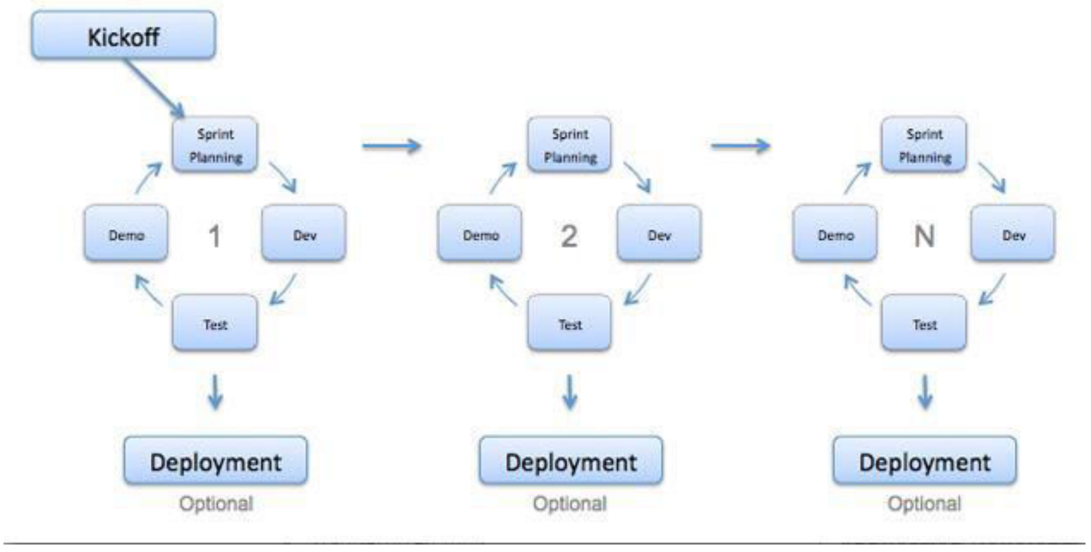


Figure #8. Rapid Development Model [11]

In conclusion, after careful consideration of the drawbacks and advantages of the presented methods, the RAD approach was selected due to the striking affinity to the nature of the project, as it is depicted on figure #8.

3.2 Experimentation

In this section, a short description regarding early experimentation with the NetLogo tool is presented.

For experimental purposes, a way to designate "safe" regions within the area was perceived by coloring certain patches so the agents can perceive them as "safe" zones by mousing over desired locations. The agents should not be able to cross these patches. However this experimentation scenario would be pointless if this case was not reflecting a real-life scenario. In our case, this could represent a scenario where there are extreme weather conditions in the area, such as snowstorms, windstorms or typhoons.

Figure #9 shows a system state where the "evacuated" agents are within the safe zones:

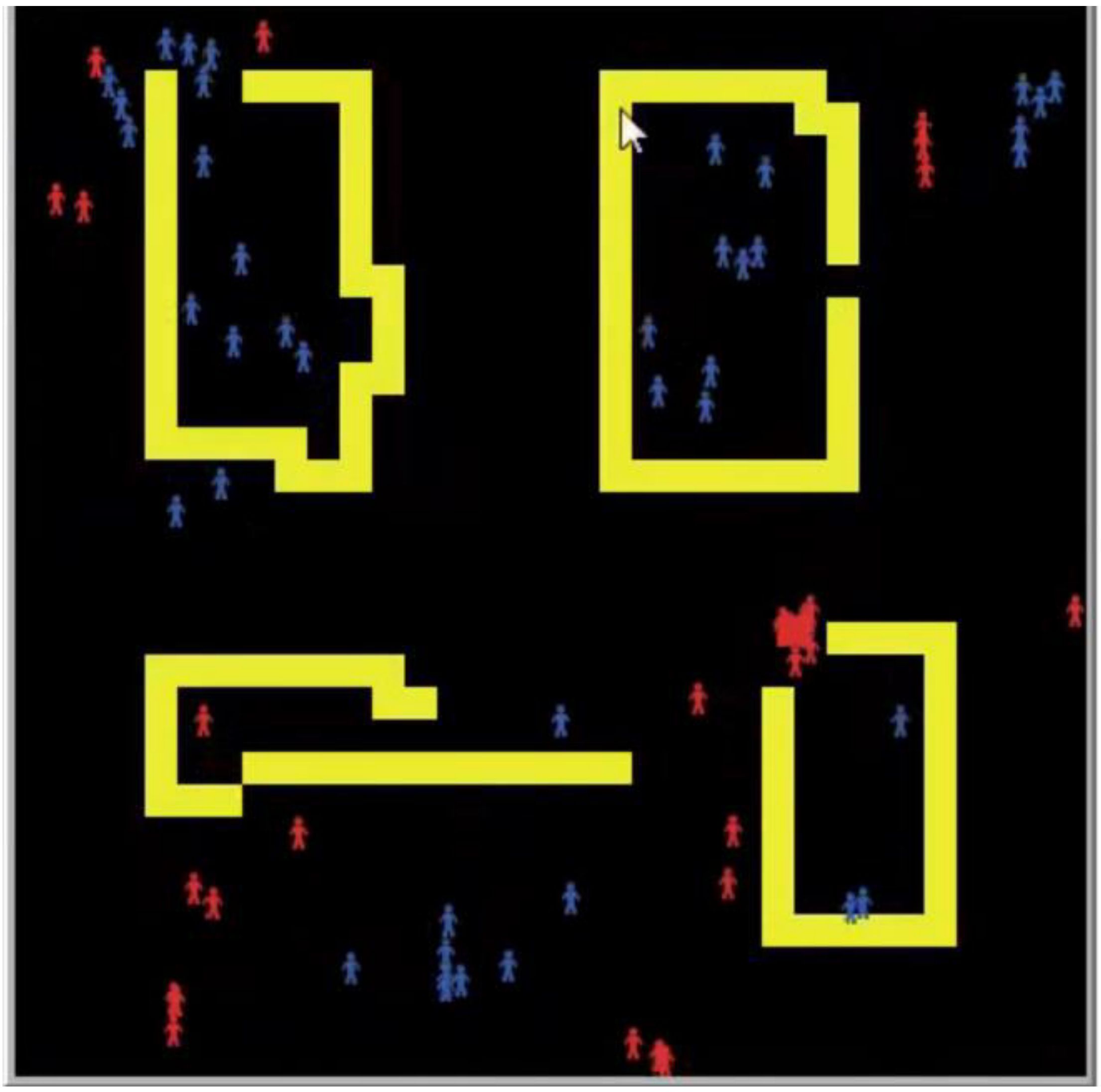


Figure #9. Experimenting with Netlogo

This early experimental model by no means implements emotion and emotional contagion, and it is not error-proof per se, as there are many agents who failed to evacuate and there is no behavior programmed to turn an agent from "panicked" to "non-panicked" state.

The practice showcased in figure #10 is part of the software implementation approach that was chosen for this project. It showcases a very early execution of the expected model as it is described on Chapter 4.3.

3.3 Project Simplification & Increments

Based on the chosen software development methodology, the software implementation phase was broken down to a series of iterations, presented on the table that follows. As it has already been mentioned on chapter 3.1, the selected methodology follows the principle of separating the implementation process to lesser "pieces", called "increments". Table #1 presents the project simplification and the individuals deliverables in an incremental-process-like fashion:

Increment No.	Description	Estimated Time Duration	Estimated Complexity	GUI IMPLEMENTATION
1.	Implementation of Exit Search Behavior	1-2 week(s)	Low	
2.	Implementation of Smart Exit Search Behavior	1-2 week(s)	Average to High	
3.	Addition of Emotional Behavior in the pre-existing system	2-4 weeks(s)	Average to High	
4.	Implementation of Emotional Contagion	2-4 week(s)	Average to High	

Table #1. Project Planning

An extra notice has to be denoted on Table #1: The Graphical User Implementation is a rather simplified process in this project, since it merely consists of a few minor additions; that require little to no code whatsoever. These buttons may or may not contain part(s) of the behavior as requested in the system requirements. Even if this not the case, that is, the buttons' appearance itself does not interfere with the system logic, they are still subject to minor changes, thus the appropriate representation in the table.

3.4 Risk Mitigation

Risks are always present when attempting to initiate a project of high caliber. If overlooked, risks can be a deciding, yet devastating factor In this section potential risks are presented and placed into tables for convenience. To elaborate, risks are always present regardless whether the execution of each step was executed in an almost perfect fashion or not. In order to ensure that potential risks are faced and mitigated, suggested mitigation strategies are listed in the table that follows:

Risk No.	Name	Description	Probability	Impact	Mitigation Strategy
1.	Misunderstanding User Requirements	The developer ignores and/or misunderstands the requirements of the system	Low	Average to High	Be in close contact with the project supervisor and consult them in a regular basis
2.	Underestimating Project Complexity	The developer believes that the project is bound to be easy	Low to Average	Catastrophic	Review the project requirements and analyze them extensively/Consult the supervisor
3.	Non-adherence to time constraints	The developer shows poor time-management skills	Average	High	Consult an expert and/or create an effective time schedule and adhere to it
4.	Change of User Requirements	During the implementation phase the requirements are (partially) changed	Extremely Low	Average to High	Stick to the Rapid Development Methodology (RAD), but alter the project plan accordingly
5.	Inappropriate and/or lacking testing and evaluation methods	During the testing and evaluation phase the developer skips or handles the entire process in a poor fashion	Low	Average	Adhere to known and effective testing techniques and evaluating methods and strictly follow the project schedule

Table #2.Risk Mitigation

By the end of this chapter, the project planning process was established. Furthermore, it also partially satisfied objective #2, that is, understanding how multi-agents system works. The next chapter focuses on creating the software schematics for laying the foundation of the NetLogo model implementation.

4. Software Analysis & Design

The software analysis and design process of a software project is very vital for delivering quality software projects that are robust, as-bug-free-as-possible, and easy to maintain.

The following section contains the software analysis and design techniques that were deemed appropriate to be utilized for this system.

4.1 Analysis

4.1.1 User Requirements

The first step towards laying the foundations of a software project is to conduct thorough software analysis on the desired system. The first step to be considered is an interview of the client in order to identify the system's requirement. In this case, the client is Prof. Petros Kefalas. During a face-to-face meeting, the project details were discussed to an extent in order to identify the requirements. The discussion included the presentation of the NetLogo model "Evacuation_SXM_Emotions" designed and developed by Dr. Ilias Sakelariou [29]. The request was to create a similar model, albeit less sophisticated. As it occurs with Dr Sakelariou's model, the agents are supposed to evacuate a pre-determined area and reach a destination designated as "exit". What makes the model complex is the interactions between the agents. Moreover, the environmental hazards also play a great role towards the goal the agents are given to accomplish. With this information taken into consideration, the foundations were laid for introducing the system description that is presented below.

4.2 Model Description

This subsection briefly describes the initial conception of the desired mode as its description was derived by the user requirements. The model description of the final software product is presented further below.

The model is initialized in a set of pre-defined locations with pre-defined obstacles that can be set at one's leisure. The number of agents can be selected as desired. The expected results/outputs from our system should be visual representations of how the agents' emotional state affects their evacuation process.

All agents should have three randomized, pre-defined values; namely personality, mood and emotional state.

There are two agent sets in this scenario:

- Panicked
- Non-panicked

Agents in this model are represented with a color scheme in order to be visually differentiated. Calm agents are represented as green, alerted agents are of yellow color, and last, but not least, panicked agents appear as red. Essentially, the agents created for the purpose of this experiment belong in the category of **model-based reflex agents**[20].

Each agent should have a randomized personality which is not affected or changed during execution of the evacuation event. Furthermore, every agent is initialized with a randomized mood factor that also does not change in short-term fashion. Since the focus of this project is not long-term observation of mood change, the mood factor and its possibility of change over the course of time is non-essential.

Various events (earthquake, flood, fire) occurring in the area of interest should have a different hazard value, thus a more dangerous type of catastrophe should have a greater impact on the emotional state of agents in the vicinity.

The role of mood should only affect how the *initial* emotional contagion is accomplished. After an act of emotional contagion has occurred, the mood factor enters a state of regression and then the main factors that shall dictate how the agent behaves are their personality and emotional state, depending on the influence range.

Also, introducing a time limit for experimentation in evacuation scenarios like this should be considered. After the time limit expires, agent(s) who fail to evacuate within zones that are considered "safe", are eventually considered terminated.

4.3 Final Model Description Differences

Since the introduction of the initial model description, the final portrayal underwent considerable changes, however the core idea remained the same. One of the drastic changes that took place was the shift from a pre-determined field in which the agents are supposed to operate. In this case, the user is given the freedom to design the world that the agents operate in on their own. The reason behind this decision was to diversify the model from pre-existing models, offering something unique in the process, which is increased flexibility. Furthermore, the concept of agent personality was scraped entirely in favor of mood and emotional state. The reason behind this decision was to simplify the system, as to compensate for the new, arbitrary world generation capability of the system. Moreover, as the only two agent sets representing emotional states were deemed inadequate, the "non-panicked" agent set was split into "alerted" and "calm". Last, but not least, the environmental hazard feature was scrapped altogether as it was deemed unnecessary towards the completion of this project. More details about scrapped ideas and features can be found in Chapter 7, subsection "Future Work".

The decision to include both the initial and the final system description in the final report was deemed a valid practice, as to assist with both the "Self-Evaluation" and "Future Work" subsections included in Chapter 7.

4.4 Design

The purpose of the analysis procedure is to identify *what* needs to be implemented, while the design procedure focuses on *how* the desired system is going to be implemented [25].

4.4.1 System as Finite State Machine

A Finite State Machine (hereby referred to as FSM) is a well-established model in the formal specification domain of Computer Science [24]. A FSM is composed of a certain number of states, which when grouped together form a set of states. These states are linked by transitions. Every time the software logic shifts state (ex. analyzing user input -> storing user input), this "shift" is depicted as a transition[24]. Usually FSMs are initiated by some sort of input, however they don't necessarily produce an output. Figure #10 showcases FSMs' most basic components that are used to create FSM models.

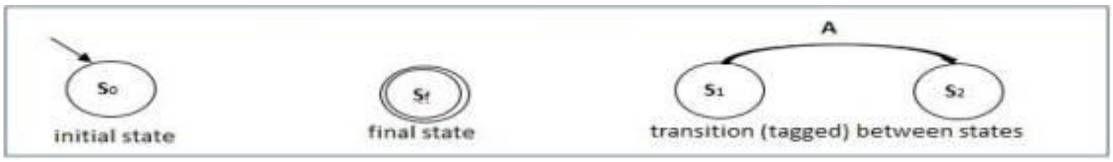


Figure #10. Representation of FSMs' most basic components [24]

FSMs follow basic patterns which exist as a means of formalizing. Those patterns are called workflows. There exist plenty of workflows to demonstrate how a task ends up from one state to another, however for this project's needs, only two are required to be listed. These are:

Sequence: A workflow that is defined as a series of activities/states, its form is of parallel states linked together by one transition. For instance, let there be three states: A, B, and C. B cannot start its execution before A finishes its own, C cannot execute before B, and so on. The example is listed in figure 11.

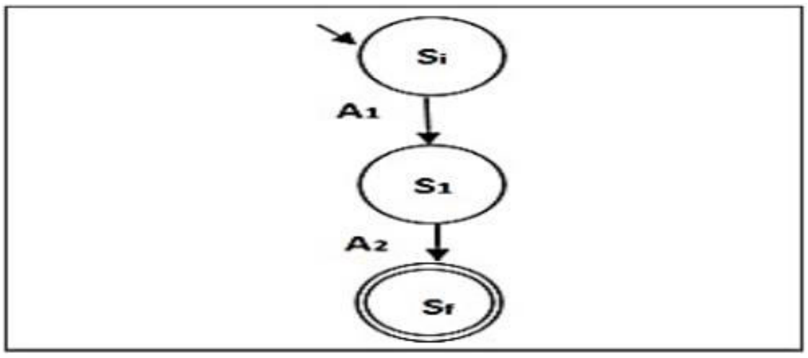


Figure #11. A basic sequence in a FSM model [24]

Parallel Split: is defined when two or more states of a FSM begin their execution at the same time. It is a point when the flow of a FSM is split into multiple control flows that are executed in parallel [24]. The principle of parallel split is showcased in figure #12.

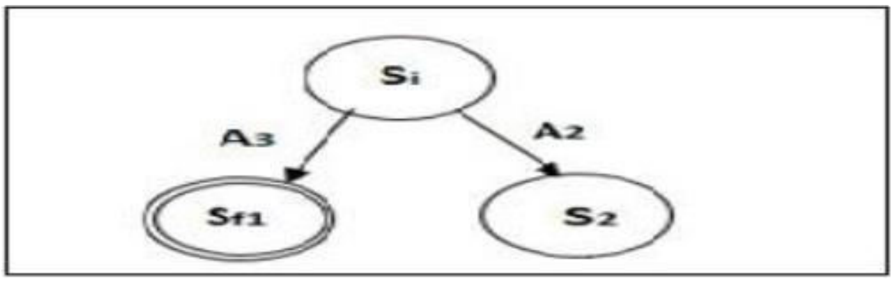


Figure #12. Most basic form of a parallel split in a FSM model[24]

By taking these two principles of FSMs and applying them to the system specifications that were explained in Chapter 4.1, the FSM model in figure #13 was established.

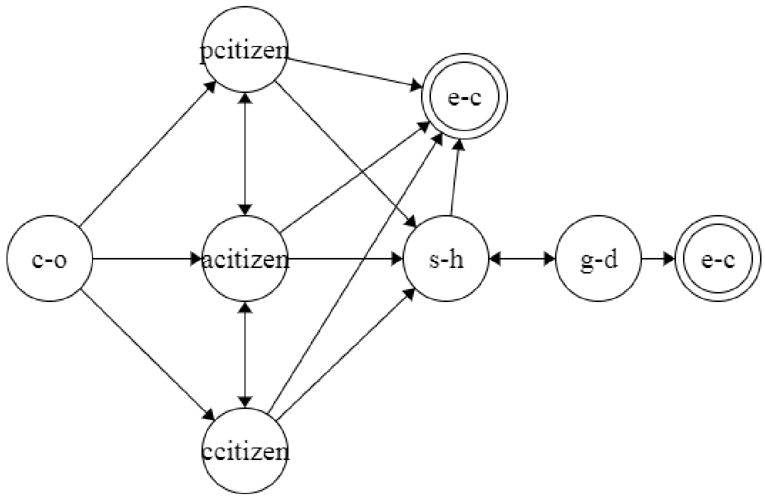


Figure #13.The desired system depicted as a FSM model

"C-o" is the initial state of the system, which stands for "Catastrophe-occurs", homage to the original plan of including environmental catastrophes and/or hazards in the model. It is supposed to transition the state of the agents as "pcitizen", "acitizen" and "ccitizen", which stand for "panicked Citizen", "alerted citizen", and "calm citizen" respectfully. As it is shown in figure 11, the states in which the "citizens" are interchangeable, hence the transitions between them. Following that, the next transition that an agent may proceed to is either "e-c", which stands for "evacuate", or "s-h", which stands for "seek-help". During the "seek-help" state, the agent is searching for either an exit or someone who can help them. In case it does not manage to escape on its own, and in case it encounters a figure of authority, it initiates the state "g-d", that is, "get-directions". This state provides instructions to the agent to locate the exit, thus facilitating the escape. Getting the directions to evacuate, however, does not imply that an agent will understand those. There may need to be clarification, and the agent may seek another figure of authority, hence the dual-arrow transition between "seek-help" and "get-directions". To conclude, meeting a figure of authority is not mandatory in order for the agent to evacuate, thus completing its goal.

4.4.2 User Interface

User Interface (therefore referred to as UI) designing is the process of defining the interaction of external entities with the system [25]. The vast majority of UI designing includes three fundamental parts: First and foremost is the navigation mechanism, which is responsible for instruction and data input in general. Second is the input mechanism, the way in which the system is fed with information to process(such as forms, etc.). Last, but not least there is the output mechanism that provides the user and other systems with processed information [25]. These three parts are closely related, in fact, the first two parts may seem to be identical. Most programming languages allow computer scientists to design their own user interfaces from scratch, however NetLogo already has predefined UI elements, thus limiting the options at our disposal considerably. There may exist ways to exceed those limitations, however such endeavors are out of the scope of this project. An example of how the UI of a NetLogo model looks like is presented on Chapter 5.2.

The end of this chapter satisfies the conditions for turning the user requirements and ideas into software schematics. The next chapter zeroes in on implementing the FSM diagram's logic (showcased in Figure #13) into NetLogo code.

5. Implementation

This chapter focuses on implementing the requested system by utilizing the agent-based modeling language NetLogo[1]. The subsections included in this chapter are the "Programming Tools", a short summary and justification about selecting NetLogo as the programming language/environment, and "The Model" itself, in which the model is featured as code snippets. To elaborate, the model's logic is explained through every listed code module, with the aid of screenshots. Due to the nature of the programming environment offered by NetLogo, every code module is also accompanied by its GUI counterpart that is directly assigned to. This practice effectively provides insight on the workings of both the inside and the outside of the model, assisted by a screenshot of the complete interface of the model in the end of the chapter.

Disclaimer: Experimental, unused pieces of code still remain within the implemented model. They could be used in attempts of future work. Those scraps of code have no influence on the model whatsoever and can be completely ignored.

The system description derived by the user requirements, along with the state machine diagram were the main foundation of the implementation process of this project. Most states of the diagram directly translate to NetLogo procedures, as it was explained on Chapter 4, "Design" subsection.

[1] https://ccl.northwestern.edu/netlogo/

5.1 Programming Tools

During the discussion regarding the implementation of the desired system, the topic regarding the tools to be utilized was discussed. Initially, there existed two major approaches: The NetLogo approach and the RePAST approach. In addition to that, the possibility to explore more innovative, less established approaches also existed, however the risk of facing varied ranges of lack of documentation and community support was higher, so this path was omitted.

Every tool that provides the ability to reproduce executions of models in order to generate their behavior is a simulator, and all computational systems that act as tools for both educational and research purposes are considered to be simulators as well. Simulators can take the form of a single computer or a multitude of systems working for the purpose of simulating models [4].

Both NetLogo and RePAST[2] are tools that belong in the category of agent-based modeling simulators. However, NetLogo goes a step further by introducing its own programming language, complete with a wide set of libraries, among which an extensive models library which itself is derived from the Logo language [5][6]. NetLogo itself, however is written in Java [5]. Since 2008, all the updates focused on improving the NetLogo language have been written in Scala [27]. The user's code is not interpreted directly; instead the compiler analyzes, annotates, and restructures it into a more efficient form.

[2] https://repast.github.io

5.2 NetLogo

Eventually, NetLogo was the chosen tool to employ due to the remarkable ease of use it offers, as well as the simplicity and superior availability of documentation and tutorials online [21]. A figure showcasing the environment of NetLogo is presented below:

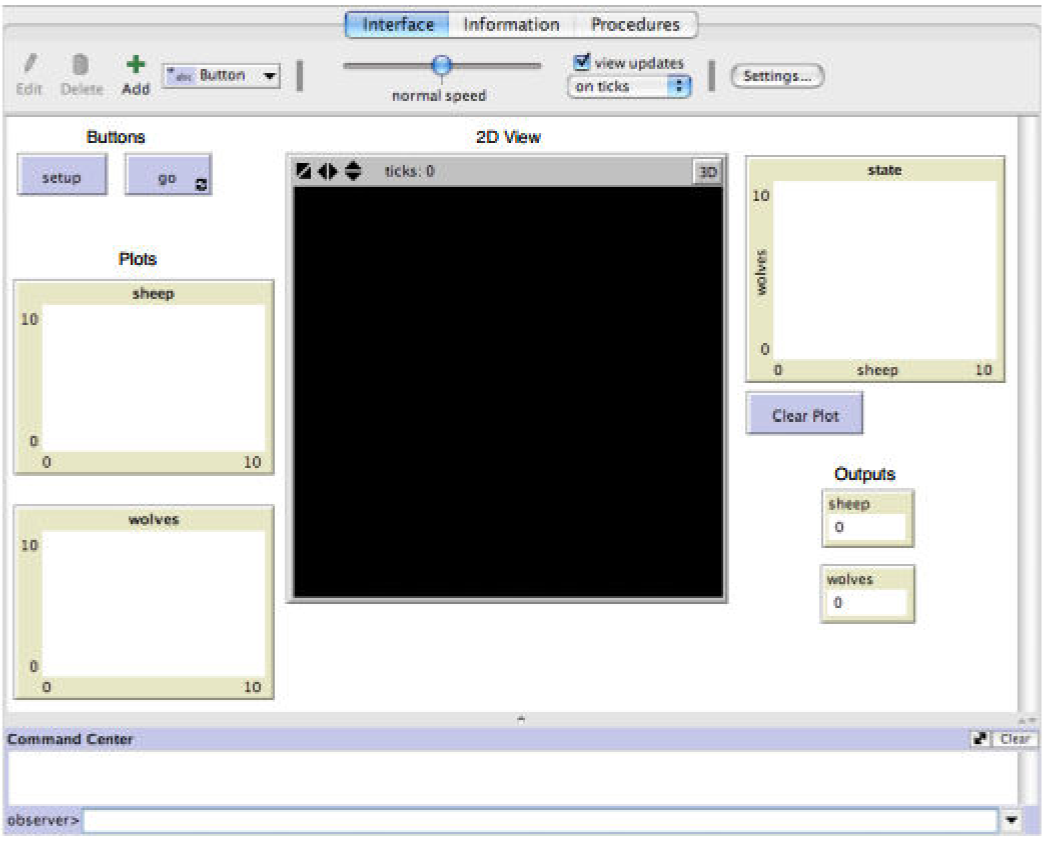


Figure #14. The interface of NetLogo [21]

Regarding the ease of use, apart from offering visual results of the executed code, NetLogo integrates a built-in models library, other more simplistic while others being more complex and advanced, with more than 150 models present that simulate various cases [6]. These models include a description regarding their subject, as well as the simulation rules and suggestions for activities, possible extensions and further experimentation. These models can either be used as a basis for creating new models based on the user's needs, or can be used as reference material for educational purposes.

In addition to the plethora of models that are pre-installed with a fresh installation of the NetLogo tool, the application also provides the ability to customize the agents' and appearance for cosmetic purposes, since as it was already mentioned, NetLogo provides visual output of its executed agent models.

There are three kinds of visual output that NetLogo provides. The first one is the movement of agents in the world of NetLogo, the second one is by utilizing a GUI element known as a "monitor" and the third one is known as a plot. Monitors function as direct output of a declared variable's value, while plots are the equivalent of a graph. Both are used for gathering statistical data .

Regarding agents, the NetLogo developers utilized their own unique way of agent implementation [21]:

- The **Observer,** a single entity with power over all the "world" of NetLogo; used for running the main parts of the program.

- The **Turtles** are the main agents used for achieving tasks and goals according to the user's needs. Different types of "turtles" can be declared by using the 'breed' primitive, which in turn can be customized to have their own states, variables, setting the level of heterogeneity or homogeneity of the model's agents[22].

- The **Patches** are the 2D squares that comprise the environment and are always in direct "contact" with the turtles. The turtles can traverse them by default without issues, unless conditions are set on either (sets of) turtles, or patches.

- The **Links** are the relationships between the turtles. They are used for agent communication; defining their level of communication (partial, complete)[22].

Additionally, NetLogo makes use of primitives to order sets of agents or a particular agent to accomplish (simple) desired tasks. Table #3 presents a few of those:

Command	**Effect**
create, crt n	Creates *n* turtles.
clear-all, ca	Clears all turtles, patches, plots and output, as well as resetting the tick counter and other global variables.
forward, fd *right, rt* *left, lt*	commands for moving the turtle. All of these commands take an integer as parameter

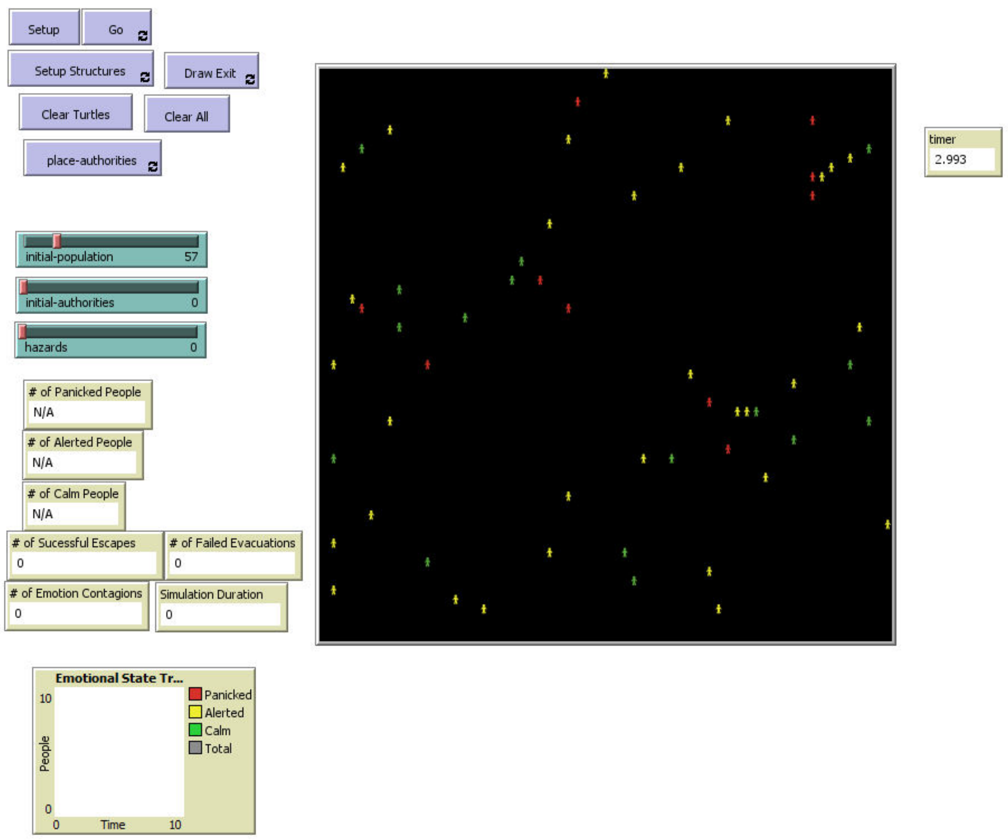
Table #3. Some basic primitives of NetLogo[21]

5.3 The Model

This subsection is the main focus of this chapter. It focuses on programming aspects of the model, namely code modules/snippets that were created based on the Analysis and Design development phases of the project. NetLogo snippets come as "procedures". Procedures are programmer-defined commands that are created by combining NetLogo primitives, a concept similar to methods in Object-Oriented Programming [27]. Any piece of code that is not used is commented out as a means to save physical memory and reduce execution time of the program. Moreover, description of self-explanatory code supported by comments in each figure is not included (mainly variable declaration). This practice is done to avoid text bloating and repetition.

The code is mainly tailored to the needs of the project, however some snippets could be utilized in other models. Further details are presented on each individual snippet's showcase.

The first thing that anyone with access to the model sees when loading the model is the UI. The project's model UI is presented in figure #13. Every single characteristic of the UI is included.



initial-population 57

initial-authorities 0

hazards 0

# of Panicked People N/A

# of Alerted People N/A

# of Calm People N/A

# of Sucessful Escapes 0

# of Failed Evacuations 0

# of Emotion Contagions 0

Simulation Duration 0

timer 2.993

Figure #15. The graphical user interface of the model

The UI is comprised of seven (7) buttons, eight (8) monitors, and one (1) plot. The environment that the agents, or "turtles" as they're called in NetLogo, are functioning in is composed of 61x61 patches, or 3721 patches in total. Every UI component is explained in the end of this subsection, along with any code tied to it.

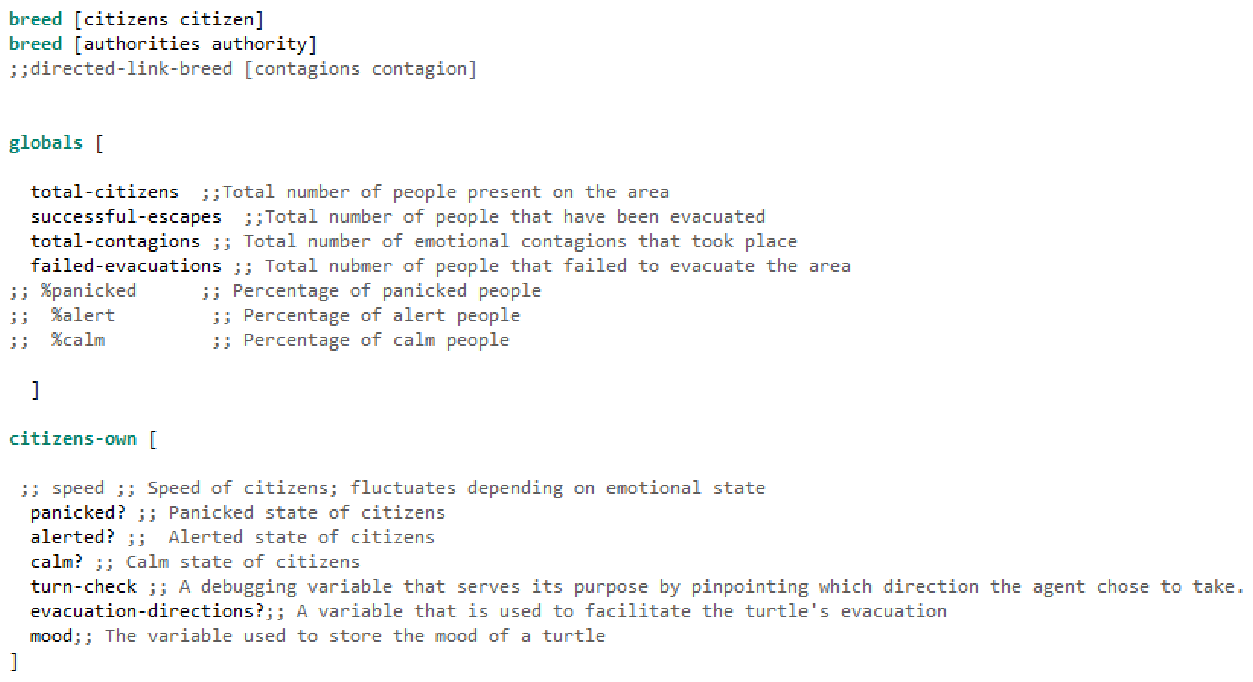
5.3.1 Modules/Snippets

Disclaimer:

Each figure showcasing code does not necessarily contain a single module/snippet. They are grouped together based on their functionality and the goal they are trying to achieve. Moreover, a fair amount of figures is accompanied by the UI elements that are directly associated with.

5.3.1.1 Breed/Variable Declaration

Code #1 depicts the variables that are used in the coding of the program. A variable is a storage location which is associated with a symbolic name (an identifier), which contains data of unknown quantity known as value [28]. Variables are used to store data invaluable for executing the logic of a program.



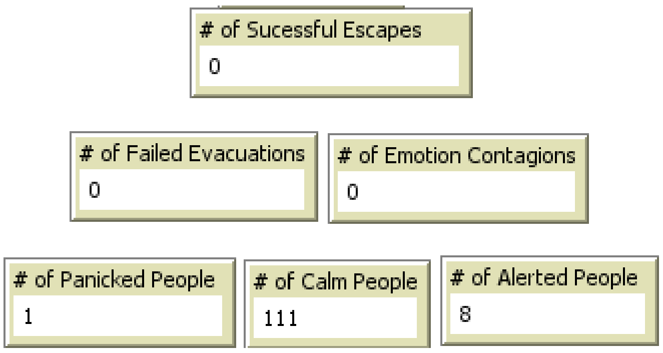
```
breed [citizens citizen]
breed [authorities authority]
;;directed-link-breed [contagions contagion]


globals [

  total-citizens  ;;Total number of people present on the area
  successful-escapes  ;;Total number of people that have been evacuated
  total-contagions ;; Total number of emotional contagions that took place
  failed-evacuations ;; Total nubmer of people that failed to evacuate the area
;; %panicked      ;; Percentage of panicked people
;;  %alert         ;; Percentage of alert people
;;  %calm          ;; Percentage of calm people

  ]

citizens-own [

 ;; speed ;; Speed of citizens; fluctuates depending on emotional state
  panicked? ;; Panicked state of citizens
  alerted? ;;  Alerted state of citizens
  calm? ;; Calm state of citizens
  turn-check ;; A debugging variable that serves its purpose by pinpointing which direction the agent chose to take.
  evacuation-directions?;; A variable that is used to facilitate the turtle's evacuation
  mood;; The variable used to store the mood of a turtle
]
```

Code #1. Variable Declaration

The model has two breeds: "Citizens" and "Authorities". Citizens are the turtles of which behavior we want to study in our model, while authorities play a supportive role to aid the turtles to evacuate, like they are supposed to in a real world scenario. The breed of links commented out was supposed to be used as a way to keep track of the times emotional contagion occurs, but was scrapped. The reason for that is that only one link can be established between two turtles, and in our case contagions between two turtles may occur more than once.

Next are the variables. NetLogo differentiates variables into global variables and variables used by turtles/patches/links. Global variables can be accessed by all agents, whereas variables declared as, for example, turtle variables, can only be accessed by turtles. It is also possible to declare variables for specific breeds of agents, as showcased in *Code #1*.

The global variables declared in the model are all used to relay information to the user of the model in real time by making use of monitors. The only exception is "total-citizens" which is used in the observer command center to count all active citizens at a given time.


# of Sucessful Escapes
0

# of Failed Evacuations
0

# of Emotion Contagions
0

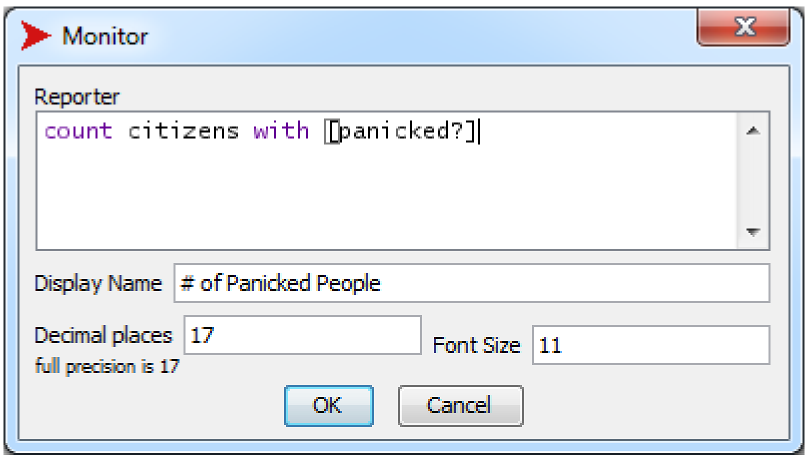
# of Panicked People
1

# of Calm People
111

# of Alerted People
8

Figure #16.Collection of monitors used to output data

The citizen variables "panicked?", "alerted?" and "calm?" are variables that hold boolean values to give the titular status to each turtle. They are also used as a means to display their numbers on monitors. An example of how this is accomplished is shown in figure #17.



Reporter
```
count citizens with [panicked?]
```
Display Name: # of Panicked People
Decimal places: 17
full precision is 17
Font Size: 11

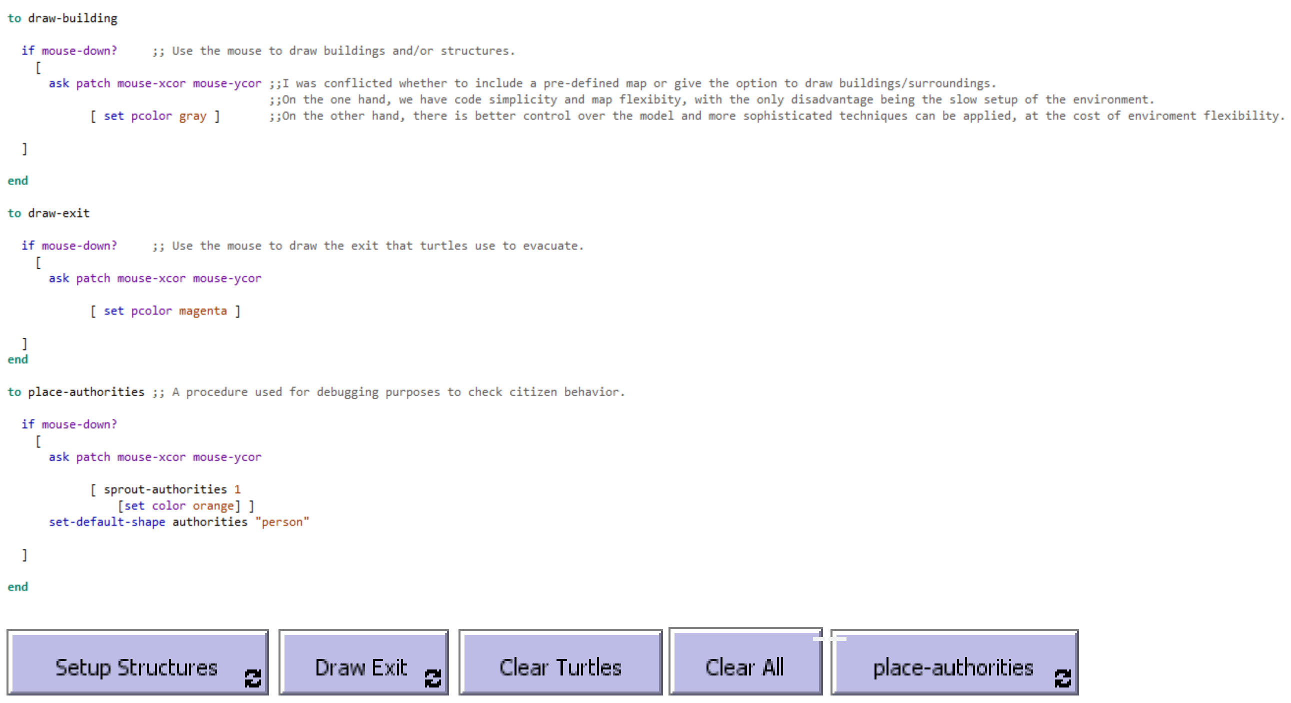
Figure #17. A monitor's code window

"Turn-check" is an integer variable used for debugging purposes and can be ignore by the user when inspecting a turtle. Its purpose is to help the developer to understand which path a turtle took when coming across an obstacle (gray patch). It is used in the "collision-check" procedure.

"Evacuation-directions?" is a boolean variable used to facilitate the evacuation of turtles. If it's true, they act slightly different to find the exit. It is used in the "get-directions" procedure.

"Mood" is an integer variable that is used in both initiating a turtle's mood and keeping track of a turtle's mood during runtime.

5.3.1.2 World Building



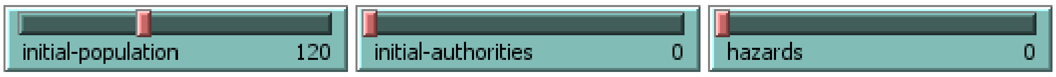
```
to draw-building

  if mouse-down?     ;; Use the mouse to draw buildings and/or structures.
    [
      ask patch mouse-xcor mouse-ycor ;;I was conflicted whether to include a pre-defined map or give the option to draw buildings/surroundings.
                                      ;;On the one hand, we have code simplicity and map flexibity, with the only disadvantage being the slow setup of the environment.
            [ set pcolor gray ]       ;;On the other hand, there is better control over the model and more sophisticated techniques can be applied, at the cost of enviroment flexibility.

  ]

end

to draw-exit

  if mouse-down?     ;; Use the mouse to draw the exit that turtles use to evacuate.
    [
      ask patch mouse-xcor mouse-ycor

            [ set pcolor magenta ]

  ]
end

to place-authorities ;; A procedure used for debugging purposes to check citizen behavior.

  if mouse-down?
    [
      ask patch mouse-xcor mouse-ycor

            [ sprout-authorities 1
                [set color orange] ]
      set-default-shape authorities "person"

  ]

end
```

Code #2. World-building feature accompanied by its UI elements

These are the deployed code snippets that assist the model's user in creating their own evacuation scenario specification. The "draw-building" procedure paints patches gray, which the turtles recognize as buildings/debris. From the user's perspective, this is done by pressing the "Setup Structures" button and then keeping the left mouse click pressed to draw. Once the building placement is done, the user has to press the button again to cancel the option. Drawing an exit that the turtles can evacuate from is done exactly the same way, with the sole difference being the color (magenta). Take note that a simulation attempt cannot be initiated if there does not exist at some form of an exit (at least one magenta patch). If the user attempts to do that a runtime error occurs. Also, the option to pause the model and draw new buildings/more exits exists. In case the user wishes to repeat a simulation with the same world structure, they merely have to press the "Clear Turtles" button. Moreover, in case they wish to completely reset the world, they can click the "Clear All" button. The "place-authorities" procedure (and titular button) has the same workings, although the this feature is only meant for debugging purposes and not for actual simulations of evacuation.

5.3.1.3 Turtle Set Up/Initialization

```
to setup-figures ;; Procedure that initializes turtle numbers, locations and variables
                 ;; Has potential for future improvements and/or additions

  ;; ifelse chance >= 50
  ;; [ask n-of 1 patches with [pcolor = black] [
  ;;   [set pcolor = red ] ] ] ;;for fire
  ;;[ask n-of 1 patches with [pcolor = black][
  ;; [set pcolor = blue ] ] ] ;;for flood

  ask n-of initial-population patches with [pcolor = black]
    [ sprout-citizens 1
      [ set color blue ] ]

  ask n-of initial-authorities patches with [pcolor = black]
    [ sprout-authorities 1
      [ set color orange ] ]

  ask n-of 1 patches with [pcolor = magenta]
   [ sprout-authorities 1
    [ set color orange ] ]



  ask n-of hazards patches with [pcolor = black]
      [ set pcolor red]

  set-default-shape citizens "person"
  set-default-shape authorities "person"
  set successful-escapes 0
  set failed-evacuations 0

  ask citizens [
    set total-contagions 0
    set panicked? false
    set alerted? false
    set calm? false
    set evacuation-directions? false
    catastrophe-occurs
  ]

end
```



initial-population 120

initial-authorities 0

hazards 0

Code #3. Turtle set up snippet accompanied by its UI elements

The first figure presented that contains a single snippet. The "setup-figures" procedure is responsible for initializing the turtles' variables' values, as well as the turtles' and authorities' agent numbers on the world. It is run only once every scenario execution. The desired numbers can be adjusted by making use of the sliders provided in the UI of the model. The reasoning behind creating a separate procedure instead of just dumping all the code in the "Setup" basic procedure is code clarity and bug avoidance, as well as better control over initialized elements. In the end of the snippet the code leads up to the "catastrophe-occurs" procedure, which is executed in this block. The "hazards" slider should not be used, as it exists for debugging purposes.

5.3.1.4 Catastrophe Occurs

```
to catastrophe-occurs;; The "event" that initiates the whole process of evacuation
                     ;; It assigns statuses to all existing citizens and starts the evacuation.

  set mood random 100

  ifelse mood >= 69
    [ calm-down ]
    [ if mood <= 68
      [ get-alerted ]
    ]

  if mood <= 15
    [get-panicked]

end
```

Code #4. Mood initiation

The "catastrophe-occurs" procedure is used to initialize the mood of turtles generated by the "setup-figures" procedure. It is run only once every scenario execution. Although the name implies that this procedure is used for other means than emotional initialization, that is not the case. Take notice that, although authorities are also human, and thus possess emotions, it is not within the scope of the project to keep track of them. By glancing at *Code #4*, one can deduct that the chance for a turtle to be, for example, panicked during the event initialization is 15%.

5.3.1.5 Agent Interaction

```
to interact;; This procedure mimics emotional contagion

  ask citizens with [alerted?] [
    if any? citizens in-radius 2 with [panicked?]
        [ set mood mood - 1
          set total-contagions total-contagions + 1]

    if any? citizens in-radius 2 with [calm?]
        [ set mood mood + 1
          set total-contagions total-contagions + 1]

    if any? other citizens in-radius 2 with [alerted?]
        [ set mood mood + 0
          set total-contagions total-contagions + 1]

  ]
  ask citizens with [calm?] [
    if any? citizens in-radius 2 with [alerted?]
      [set mood mood - 1.5
        set total-contagions total-contagions + 1 ]

    if any? citizens in-radius 2 with [panicked?]
      [set mood mood - 4
        set total-contagions total-contagions + 1 ]

    if any? other citizens in-radius 2 with [calm?]
      [set mood mood + 0.5
        set total-contagions total-contagions + 1]
  ]

  ask citizens with [panicked?] [
    if any? other citizens in-radius 2 with [panicked?]
      [ set mood mood - 2
        set total-contagions total-contagions + 1 ]

    if any? citizens in-radius 2 with [alerted?]
      [ set total-contagions total-contagions + 1 ]

    if any? citizens in-radius 2 with [calm?]
      [ set total-contagions total-contagions + 1 ]

  ]

end
```

Code #5. Agent interaction

The "interact" procedure is the main procedure that handles the mimicry of emotional contagion between turtles during their interaction. To elaborate, turtles with aggravated mood values affect those with high mood values more, just as it is showcased on *Code #5.*Each turtles has behaves differently according to both their emotional status, as well as the emotional status of the turtle they come across. To provide an example, let us assume that there exist three turtles, one alerted, one panicked and one calm and they come in contact with each other. In the "alerted" turtle's case, if all meet at the same time, its mood will stay the same (-1+1+0). In the "panicked" turtle's case, its mood won't change, since the other two turtles (calm and alerted) won't affect its mood.

6. Simulation & Results

It is of paramount importance to assess the capabilities of a model with possible applications in systems that handle emergency situations. In order to achieve the goals set in the beginning of the project, a specific number of scenarios were created based on the model's capabilities. Thus, this section focuses entirely on simulating the aforementioned scenarios based on real world representations, and therefore is the ultimate purpose of this project. The performance of the model has been measured on runtime. The expected results are listed as tables under the description of every scenario included below.

The world that every scenario takes place in is 61x61 patches wide (3721 patches). The exit(s) are 10x5 patches wide (50 patches) and are always placed at the sides of the world. Each scenario presentation is accompanied by one supplementary screenshot showcasing the object distribution(buildings, exit(s),etc.) in the world of the model, adhering to each scenario's specifications.

Furthermore, every scenario is executed with varying parameters that reflect different situations under which the evacuation has to take place. Their purpose is to offer more detailed information by yielding varying results based on the input. The mentioned parameters are listed below:

- Population (number of "citizen" turtles)
- Presence of Authoritative Figures (number of "authority" turtles)

To simplify matters, each scenario comes with four versions, each version with different values for the parameters listed above. "Low population" equals to fifteen (15) turtles, "medium population" stands for seventy-five (75) turtles, while "high population" stands for one-hundred-fifty turtles (150).

There existed plans to include a third parameter regarding hazards such as fires/floods, however that parameter was not included. Further details about this exclusion are presented in Chapter 7, "Future Work" subsection.

6.1 Scenario 1

6.1.1 Open Field

The first scenario pertains to scarcely-populated areas, fields, and areas in which buildings do not exist for various reasons. It is also effective as a simulation of enclosed spaces, such as clubs, cinemas, theatres, etc. In this case scenario, the simulation does not employ any buildings whatsoever in order to measure the agent's performance under these circumstances. It is showcased in figure #16.

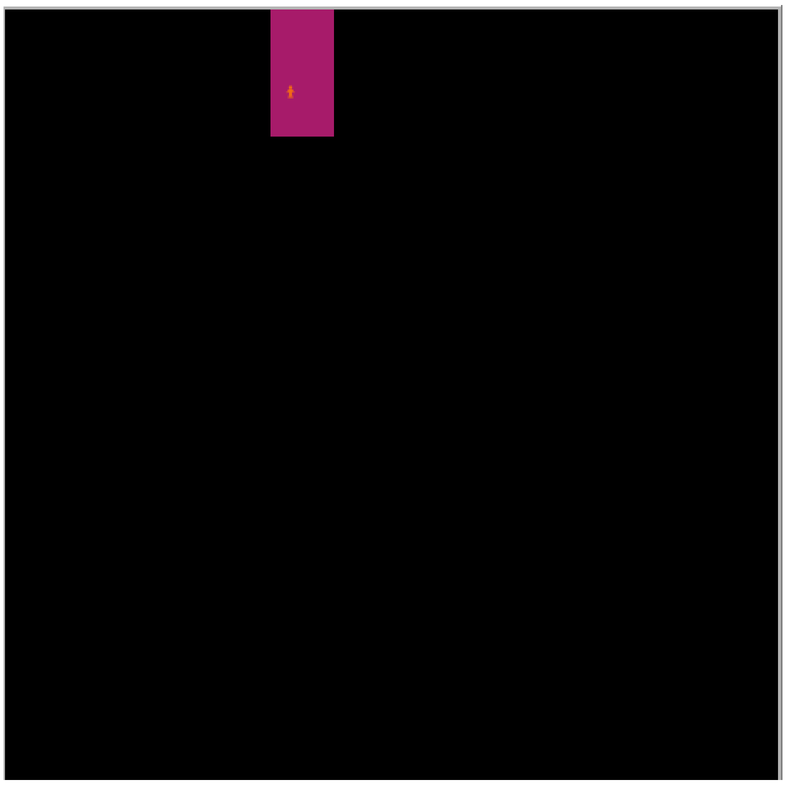


Figure #18. Open field scenario

Low Population (15)

No authorities (0)

Simulation Attempts	Successful Evacuations	Failed Evacuations	Emotional Contagions	Evacuation Duration (in ticks)
#1	14	1	80	190
#2	12	3	74	65
#3	15	0	30	58
#4	15	0	90	195

#5	15	0	76	183
#6	15	0	97	74
#7	14	1	80	140
#8	15	0	59	49
#9	15	0	41	98
#10	15	0	59	156
Average(mean)	14.7	0.3	68.9	120.8
	96.55% of Citizens			

Table #4.

Medium Population(75)
Four authorities (4)

Simulation Attempts	Successful Evacuations	Failed Evacuations	Emotional Contagions	Evacuation Duration (in ticks)
#1	70	5	4402	205
#2	66	9	3709	346
#3	73	2	3453	74
#4	69	6	3430	92
#5	73	2	4050	195
#6	67	8	3146	45
#7	69	6	3168	121
#8	69	6	3918	224
#9	69	6	3121	168
#10	69	6	3857	45
Average(mean)	69.4	5.6	3625.4	151.5
	91.93%			

Table #5.

Medium Population (75)
No Authorities (0)

Simulation Attempts	Successful Evacuations	Failed Evacuations	Emotional Contagions	Evacuation Duration (in ticks)
#1	62	13	2833	98
#2	68	7	4665	82
#3	68	7	4839	35
#4	68	7	3496	478

#5	70	5	4815	183
#6	61	14	4052	346
#7	63	12	3646	134
#8	69	6	3892	163
#9	67	8	4120	150
#10	58	17	3767	178
Average(mean)	65.4	9.6	4012.5	184.7
	83.32%			

Table #6.

High Population (150)
Four authorities (4)

Simulation Attempts	Successful Evacuations	Failed Evacuations	Emotional Contagions	Evacuation Duration (in ticks)
#1	111	39	13208	98
#2	104	46	15920	121
#3	102	48	12169	88
#4	122	28	15679	103
#5	124	26	15069	84
#6	128	22	14998	145
#7	114	36	14484	63
#8	115	35	14173	85
#9	109	41	14022	58
#10	114	36	13767	102
Average(mean)	114.3	35.7	14348.9	94.7
	68.77%			

Table #7.

High Population (150)
No authorities (0)

Simulation Attempts	Successful Evacuations	Failed Evacuations	Emotional Contagions	Evacuation Duration (in ticks)
#1	107	43	17487	174
#2	114	36	14780	117
#3	103	47	13626	241
#4	113	37	14657	205
#5	118	32	16247	106

#6	113	37	20478	217
#7	111	39	15838	146
#8	104	46	14766	76
#9	107	43	15255	343
#10	101	49	15240	368
Average(mean)	109.1	40.9	15837.4	199.3
	62.51%			

Table #8.

6.2 Scenario 2

6.2.1 Village

The second scenario involves the evacuation of underpopulated areas, villages, and other rural settlements. It consists of three (3) small buildings and the usual 10x5 exit as shown on figure #17.

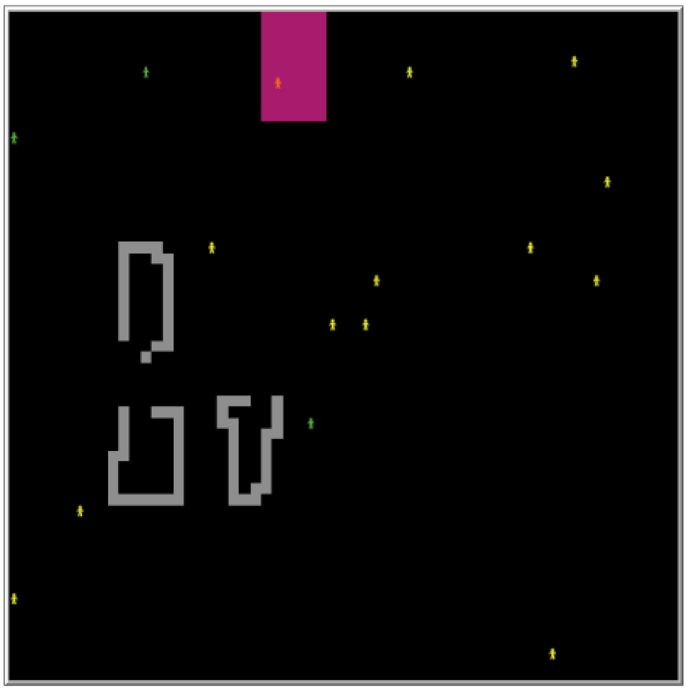


Figure #19. Village Scenario

Low population (15)
No authorities (0)

Simulation Attempts	Successful Evacuations	Failed Evacuations	Emotional Contagions	Evacuation Duration (in ticks)

#1	15	0	265	81
#2	15	0	153	614
#3	14	1	182	112
#4	15	0	21	104
#5	15	0	37	207
#6	14	1	50	164
#7	14	1	139	35
#8	14	1	40	150
#9	14	1	22	188
#10	15	0	260	311
Average(mean)	14.5	0.5	116.9	196.6
	96.55%			

Table #9.

Medium population (75)
Four authorities (4)

Simulation Attempts	Successful Evacuations	Failed Evacuations	Emotional Contagions	Evacuation Duration (in ticks)
#1	64	11	3424	79
#2	67	8	3043	220
#3	64	11	3093	208
#4	72	3	3964	126
#5	65	10	3271	81
#6	72	3	3547	134
#7	73	2	3303	151
#8	67	8	4140	215
#9	64	11	3411	170
#10	68	7	3522	98
Average(mean)	676	7.4	3471.8	148.2
	88.05%			

Table #10.

Medium Population (75)
No authorities (0)

Simulation Attempts	Successful Evacuations	Failed Evacuations	Emotional Contagions	Evacuation Duration (in ticks)

#1	66	9	4636	220
#2	63	12	3890	112
#3	67	8	7772	230
#4	72	3	5392	250
#5	66	9	4404	136
#6	67	8	4637	95
#7	60	15	3438	85
#8	68	7	6301	154
#9	70	5	3606	114
#10	70	5	4254	404
Average(mean)	66.9	8.1	4833	180
	87.89%			

Table #11.

High Population (150)
Four authorities (4)

Simulation Attempts	Successful Evacuations	Failed Evacuations	Emotional Contagions	Evacuation Duration (in ticks)
#1	130	20	18117	172
#2	114	36	12931	28
#3	106	44	18047	257
#4	109	41	17156	75
#5	125	25	15679	23
#6	114	36	13421	70
#7	125	25	17096	124
#8	103	47	14168	133
#9	116	34	14522	131
#10	124	26	15020	163
Average(mean)	116.6	33.4	15615.7	117.6
	71.46%			

Table #12.

High Population (150)
No authorities (0)

Simulation Attempts	Successful Evacuations	Failed Evacuations	Emotional Contagions	Evacuation Duration (in ticks)
#1	111	39	16378	278
#2	124	26	17965	97
#3	95	55	14629	96
#4	114	36	19691	118
#5	112	38	16453	142
#6	112	38	16158	84
#7	115	35	17103	122
#8	109	41	14304	145
#9	113	37	17021	128
#10	111	39	16792	131
Average(mean)	111.6	38.4	16649.4	134.1

Table #13.

6.3 Scenario 3

6.3.1 Town

The penultimate scenario that assists in yielding results from medium-sized towns. It consists of six (6) buildings and the usual 10x5 exit, as shown on figure #18.

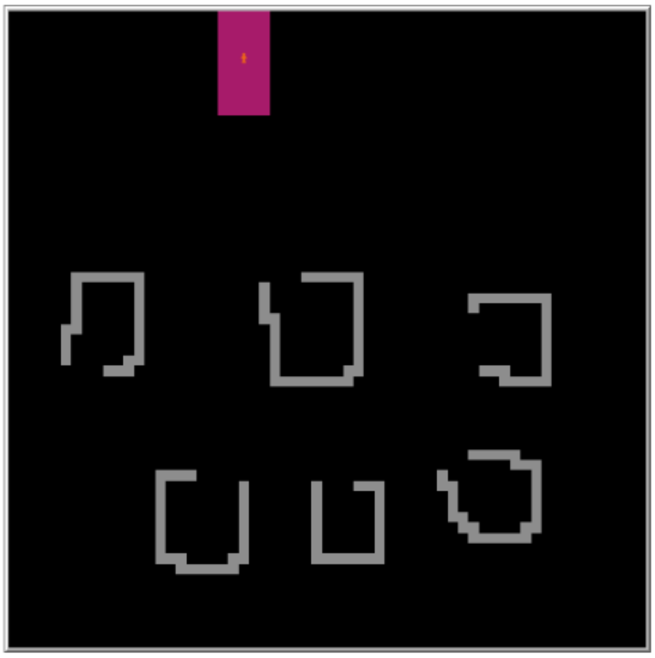


Figure #18. Town Scenario

Low population (15)
No authorities (0)

Simulation Attempts	Successful Evacuations	Failed Evacuations	Emotional Contagions	Evacuation Duration (in ticks)
#1	15	0	35	141
#2	14	1	2	152
#3	15	0	77	439
#4	15	0	573	1315
#5	15	0	457	87
#6	14	1	103	226
#7	15	0	40	96
#8	14	1	484	147
#9	15	0	118	94

#10	15	0	187	77
Average(mean)	14.7	0.3	207.6	269.7
	97.96%			

Table #14.

Medium population (75)
Four authorities (4)

Simulation Attempts	Successful Evacuations	Failed Evacuations	Emotional Contagions	Evacuation Duration (in ticks)
#1	70	5	4203	170
#2	65	10	2943	151
#3	68	7	5624	212
#4	71	4	4246	77
#5	70	5	5389	82
#6	65	10	6324	79
#7	69	6	4451	80
#8	70	5	5046	120
#9	61	14	3205	92
#10	65	10	2337	217
Average(mean)	67.4	7.6	4376.8	128
	88.72%			

Table #15.

Medium Population (75)
No authorities (0)

Simulation Attempts	Successful Evacuations	Failed Evacuations	Emotional Contagions	Evacuation Duration (in ticks)
#1	67	8	5725	108
#2	67	8	6007	434
#3	72	3	4572	165
#4	73	2	4550	105
#5	66	9	6935	235
#6	67	8	4762	258
#7	65	10	3830	276
#8	67	8	4460	80
#9	72	3	5892	93
#10	71	4	5721	152

Average(mean)	68.7	6.3	5245.4	190.6
	92.87%			

Table #16.

High Population (150)
Four authorities (4)

Simulation Attempts	Successful Evacuations	Failed Evacuations	Emotional Contagions	Evacuation Duration (in ticks)
#1	110	40	14322	92
#2	115	35	14566	148
#3	118	32	15142	95
#4	112	38	19476	38
#5	113	37	15625	222
#6	110	40	16854	69
#7	119	31	15900	920
#8	117	33	17917	171
#9	116	34	15254	98
#10	122	28	19141	352
Average(mean)	115.2	34.8	16419.7	220.5
	69.79%			

Table #17.

High population (150)
No authorities (0)

Simulation Attempts	Successful Evacuations	Failed Evacuations	Emotional Contagions	Evacuation Duration (in ticks)
#1	104	46	26380	303
#2	112	38	20991	210
#3	103	47	15670	90
#4	100	50	15479	190
#5	109	41	18699	134
#6	102	48	15987	81
#7	116	34	26642	390
#8	112	38	19192	98
#9	103	47	13500	80
#10	103	47	16504	70

Average(mean)	106.4	43.6	18904.4	164.6
	59.02%			

Table #18.

6.4 Scenario 4

6.4.1 City

Last, but not least, we have the "city" scenario which represents big cities, capitals, etc. It consists of nine (9) buildings and the usual 10x5 patches exit, as showcased in figure #19.

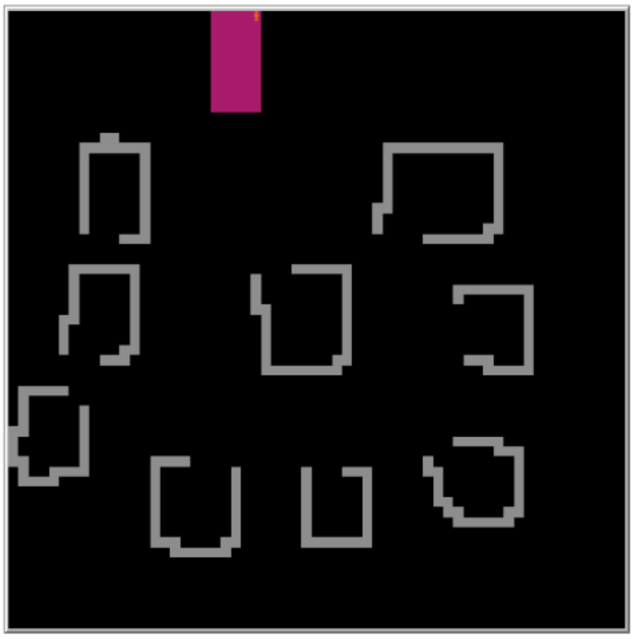


Figure #21. City Scenario

Low population (15)
No authorities (0)

Simulation Attempts	Successful Evacuations	Failed Evacuations	Emotional Contagions	Evacuation Duration (in ticks)
#1	15	0	560	748
#2	15	0	408	200
#3	14	1	483	72
#4	15	0	61	132
#5	15	0	412	349
#6	15	0	348	142
#7	13	2	410	98

#8	15	0	109	282
#9	14	1	143	132
#10	14	1	70	470
Average(mean)	14.5	0.5	300.4	262.5
	96.55%			

Table #19.

Medium Population (75)
Four authorities (4)

Simulation Attempts	Successful Evacuations	Failed Evacuations	Emotional Contagions	Evacuation Duration (in ticks)
#1	65	10	6876	138
#2	63	12	5918	112
#3	66	9	3469	201
#4	68	7	5529	385
#5	69	6	4148	231
#6	67	8	6077	310
#7	63	12	3246	126
#8	68	7	5975	154
#9	68	7	4938	168
#10	63	12	4668	171
Average(mean)	66	9	5084.4	199.6
	86.36%			

Table #20.

[]

Medium population (75)
No authorities (0)

Simulation Attempts	Successful Evacuations	Failed Evacuations	Emotional Contagions	Evacuation Duration (in ticks)
#1	70	5	5604	181
#2	66	9	6067	175
#3	66	9	4947	214
#4	62	13	12367	293
#5	67	8	9715	275
#6	59	16	8796	192
#7	64	11	6168	158
#8	69	6	7268	483
#9	66	9	5802	84
#10	63	12	3751	122
Average(mean)	65.2	9.8	7048.5	217.7
	84.97%			

Table #21.

High Population (150)
Four Authorities (4)

Simulation Attempts	Successful Evacuations	Failed Evacuations	Emotional Contagions	Evacuation Duration (in ticks)
#1	100	50	14071	195
#2	110	40	17295	124
#3	108	42	18369	78
#4	109	41	17707	149
#5	116	34	16158	609
#6	102	48	17839	198
#7	110	40	17481	140
#8	112	38	16625	282
#9	116	34	17188	315
#10	102	48	19109	229
Average(mean)	108.5	41.5	17184.2	231.9
	61.75%			

Table #22.

High Population (150)
No Authorities (0)

Simulation Attempts	Successful Evacuations	Failed Evacuations	Emotional Contagions	Evacuation Duration (in ticks)
#1	100	50	25143	158
#2	112	38	20005	203
#3	108	42	25806	184
#4	114	36	20683	312
#5	106	44	18806	122
#6	109	41	18091	140
#7	110	40	19127	106
#8	114	36	22926	225
#9	96	54	21983	157
#10	100	50	20393	391
Average(mean)	106.9	43.1	21296.3	199.8
	59.68%			

Table #23.

With the conclusion of the final simulation experiment of the final scenario, four final aggregation tables are listed below to summarize the findings. These four tables offer assistance at assessing the overall capabilities of the model by providing easier access to the findings.

Scenario #1. "Open Field"	Low population	Medium Population/ Four authorities	Medium Population/ No authorities	High Population/ Four authorities	High population/ No authorities
Successful Evacuations	14.7	69.4	65.5	114.3	109.1
Failed Evacuations	0.3	5.6	9.6	35.7	40.9
Emotional Contagions	68.9	3625.4	4012.5	14348.9	15837.4

Evacuation Duration (in ticks)	120.8	151.5	184.7	94.7	199.3

Table #24.

Scenario #2. "City"	Low population	Medium Population/ Four authorities	Medium Population/ No authorities	High Population/ Four authorities	High population/ No authorities
Successful Evacuations	14.5	67.6	66.9	116.6	111.6
Failed Evacuations	0.5	7.4	8.1	33.4	38.4
Emotional Contagions	116.9	3471.8	4483	15615.7	16649.4
Evacuation Duration (in ticks)	196.6	148.2	180	117.6	134.1

Table #25.

Scenario #3. "Town"	Low population	Medium Population/ Four authorities	Medium Population/ No authorities	High Population/ Four authorities	High population/ No authorities
Successful Evacuations	14.7	68.7	67.4	115.2	106.4
Failed Evacuations	0.3	7.6	6.3	34.8	43.6
Emotional Contagions	207.6	4376.8	5245.4	16419.7	18904.4
Evacuation Duration (in ticks)	269.7	128	190.6	220.5	164.6

Table #26.

Scenario #4. "City"	Low populati on	Medium Population/ Four authorities	Medium Population/ No authorities	High Population/ Four authorities	High population/ No authorities
Successful Evacuations	14.5	66	65.2	108.5	106.9
Failed Evacuations	0.5	9	9.8	41.5	43.1
Emotional Contagions	300.4	5084.4	7048.5	17184.2	21296.3
Evacuation Duration (in ticks)	262.5	199.6	217.7	231.9	199.8

Table #27.

The conclusion that can be drawn from these statistical results is that authorities **do** increase the chances of citizens reaching the exit, albeit not by a great margin. In addition to that, the greater the population in each of the four version of each scenario, the more emotional contagions occur. Moreover, it is apparent that the larger the population count, the higher are the casualties, which is a system flaw attributed to the (mostly)randomized way the turtles move while searching for the exit. More accurate results could be drawn with a larger sample, however this suffices for the scope of this project.

7. Testing and Evaluation

7.1 Testing

There exist many testing techniques that are utilized in a software development cycle. These techniques are classified under two distinct categories, named White and Black Box testing[26].

White box testing is when the software logic is transparent and available for the purpose of testing and it is conducted by developers to test their own modules.

Black box is the opposite of white box testing: It is achieved by not tampering whatsoever with the software logic and inner workings of a software and focuses on input and output. It is often conducted along with the customer who is almost always not acquainted with software development [26].

During the simulation of scenarios used for creating simulations and cataloguing their results in Chapter 6, the testing techniques that were conducted are as follows:

Unit Testing:

Unit testing is a technique that is used for testing small pieces of code. For this project's needs, unit testing was conducted on each NetLogo procedure separately. This technique is classified as a white box technique.

Integration Testing:

Integration testing is a technique that is used for testing how two or more classes function(in our case, procedures) when pieced together.

Functional Testing:

Functional testing is a technique that is used on testing complete software products. This is extremely similar with the simulation scenario presentation that is showcased in Chapter 6. This technique is classified as a black box technique.

7.2 Bugs

There is no software product in existence that is error/bug-free, and this is also the case with the model of this project. Bugs are faulty, unintentional software behaviors that may or may not affect the functionality of a software system. The bugs found so far are listed below as bullet points with ascending level of importance:

1. When the model is loaded for the first time and set up, the setup button has to be pressed twice so that the turtles can appear as "person" shape.

2. Sometimes when one turtle is left wandering on its own, it changes its state from Alerted to Panicked to Calm repeatedly.

3. Sometimes emotional contagion count is far greater than expected. A possible reason for that may be the count handling when more than two turtles come in contact with each other.

4. Often turtles may ignore gray patches and go through them to reach the exit.

Although there exist ways to fix #2 and #3 bugs, the time constraints are not allowing such a venture at this point. Bug #4 could sometimes be misinterpreted as a feature. For example, by drawing a diagonal gray line that is supposed to represent debris, one could say that the turtles eventually manage to overcome the debris by climbing it. This is fine, however it is not the intended behavior.

7.3 Self-Evaluation

The following subsection is a short summary of self-reflection regarding the planning of the project, the actions that were performed to see to its conclusion, as well as the outcome. Furthermore, any acquisition of skills and experiences are also included.

There existed great amounts of enthusiasm when this project begun, which also clouded the judgment of the author about what could be accomplished and what not. However the most grave problems were not about the author's capabilities, but about their personal issues. Also, the project was certainly underestimated to some degree, partly due to the fact that NetLogo seemed to be an easy to learn, but it certainly proved to be a hard language to master.

7.4 Project Evaluation

This subsection presents a short summary of the project's accomplishments, as well as its shortcomings and future possibilities that it may offer. In addition to the above, a table about the objectives completion rate is provided. To elaborate, every objective listed on Chapter 1.2 is rated from 1 to 5 (Poor to Great). The table's projection of the project's evaluation is formed subjectively, thus the values may be higher or lower.

Objective Completion Evaluation

Objectives	Poor	Average	Good	Very Good	Excellent
#1			X		
#2			X		
#3		X			
#4				X	
#5			X		

Table #28.

7.5 Personal Gain

The following subsection is a short summary of self-reflection regarding the planning of the project, the actions that were performed to see to its conclusion, as well as the outcome. Furthermore, any acquisition of skills and experiences are also included.

Upon the completion of this project, the author of this project contemplated the outcome, along with all shortcomings and accomplishments tied to it. The emerging results were four acquired, or at least improved, skills that shall prove invaluable assets to not only future projects, but endeavors in general.

1. Project Planning
2. Improvisation
3. Use of questions-and-answer websites (Stackoverflow)
4. Individual Snippet/Module Testing

Project planning is a skill that is often overlooked by fledgling computer scientists and computer scientist students. It is a very important skill to own in order to establish sound foundations for achieving various goals. In this case, the project planning was not effective from the get-go and it led to many mishaps over the course of the project. It is regrettable to admit that most of the planning that is discussed in Chapter 3 was not followed. This leads us to the first skill:

1. Improvisation

Improvising when something does not go according to plan is invaluable. Across the duration of the project, and especially during the software implementation and testing phases, many things went awry. Faulty code, abnormal output (more contagions than intended, etc.) are just a few example of mishaps. Working around them by improvising solutions was a stimulating experience, both exciting and stressful at the same time.

2. Use of questions-and-answer websites (Stackoverflow)

Due to the fact that NetLogo is an unpopular programming language, the sources online for studying and learning the language are very limited. Necessity gave rise to the thought of using alternative sources otherwise avoided in the past (for reasons irrelevant at this point), and so the first step towards using Stackoverflow was made. Overall, two (2) questions were asked to the Stackoverflow community to assist the cause of this project.

3. Individual Snippet/Module Testing

One of the major issues that had to be tackled was trouble created by lack of module testing. During implementation, each code snippet/module were not tested individually. Although the RAD software development method (Chapter 3) was used ,this problem was still quite the hassle. The lack of unit testing, along with the fact that the experience with NetLogo was inefficient, led to a great progress slowdown. This project helped the author realize how important this skill is for a computer scientist.

8. Conclusion and Future Work

This subsection concludes this report, and focuses on future work that can be conducted on top of the existing model to improve various aspects of it.

Although this project underwent great hurdles and exertions, it was deemed successful. Moreover, although the project succeeded, the model is by no means applicable in real-life situations, due to reasons listed in Chapter 7. The main objectives set on Chapter 1 and evaluated on Chapter 7 were all completed to varying degrees, which in the end matters little as long the end results are satisfactory, which they are.

However, there exists always room for improvement for every piece of software and this project is no different.

The model current features can be expanded further. As an example, more monitors could be added to capture the states of evacuated turtles (alerted, etc.) for statistical purposes. The existing procedures could greatly facilitate such an attempt.

In addition to that, environmental hazards that were discussed on the initial system description could be added as a new feature. They could act as a deterrent for turtles in their vicinity and affect their emotional state, as well as pose a hazard for their well-being during the evacuation.

References

[1]"Emergency Preparedness and Evacuation Plan." California, pp. 1,3,4 [Online]. Available: http://www.newportdunes.com/wp-content/uploads/Evacuation-Plan.pdf. [Accessed: 13- Jan- 2018]

[2]"Guidance on Enclosed Space Entry and Rescue." (n.d.). [online] pp.2-3. Available at: https://tinyurl.com/ycywdcp8 [Accessed 14 Jan. 2018].

[3]Nabil Mohammed Ali Munassar, A. Govardhan, "A Comparison Between Five Models of Software Engineering", *IJCSI International Journal of Computer Science Issues,* Vo.7, Issue 5, pp. 95-97, September 2010

[4]B. Zeigler, A. Muzy and L. Yilmaz, "Artificial Intelligence in Modeling and Simulation", pp. 3,4,9,10

[5] P. Leitao, "Multi-agent Systems in Industry: Current Trends & Future Challenges", pp. 1-3

[6]S. Tisue and U. Wilensky, "NetLogo: Design and implementation of a multi-agent modeling environment", p. 2-14, 2004.

[7]K. Tate, "Dyson Spheres: How Advanced Alien Civilizations Would Conquer the Galaxy (Infographic)", *LiveScience*, January 14 2014. [Online]. Available: https://www.livescience.com/42554-dyson-spheres-how-advanced-alien-civilizations-would-conquer-the-galaxy-infographic.html. [Accessed: 19- Jan- 2018].

[8]L. Geggel, "Rare, Mohawk-Wearing Fish Discovered 'Walking' on Seafloor", *LiveScience*, 2018. [Online]. Available: https://www.livescience.com/61534-rare-red-handfish-discovered.html. [Accessed: 25- Jan- 2018]

[9]S. Barsade, "The Ripple Effect: Emotional Contagion and Its Influence on Group Behavior", *Administrative Science Quarterly*, vol. 47, no. 4, pp. 646-656, December 2002.

[10]"SELECTING A DEVELOPMENT APPROACH", *Office of Information Services*, pp. 1-2, 5-6, February 2005, Revalidated: March 2008

[11]"What is Agile model – advantages, disadvantages and when to use it?", *Istqbexamcertification.com*, 2018. [Online]. Available: http://istqbexamcertification.com/what-is-agile-model-advantages-disadvantages-and-when-to-use-it/. [Accessed: 25- Jan- 2018]

[12]]F. Mills and R. Stufflebeam, "Introduction to Intelligent Agents", *http://www.mind.ilstu.edu/*, 2005. [Online]. Available: http://www.mind.ilstu.edu/curriculum/ants_nasa/intelligent_agents.php?modGUI=237&compGUI=1751&itemGUI=3028. [Accessed: 19- Jan- 2018]

[13]Ilias Sakellariou, Petros Kefalas, Suzie Savvidou, Ioanna Stamatopoulou, Marina Ntika, "The role of emotions, mood, personality and contagion in Multi-agent system decision making", pp. 1-4, 5-6

[14]C. Lisetti, "Personality, Affect and Emotion Taxonomy for Socially Intelligent Agents", *American Association for Artificial Intelligence*, pp. 398-400, 2002.

[15]P. Kefalas, I. Sakellariou, S. Savvidou, I. Stamatopoulou and M. Ntika, "The role of mood on emotional agents behavior", pp. 1-3, 6-8.

[16]M. Ntika, P. Kefalas, I. Stamatopoulou and M. Gheorghe, "The Role of Contagion in Emotional Multi-Agent Systems", pp. 2-5, 8-10.

[17]B. López, *Artificial intelligence research and development*. Amsterdam: IOS Press, 2006, pp. 91,255.

[18]S. Robbins and T. Judge, *Organizational behavior*. Pearson Education, Inc., 2010, pp. chapter 8, pp. 260-267.

[19] D. Watson, Mood and Temperament, New York, Guilford Publications, 2000.

[20]S. Russell and P. Norvig, *Artificial intelligence: A Modern Approach*, 3rd ed. Upper Saddle River, New Jersey: Pearson Education, Inc., 2010, pp. pp. 34-46.

[21]M. Berryman and S. Angus, "Tutorials on Agent-based modeling with NetLogo and Network Analysis with Pajek", *World Scientific Review*, pp. 2-11, 2009.

[22]P. Stone and M. Veloso, "Multiagent Systems: A Survey from a Machine Learning Perspective", *Autonomous Robotics*, vol. 8, no. 3, 2000.

[23]Costa, P.T, J., McCrae, R.: Revised NEO Personality Inventory (Neo-PI-R) and NEO Five-Factor Inventory (Neo-FFI) manual (1992)

[24]B. Khawla and B. Molnár, "An FSM Approach for Hypergraph Extraction Based on Business Process Modeling", The 3rd conference on Computing Systems and Applications, CSA'2018EMP, Algiers., pp. 3-5, 24th-25th of April, 2018.

[25]A. Dennis, B. Wixom, D. Tegarden and E. Seeman, *System Analysis And Design*, 5th ed. John Wiley & Sons, Inc., pp. 104,114 260-262,314, 321-322, 366-369.

[26]S. Nidhra, "Black Box and White Box Testing Techniques - A Literature Review", *International Journal of Embedded Systems and Applications*, vol. 2, no. 2, pp. 29-50, 2012.

[27] "NetLogo FAQ (Frequently Asked Questions)", http://ccl.northwestern.edu/netlogo/faq.html , December 14, 2015.

[28]A. Aho, M. Lam, R. Sethi and J. Ullman, *Compilers*, 2nd ed. Pearson Education, Inc., 2007, pp. 26-28.

[29]I. Sakellariou, Evacuation_SXM_emotions, Netlogo model, 2014

APPENDIX

This section includes content that was omitted from the final report due to not meeting expectations set by the supervisor, or not adhering to various criteria. Furthermore, requested meeting logs by the final year project Dr. Dimopoulos are listed in the end of the Appendix section.

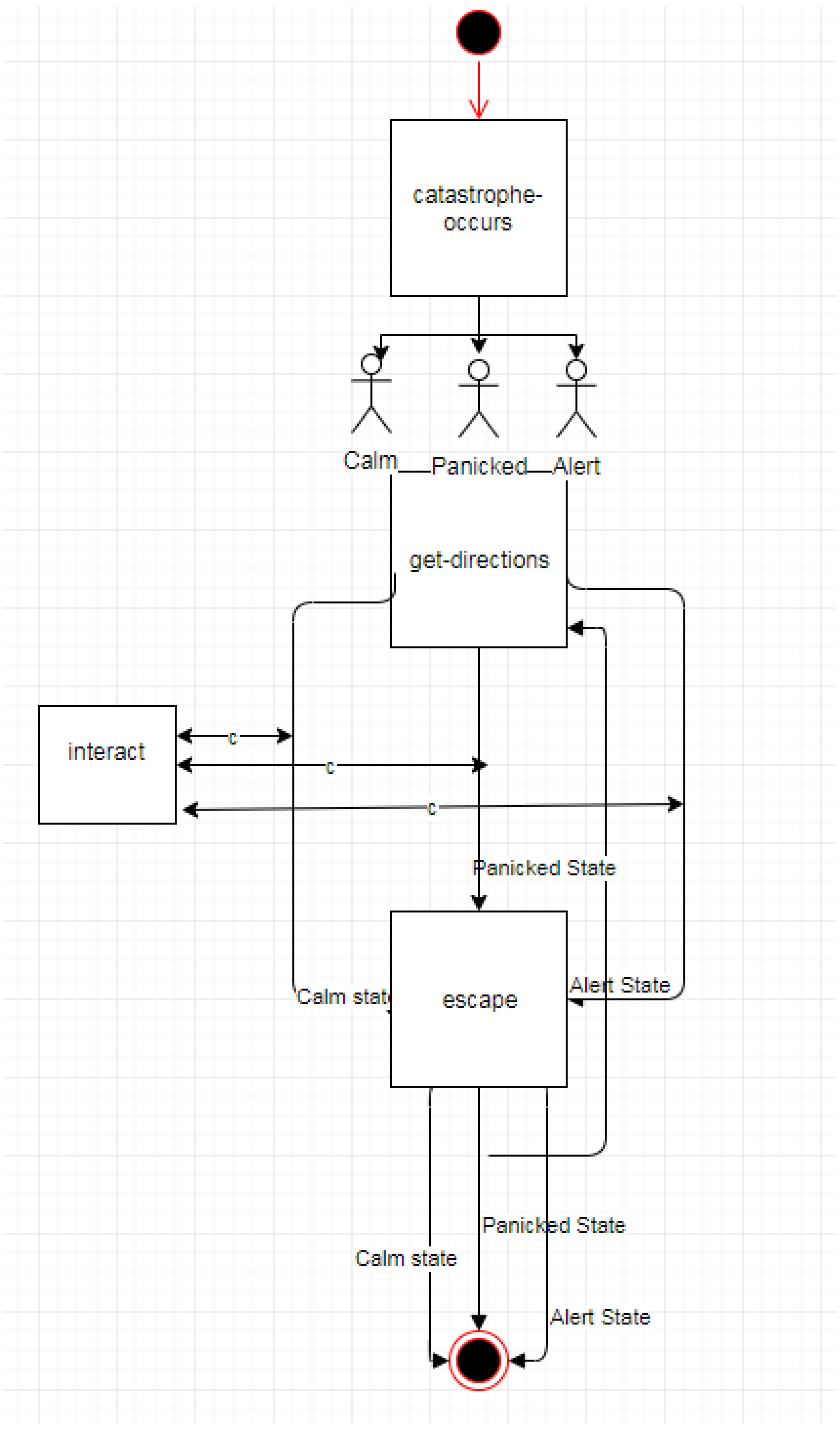


Figure #22. Experimental Flow Chart

The figure presented above was an early conception of how the user requirements could be captured in a diagram. It does not follow any UML standards to be classified as an indentified type of diagram, so it was deemed unnecessary for demonstrating the software logic of the system.

Meeting Logs

The meeting logs listed below follow a specific format under the instructions of 3rd Year Project Coordinator, Dr. Dimopoulos.

Date of (Skype) meeting with Supervisor:
3/11/2017   10:05 PM
Meeting Duration:

18minutes 34 seconds

Subject of Discussion:

System Description and Specification

Plan until Net Meeting:

Get Familiar with NetLogo

Self-reflection

How many hours did you work during that period?:

2-4

Are you personally satisfied with the effort since last meeting?:

Not really

Are you satisfied with the progress of your project overall so far?:

Somewhat

Date of (Skype) meeting with Supervisor:
20/11/2017   10:05 PM
Meeting Duration:

16minutes 35 seconds

Subject of Discussion:

Feedback and further directions

Plan until Next Meeting:

Get Familiar with NetLogo

Self-reflection

How many hours did you work during that period?:

2-4

Are you personally satisfied with the effort since last meeting?:

Not really

Are you satisfied with the progress of your project overall so far?:

Somewhat

Date of (Skype) meeting with Supervisor:
18/12/2017 10:05 PM
Meeting Duration:

8minutes 58 seconds

Subject of Discussion:

Feedback on previous work

Plan until Next Meeting:

Conduct system design and work on the Interim Report

Self-reflection

How many hours did you work during that period?:

5-6

Are you personally satisfied with the effort since last meeting?:

More than the previous week

Are you satisfied with the progress of your project overall so far?:

Somewhat

Date of (Skype) meeting with Supervisor:
23/1/2019 10:05 PM
Meeting Duration:

24minutes 33 seconds

Subject of Discussion:

Contact reestablishment, Overall project discussion and plans

Plan until Net Meeting:

<u>Self-reflection</u>

How many hours did you work during that period?:

26-28

Are you personally satisfied with the effort since last meeting?:

Yes

Are you satisfied with the progress of your project overall so far?:

Somewhat

Date of (Skype) meeting with Supervisor:
14/2/2019 10:05 PM
Meeting Duration:

18minutes 34 seconds

Subject of Discussion:

Progress report and Project finalization

Plan until Next Meeting:

Write the Final Report

<u>Self-reflection</u>

How many hours did you work during that period?:

35-38

Are you personally satisfied with the effort since last meeting?:

Yes, to a great degree

Are you satisfied with the progress of your project overall so far?:
Of course

Disclaimer: Due to the long-drawn duration of the project and the great gaps between the initial project attempt and the resuming of it, any data before 2019 may not be accurately recorded. Any meeting(s) that took place in person after fall of 2018 are also not recorded, since during these meetings there was no discussion about the project. Furthermore, any project work hours that are not tied to any meeting are not listed.