A

Major Project Report on

EMERGENCY EVACUATION SIMULATION USING ABMS

Submitted in partial fulfilment of the requirements for the award of the Degree of Bachelor of Technology

By

Chittoji Hithisha

(20EG105612)

Under The Guidance Of

Dr. N. Swapna Goud

Assistant Professor

Department of CSE

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

ANURAG UNIVERSITY VENTAPUR (V), GHATKESAR (M),

MEDCHAL (D), T.S 500088

(2023-24)

DECLARATION

I hereby declare that the Report entitled **"EMERGENCY EVACUATION SIMULATION USING ABMS"** submitted to the **Anurag University** in partial fulfilment of the requirements for the award of the degree of **Bachelor of Technology (B.Tech)** in **Computer Science and Engineering** is a record of original work done by me under the guidance of **Dr. N. Swapna Goud, Assistant Professor** and this report has not been submitted to any other University or Institution for the award of any other degree or diploma.

Place: Anurag University, Hyderabad — Chittoji Hithisha

Date: — (20EG105612)

ANURAG UNIVERSITY | SCHOOL OF ENGINEERING

CERTIFICATE

This is to certify that the project report entitled "**Emergency Evacuation Simulation Using ABMS"** being submitted by **Ms. Chittoji Hithisha** bearing the Hall Ticket number **20EG105612** in partial fulfilment of the requirements for the award of the degree of the **Bachelor of Technology** in **Computer Science and Engineering** to the Anurag University is a record of bonafide work carried out by them under my guidance and supervision from 2023 to 2024.

The results presented in this report have been verified and found to be satisfactory. The results embodied in this report have not been submitted to any other University or Institute for the award of any other degree or diploma.

Signature of Supervisor

Dr. N. Swapna Goud

Assistant Professor

Signature Dean, CSE

Dr. G. Vishnu Murthy

External Examiner

ACKNOWLEDGMENT

I would like to express my sincere thanks and deep sense of gratitude to project supervisor and Co-ordinator **Dr. N. SWAPNA GOUD, Assistant Professor, Department of Computer Science and Engineering,** Anurag University for her constant encouragement and inspiring guidance without which this project could not have been completed. Her critical reviews and constructive comments improved our grasp of the subject and steered me towards the fruitful completion of the work. Her patience, guidance and encouragement made this project possible.

I would like acknowledge our sincere gratitude for the support extended by **Dr. G. VISHNU MURTHY, Dean, Department of Computer Science and Engineering,** Anurag University. I also express my deep sense of gratitude to **Dr. V. V. S. S. S. BALARAM, Academic co-ordinator**. Project Co-ordinator and Project review committee members, whose research expertise and commitment to the highest standards continuously motivated me during the crucial stage of my project work.

I would like express my special thanks to **Dr. V. VIJAYA KUMAR, Dean School of Engineering,** Anurag University, for his encouragement and timely support in our B.Tech program.

Chittoji Hithisha

(20EG105612)

ABSTRACT

Fire disasters occur worldwide, prompting urgent movements of people seeking to escape danger. Evacuation drills are conducted to familiarize individuals with emergency procedures, but they often fall short of replicating real emergencies and can pose risks to participants. Modeling pedestrian motion and crowd dynamics during evacuations is crucial for enhancing human safety, building design, and evacuation protocols.

The Emergency evacuation simulation using ABMS focuses on indoor pedestrian evacuation during fire emergencies. By analyzing human behavior in real fire incidents documented in fire investigation reports, a generic agent-based evacuation model is developed. This model integrates common human behaviors identified in these reports, representing diverse agent types with specific characteristics. Interactions among agents and an evacuation timeline are simulated to accurately replicate human behavior and evacuation dynamics.

Validating of the model using real fire cases demonstrates its ability to mirror real-world evacuation scenarios. Simulation outcomes offer insights into casualty numbers, high-risk areas, egress choices, and evacuation times. Furthermore, the model evaluates the impact of building configurations, occupancy levels, and fire locations on evacuation outcomes, informing building designs, evacuation plans, and rescue strategies.

TABLE OF CONTENT

LIST OF FIGURES

LIST OF TABLES

1.INTRODUCTION

1.1 OVERVIEW

Emergency evacuations are pivotal moments where swift and effective actions can mean the difference between life and death. Whether it's a fire in a crowded stadium, a natural disaster in a city, or a terrorist attack in a public space, the ability to evacuate efficiently and safely is paramount. However, the complexity of human behavior in these high-stress situations presents significant challenges for emergency planners and responders. Traditional approaches to modeling and planning for evacuations often rely on simplified assumptions and do not adequately capture the dynamic interactions and decision-making processes of individuals during emergencies.

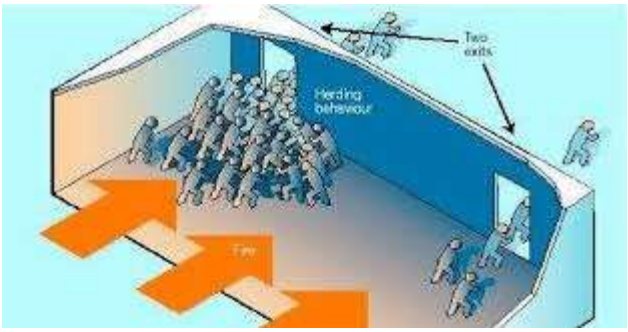

Figure 1.1 Herding Behaviour

To address these challenges, this project harnesses the power of Agent-Based Modeling (ABM) to simulate and analyze emergency evacuations. ABM provides a computational framework for representing individual agents with autonomous behaviors within a virtual environment. By modeling the actions and interactions of these agents, ABM enables the exploration of emergent phenomena and the examination of complex systems dynamics.

1.2 PURPOSE OF THE PROJECT

The purpose of the project is to analyse and improve emergency evacuation procedures , particularly focusing on mitigating the stampede effect during crises. By creating a simulation model that replicates the venue's layout and population dynamics, the project

aims to identify factors contributing to stampede casualties, such as lack of knowledge of the venue layout, seat locations, and demographic considerations. Through this analysis, the project seeks to provide insights and recommendations for enhancing survivability in crowd evacuations, ultimately aiming to increase safety and reduce casualties during emergencies.

1.3 MOTIVATION

The aim of this project is to utilize software simulation tools in order to achieve creating a multi-agents systems, comprised of intelligent software agents. To elaborate on that, the agents follow the two different strategies that are smart and follow.

Consequently, the aforementioned agents should possess the ability to mimic emotions expressed by actual human beings in the 'real' world. Such a system could be utilized by specialists in other fields to study human behaviour in states of heightened alertness and subsequently apply those findings in various contexts, such as designing the interiors of congregation rooms, theatres, cinemas, lecture halls, etc.

Ultimately, the aim is to develop a system that fully simulates the conditions that arise in hazardous areas populated with individuals of varying backgrounds, personalities, moods, and behaviours.

1.4 EXAMPLE ILLUSTRATION

Figure 1.4 Tragic stampede incident

The tragic stampede incident at a military recruitment event in Brazzaville, Republic of Congo, serves as a poignant real-life example of the complexities and consequences of crowd behavior during emergencies. Here's an analysis of the event focusing on the stampede

Incident Overview:

The stampede occurred during a military recruitment event at Ornado stadium in Brazzaville, resulting in the deaths of 31 people and injuries to 145 others. The tragedy prompted the Republic of Congo government to declare a national day of mourning and establish a crisis committee to oversee the response efforts. The government pledged to cover medical and funeral expenses for the victims and initiated an investigation to determine the cause of the stampede.

Factors Contributing to the Stampede:

High Demand: The event drew a large crowd of individuals eager to participate in the recruitment process, reflecting the significant unemployment challenges facing the country.

Limited Resources: The recruitment event may have been inadequately staffed or equipped to handle the volume of attendees, leading to overcrowding and chaos.

Impatience and Desperation: Some attendees, driven by a sense of urgency or desperation for employment opportunities, may have attempted to force their way into the venue, triggering the stampede.

Organizational Issues: Poor crowd management and communication by event organizers could have exacerbated the situation, contributing to confusion and panic among attendees.

Impact and Implications:

The stampede resulted in loss of life and widespread injuries, highlighting the urgent need for improved safety measures and crowd control protocols during public events. The incident underscores broader socioeconomic challenges facing the Republic of Congo, including high youth unemployment rates and pervasive poverty despite being an oil-producing nation. It emphasizes the importance of addressing underlying issues such as

access to employment opportunities, social support systems, and infrastructure development to prevent similar tragedies in the future.

Lessons Learned:

Enhanced Event Planning: Authorities and event organizers must prioritize safety and adequately plan for crowd management, particularly during high-demand events.

Public Awareness and Education: Efforts to educate the public on emergency procedures, crowd behavior, and the importance of patience and cooperation can help mitigate risks during large gatherings.

Socioeconomic Reforms: Addressing systemic issues such as unemployment, poverty, and inequality is essential for fostering social stability and preventing situations of desperation that can lead to stampedes and other emergencies.

2. LITERATURE SURVEY

1) "Agent-Based Modeling of Human Behavior during Building Evacuations: A Review" by Nilsson, D., Johansson, A., Frantzich, H., & Helbing, D. (2018)

A thorough examination of the use of agent-based modeling (ABM) techniques in simulating human behavior during building evacuations. The review delves into a wide range of ABM methodologies and their applications, providing insights into how these models can effectively capture the complexities of human decision-making and interaction dynamics in emergency scenarios. The article discusses key components of ABM, including the characterization of individual agents, modeling their decision-making processes, and representing their interactions within the evacuation environment. It emphasizes the importance of incorporating realistic agent characteristics, such as movement speed, awareness of the emergency, and social influences, to accurately simulate human behavior during evacuations. Additionally, the review evaluates various ABM approaches in the context of building evacuations, such as rule-based models, social force models, and cognitive models, highlighting their strengths and limitations. It also addresses methodological considerations and explores challenges and opportunities associated with ABM applications in building evacuation simulations. The article identifies areas for future research, such as integrating real-time data and sensor technologies, developing sophisticated agent decision-making algorithms, and exploring novel approaches to modeling social dynamics and crowd behavior. Overall, the review provides a comprehensive overview of the current state of research in this field and serves as a valuable resource for enhancing emergency preparedness and response strategies through ABM techniques. [1]

2) "Simulation of Human Evacuation in Built Environments: A Review of Current Approaches and Research Needs" by Chowdhury D., Alam M., & Zhang Y. (2017)

A seminal work in the domain of evacuation modeling, offering a meticulous examination of the methodologies employed to simulate human behavior during evacuation scenarios within built environments. Through a rigorous comparison of diverse modeling approaches such as agent-based, cellular automata, and continuum models, the paper not only illuminates the strengths and limitations inherent in each method but also underscores

the complexities involved in accurately capturing human dynamics in emergency situations. Furthermore, by pinpointing key research gaps, the paper steers attention towards areas ripe for exploration, advocating for a concerted effort to refine existing techniques and develop novel methodologies to enhance the precision and reliability of evacuation simulations. In doing so, it not only contributes to the advancement of theoretical knowledge but also holds profound implications for practical applications, facilitating the design of safer built environments and more effective emergency management strategies. [2]

3) "A Review of Crowd Evacuation Models" by Hu, X., & Liao, L. (2017)

A comprehensive survey of crowd evacuation models, emphasizing the computational techniques employed to simulate large-scale evacuations across diverse environments. The paper meticulously compares and contrasts various modeling methodologies, ranging from agent-based to macroscopic and microscopic models, providing insights into their respective capabilities and limitations. By delving into the applications of these models in analyzing crowd dynamics, evacuation strategies, and safety considerations, the review elucidates their significance in enhancing our understanding of complex evacuation scenarios. Furthermore, it not only identifies current trends but also anticipates future challenges in the realm of crowd evacuation modeling, paving the way for innovative research endeavors aimed at addressing emerging issues and refining existing methodologies. Through its thorough examination and forward-looking insights, this review serves as a valuable resource for researchers and practitioners involved in the development of evacuation strategies and the design of safer built environments. [3]

4) "Pedestrian and Evacuation Dynamics" by Helbing, D., Farkas, I., & Vicsek, T. (2002)

An exhaustive exploration of pedestrian and evacuation dynamics, spanning theoretical underpinnings, empirical observations, and computational modeling methodologies. This seminal work meticulously evaluates various modeling paradigms, ranging from agent-based and cellular automata to fluid-dynamic models, offering a comprehensive understanding of their utility in deciphering pedestrian behavior and refining evacuation strategies. Beyond theoretical insights, the book delves into the practical implications of

these models, elucidating their relevance in informing building design, urban planning, and emergency management practices. By synthesizing theoretical frameworks with real-world applications, this book not only enriches our understanding of pedestrian dynamics but also provides invaluable guidance for creating safer and more efficient built environments. [4]

2.1 EXISTING SYSTEM

In traditional fire evacuation procedures, occupants in a high-rise building or large commercial complex rely on established protocols and guidelines to evacuate safely during emergencies. These procedures typically involve predetermined evacuation routes, designated assembly points, and trained personnel to coordinate the evacuation process. Occupants are instructed to calmly exit the building via stairwells or designated fire exits, following the guidance of evacuation signs and emergency personnel. Fire drills and training sessions are conducted periodically to familiarize occupants with evacuation procedures and ensure a swift and orderly evacuation in case of a real emergency. While traditional fire evacuation procedures are effective, they may lack the dynamic and scenario-specific analysis provided by simulation software, which can help identify potential weaknesses, optimize evacuation routes, and improve overall preparedness for emergencies.

S.NO	STRATEGIES	ADVANTAGES	DISADVANTAGES
1	Traditional drill	Familiarity with emergency procedure	Lack of realism; participants may not take drills seriously
2	Multi-Agent systems	Accurate modeling of individual behaviors	Complexity of implementation; resource-intensive
3	Civil defense forces	Professional response to emergencies; can provide aid and support	Response time may be delayed; may not be able to handle largescale evacuations effectively

Table 2.1.1 Comparison of Existing Strategies

2.2 DISADVANTAGES OF EXISTING SYSTEM

Limited Realism: Traditional fire emergency evacuation simulations may lack realism due to their simplified representations of scenarios. These simulations often rely on basic models of human behavior, evacuation routes, and building layouts, which may not accurately capture the complexity of real-world emergencies. Factors such as crowd dynamics, individual decision-making, and environmental conditions are often oversimplified or overlooked, leading to discrepancies between simulated and actual evacuation outcomes.

Risk of Injury or Panic: Traditional evacuation simulations may fail to adequately simulate the risk of injury or panic among occupants during emergencies. Without realistic modeling of human behavior, including the potential for panic, pushing, and stampedes, these simulations may underestimate the dangers inherent in crowded evacuation scenarios. As a result, evacuation plans based on such simulations may not adequately account for the potential for injuries or fatalities during actual emergencies.

Difficulty in Assessing Evacuation Efficiency: Traditional evacuation simulations may struggle to accurately assess the efficiency of evacuation plans. Without detailed modeling of evacuation routes, occupant behavior, and environmental factors, it can be challenging to determine how quickly occupants can safely evacuate a building during an emergency. As a result, evacuation plans based on these simulations may not be optimized for efficiency, leading to delays or congestion during actual evacuations.

Inability to Simulate Complex Scenarios: Traditional fire emergency evacuation simulations may be limited in their ability to simulate complex scenarios, such as multi-floor evacuations, building collapses, or simultaneous emergencies in different areas of a building. These simulations often rely on simplified assumptions and scenarios, which may not accurately reflect the range of challenges that occupants and emergency responders may face during real emergencies. As a result, evacuation plans based on these

simulations may not adequately prepare for or mitigate the risks associated with complex emergency scenarios.

3. PROPOSED SYSTEM

3.1 PROPOSED SYSTEM

In our proposed method, we present a simulation framework to study emergency evacuation scenarios in a stadium environment. The framework incorporates various aspects such as agent characteristics, environment modeling, strategies for evacuation, and detailed modeling of crowd behavior.

Agent Characteristics:

Agents in the simulation exhibit diverse characteristics, including group age, sex, weight, and peripheral vision. These characteristics are sampled from distributions collected from Pordata, ensuring realism in the simulation. Additionally, agent panic level, speed, health, and muscular force are calculated based on these characteristics, enabling nuanced modeling of individual agent behavior.

Environment Modeling:

The simulation environment represents a stadium bench, which is part of a larger interconnected space. The focus is on the interactions and behaviors of individuals seated in the stadium during an emergency event. The stadium is divided into six distinct areas, with agents uniformly assigned to each area.

Trigger Event:

A random fire serves as the trigger event for the emergency scenario. The fire can occur in any location within the stadium and spreads uniformly with a speed adjustable by the user during simulation time.

Evacuation Strategies:

Three distinct evacuation strategies are modeled for agents to safely exit the stadium field during the emergency:

“Smart”

The “smart” strategy assumes that all survivors are equipped with the knowledge of the nearest exit location from where they are, and will try to proceed to the nearest possible exit

with the use of the best-first search algorithm. In the event that the designated exit has been blocked by the fire, they will locate the next nearest exit.

“Follow”

The “follow” strategy is used to model the ‘herding behaviour’ of survivors, as similar in the flocking library. In this strategy, survivors only have limited vision with no knowledge of the nearest exits, and they will follow the exact action of the other survivors 1 patch in front of them. If the fire is within their vision, they would run in the opposite direction from the fire. If they see an available exit, they will run straight for the exit.

Panic Behavior:

Agents start with a panic level of 1, which increases to level 3 as the fire approaches. Higher panic levels correspond to increased agent speed as they perceive the imminent threat.

This comprehensive framework provides a robust platform for studying emergency evacuation strategies and crowd behavior in stadium environments, facilitating insights into optimizing safety measures and response protocols.

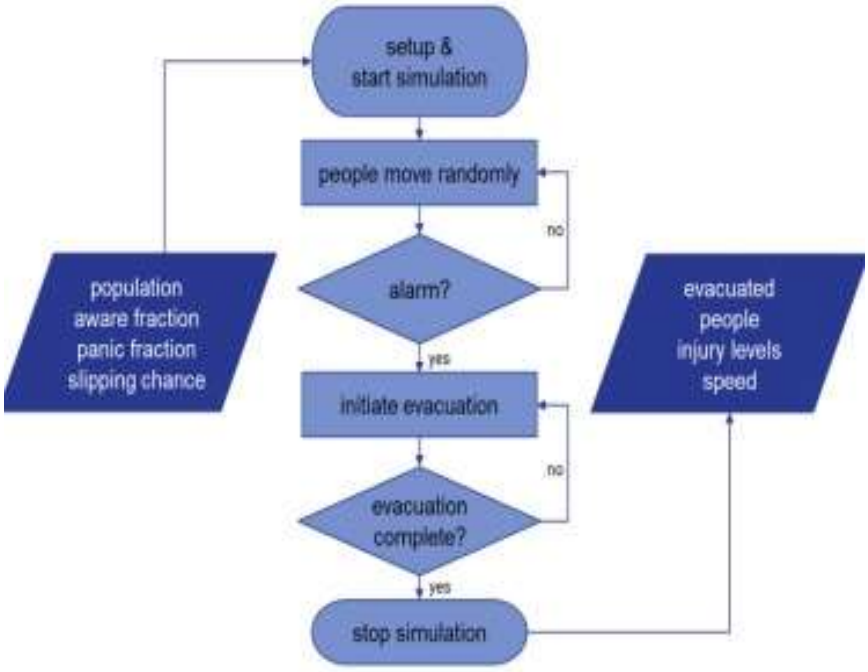

Figure 3.1 Flowchart

3.2 ADVANTAGES

Enhanced Safety Preparedness: By accurately modeling crowd behavior and evacuation strategies, the proposed method significantly enhances safety preparedness in stadiums during emergencies. This ensures that stadium operators and emergency responders are better equipped to handle various crisis scenarios effectively.

Risk Mitigation and Prevention: The simulation tool allows for the identification and mitigation of potential risks and hazards within stadium environments, enabling proactive measures to prevent emergencies or minimize their impact.

Optimized Emergency Response Plans: Through the simulation of different evacuation strategies and scenarios, stadium operators can develop and optimize emergency response plans tailored to specific stadium layouts and capacities. This leads to more efficient and effective evacuation procedures in real-life situations.

Improved Training and Preparedness: The simulation tool serves as a valuable training resource for stadium staff, security personnel, and emergency responders, enhancing their preparedness to manage emergencies such as fires, stampedes, or other critical incidents.

Enhanced Design and Layout: Insights gained from the simulations can inform the design and layout of stadiums to minimize congestion, improve crowd flow dynamics, and enhance overall safety during both normal operations and emergencies. This ensures that stadiums are built or renovated with safety considerations in mind, ultimately benefiting spectators and event organizers.

3.3 SYSTEM REQUIREMENTS

The system requirements for the development and deployment of the project as an applicationare specified in this section. These requirements are not be confused with the end-user systemrequirements. There are no specific, end-user requirements as the intended application is cross-platform and is supposed to work on devices of all form-factors and configurations.

3.3.1 SOFTWARE REQUIREMENTS

Operating System	:	Windows , MacOS
Technology	:	NetLogo
Code Development Tool	:	NetLogo 6.4.0 IDE

3.3.2 HARDWARE REQUIREMENTS

Processor	:	Any Update Processor
Ram	:	Min 4 GB
Hard Disk	:	Min 250 GB

4. IMPLEMENTATION

4.1 THE FUNDAMENTALS OF NETLOGO

The fundamentals of NetLogo typically refer to the basic concepts, principles, and components of the NetLogo programming language and environment. NetLogo is a multi-agent programming language and modeling environment primarily used for simulating complex systems. Its fundamentals include:

4.1.1 THE THREE TABS

The main window of NetLogo contains three tabs, i.e. the interface tab, the info tab and the code tab

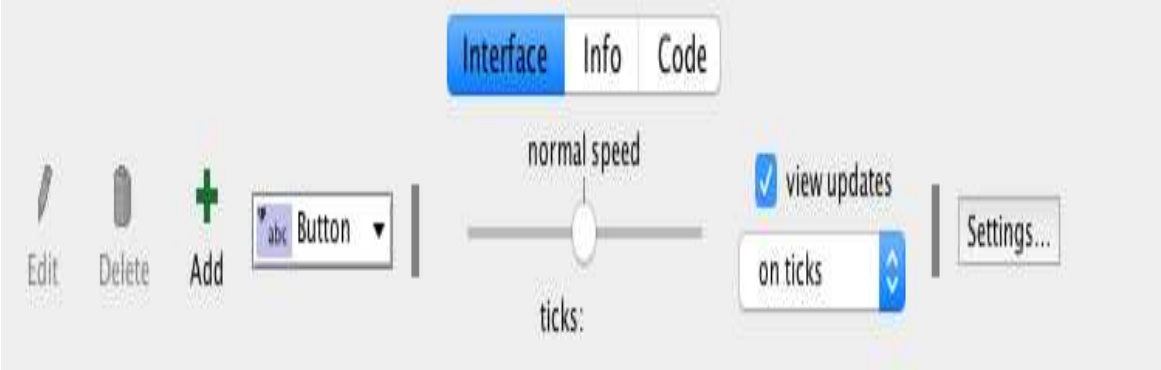

Figure 4.1.1 Tabs in NetLogo

Interface Tab: The Interface tab is where users interact with the model. It contains various user interface elements like buttons, sliders, switches, plots, monitors, and input fields.

- Buttons: These are clickable elements that trigger specific actions in the model. For example, a "setup" button initializes the model, while a "go" button starts or advances the simulation.
- Sliders: Sliders allow users to adjust parameters of the model dynamically. They can control variables such as speed, population size, or interaction rates.

- Switches: Switches are binary toggles that turn features on or off within the model.
- Plots: Plots visualize data generated during the simulation. They can display trends, distributions, or relationships between variables over time.
- Monitors: Monitors display the current value of a variable or expression during the simulation.
- Input Fields: Input fields allow users to input specific values or text into the model.

These elements collectively provide a user-friendly interface for controlling and observing the simulation.

Info Tab: The Info tab is used to provide documentation and information about the model. It typically contains details about the purpose of the model, its assumptions, how to use it, and any relevant references or citations. Documentation here can include explanations of key concepts, descriptions of variables and parameters, instructions for running the model, and interpretations of the results.

Code Tab: The Code tab contains the actual code that defines the behavior of the model. It is where users write NetLogo code using the NetLogo programming language. This tab includes procedures, functions, and other code constructs that define the rules governing agent behavior, environmental dynamics, and model logic. Users can define agent behaviors, set up initial conditions, specify model parameters, and create custom functions and procedures to implement specific features. While most of the code resides in this tab, some models may include snippets of code embedded within interface elements like plots for convenience or to enhance interactivity.

4.1.2 TYPES OF AGENTS

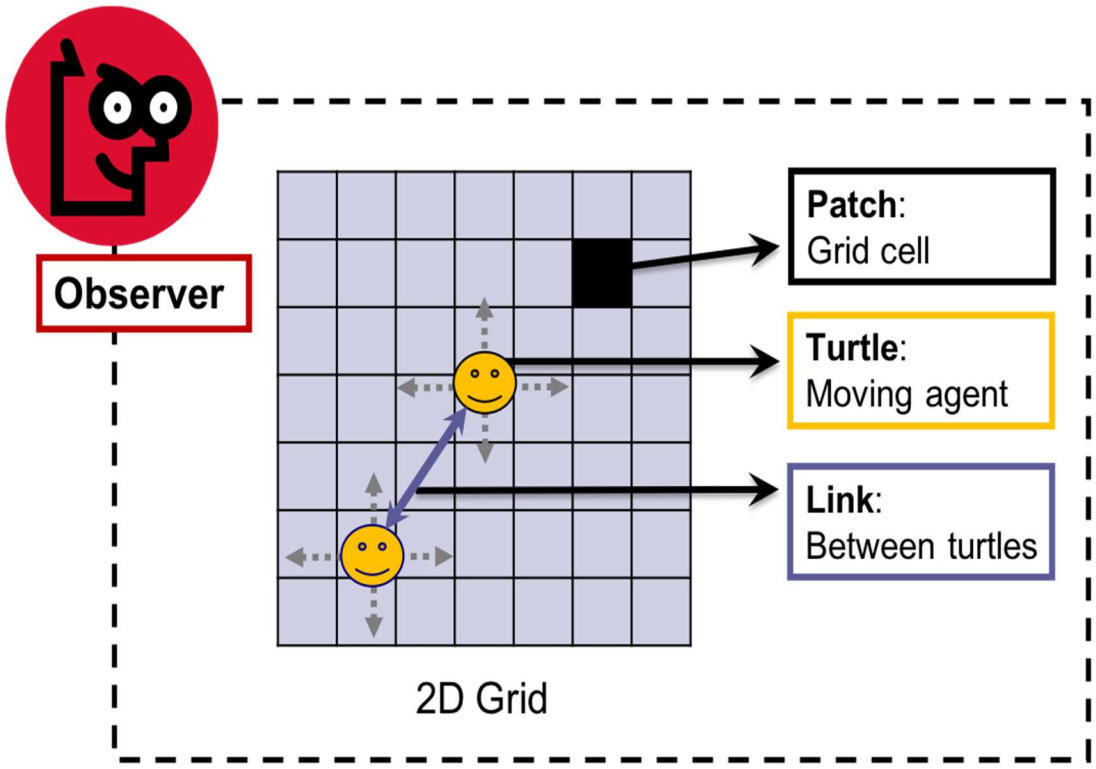

Figure 4.1.2 Agents in NetLogo

In NetLogo, the world is composed of four types of agents, each playing a distinct role in the simulation:

Turtle: Turtles are mobile agents capable of moving within the simulation environment. They represent individual entities that can interact with each other and their surroundings. Turtles can have attributes such as position, heading, color, size, and energy level, which can change dynamically as they interact with the environment and other agents. Turtles are often the primary focus of agent-based models in NetLogo, as they embody the individual entities whose behaviors and interactions drive emergent patterns in the simulation.

Patches: Patches constitute the two-dimensional grid that forms the NetLogo world. Each patch represents a square piece of "ground" over which turtles can move. Patches have properties such as color, elevation, and the presence of resources or obstacles. While patches themselves are not mobile agents, they serve as the background environment within which turtles operate. Patches can influence turtle behavior by providing information about local conditions and serving as sites for activities such as resource collection or territorial marking.

Links: Links are agents that establish connections between pairs of turtles. They can be either directed or undirected, depending on whether they have a specific directionality or not. Links enable the representation of relationships and interactions between individual turtles in the simulation. Examples of link-based interactions include social networks, communication channels, predator-prey relationships, or spatial dependencies between agents. Links allow for the modeling of complex network structures and the propagation of influences between connected agents.

Observer: The observer is a unique agent in NetLogo that does not have a specific location within the world. It serves as the central controller or "conductor" of the entire simulation. The observer oversees the execution of the model, coordinates interactions between agents, and manages the overall state of the simulation. While not directly represented within the simulation environment, the observer plays a crucial role in orchestrating the behavior and interactions of turtles, patches, and links.

Understanding the roles and interactions of these four types of agents is essential for designing and implementing agent-based models in NetLogo. Each type contributes to the dynamic and emergent properties of the simulation, allowing for the exploration of complex systems and phenomena.

4.1.3 VARIABLES

Variables are places to store values (such as numbers). A variable can be a global variable, a turtle variable, a patch variable, a link variable, or a local variable (local to a procedure). To change the value of a variable you can use the set command. If you don't set the variable to any value, it starts out storing a value of zero.

Global Variables: Global variables have a single value shared by all agents in the simulation. They can be declared in either the Interface tab using interface elements like switches, sliders, choosers, or input boxes, or in the Code tab using the globals keyword followed by a list of variable names.Global variables are accessible to all agents, allowing them to read or modify their values as needed.

Turtle, patch, and link variables: Each turtle has its own value for every turtle variable, each patch has its own value for every patch variable, and each link has its own value for every link variable. Turtle, patch, and link variables can be built-in or defined by the user.

- **Built-in** variables: For example, all turtles and all links have a color variable, and all patches have a pcolor variable. If you set this variable, the corresponding turtle, link or patch changes color. Other built-in turtle variables are xcor, ycor, and heading. Other built-in patch variables include pxcor and pycor. Other built-in link variables are end1, end2, and thickness.
- **User-defined** turtle, patch and link variables: You can also define new turtle, patch or link variables using the turtles-own, patches-own, and links-own keywords respectively.

Local Variables: Local variables are defined and used within the context of a specific procedure or part of a procedure. They are declared using the let command and are typically used for temporary storage or calculations within a procedure. Local variables defined at the top of a procedure exist throughout the entire procedure, while those defined within square brackets, such as inside an ask block, exist only within that specific context.

4.1.4 SYNCHRONIZATION

When you ask a set of agents to run more than one command, each agent must finish all the commands in the block before the next agent starts. One agent runs all the commands, then the next agent runs all of them, and so on. As mentioned before, the order in which agents are chosen to run the commands is random. To be clear, consider the following code:

```
ask turtles [
    forward random 10  ;; move forward a random number of steps (0–9)
    wait 0.5           ;; wait half a second
    set color blue     ;; set your color to blue
      ]
```

The first (randomly chosen) turtle will move forward some steps, she will then wait half a second, and she will finally set her color to blue. Then, and only then, another turtle will start doing the same; and so on until all turtles have run the commands inside ask without being interrupted by any other turtle. The order in which turtles are selected to run the commands is random. If you want all turtles to move, and then all wait, and then all become blue, you can write it this way:

```
ask turtles [ forward random 10 ]
ask turtles [ wait 0.5 ]          ;; note that you will have to wait
ask turtles [ set color blue ]  ;; (0.5 * number-of-turtles) seconds
```

Finally, you can make agents execute a set of commands in a certain order by converting the agentset into a list. There are three primitives that help you do this: sort, sort-by and sort-on.

4.1.5 BREEDS

NetLogo allows you to have different types of turtles and different types of links. There are called breeds. This is a built-in turtle and link variable. It holds the agentset of all turtles or links of the same breed as this turtle or link. Breeds are defined with the syntax:

breed [plural-name singular-name]

4.1.6 TICKS AND PLOTTING

In NetLogo, time is typically represented in discrete steps called "ticks." These ticks serve as a fundamental unit of time progression within a simulation. NetLogo includes a built-in tick counter that keeps track of the number of ticks that have passed during the simulation. The current tick count is displayed above the view in the NetLogo interface. Here's a breakdown of how to use the tick counter effectively in NetLogo:

Resetting the Tick Counter: To initialize the tick counter and set up all plots in the interface, you use the reset-ticks command. This command resets the tick counter to zero and triggers the setup code for all plots, ensuring that the initial state of the world is plotted accurately.

Advancing Time with Ticks: After resetting the tick counter, you can use the tick command to advance time by one tick. Each time you invoke the tick command, the tick counter is incremented by one, and all plots are updated to reflect any changes in the simulation.

Using Plots with the Tick Counter: NetLogo allows you to embed code directly inside plots, enabling you to visualize data dynamically as the simulation progresses. Each plot and its pens have setup and update code fields where you can write commands to customize their appearance and behavior. We can write code directly within these fields in the interface, without needing to access the code tab. The setup-plots and update-plots

primitives can be used to execute the setup and update code fields in every plot and pen, respectively. However, in models that use the tick counter, these primitives are often unnecessary because they are automatically triggered by tick-related commands.

The Overall idea of the implementation

Agent Characteristics: The agents in the simulation are characterized by various attributes such as age, sex, weight, and peripheral vision. These attributes influence the agents' behavior and response during the emergency.

Environment: The environment is modeled as a stadium bench, which forms part of a larger interconnected space. The focus is on the interactions and movements of individuals seated on the bench during an emergency event.

Trigger Event: The trigger event corresponds to a random fire outbreak within the stadium. The fire spreads uniformly, and its speed can be adjusted by the user during simulation time.

Agent Strategies: Three different strategies are modeled for agents to safely exit the stadium field during the emergency:

- **Smart Strategy:** Agents have full knowledge of all possible exits and select the nearest available exit based on their current position.
- **Follow Strategy:** Agents have limited vision and follow the actions of other agents nearest to them. They move towards exits within their range of vision or move away from the fire if detected.

Modeling: The simulation models crowded scenarios where herding or flocking behaviors may occur. Traumatic asphyxia due to external compression of the thorax is considered, highlighting the potential risks of overcrowded areas during emergencies.

Force Exertion: The force exerted on a patch is calculated based on the combined mass and speed of agents occupying the patch. This helps model the pressure exerted by crowds on specific areas, which can affect movement and evacuation.

Agent Health and Panic: Agent health is determined by their mass (which encapsulates age group and gender), speed, and a tuning parameter. Agents may die if they come into contact with fire or if the force exerted on their current patch exceeds their health level. Panic levels increase as agents perceive the fire, leading to increased speed and urgency in their movements.

4.2 IMPLEMENTATION

CODE:

```
globals [
  blue-escapees
  cyan-escapees
  yellow-escapees
  red-escapees
  beige-escapees
  green-escapees
  blue-fire-deaths
  cyan-fire-deaths
  yellow-fire-deaths
  red-fire-deaths
  beige-fire-deaths
  green-fire-deaths
  blue-stampede-deaths
  cyan-stampede-deaths
  yellow-stampede-deaths
  red-stampede-deaths
  beige-stampede-deaths
  green-stampede-deaths
  female-escapees
  female-fire-deaths
  female-stampede-deaths
  male-escapees
  male-fire-deaths
  male-stampede-deaths
  child-escapees
  child-fire-deaths
```

```
  child-stampede-deaths
  adult-escapees
  adult-fire-deaths
  adult-stampede-deaths
  elderly-escapees
  elderly-fire-deaths
  elderly-stampede-deaths
  oldgoal
]
breed [survivors survivor]
breed[doors door]
patches-own [
  distance1
  distance2
  distance3
  distance4
  distance5
  distance6
  distance7
  distance8
  distance9
  distance10
  distancefire
]
survivors-own [
  goal
  health
  base-speed
  speed  ; impacted by status and patch pressure (sum surrounding patch pressure)
  vision
  gender
```

age
mass
panic
; reaction-time
; collaboration
; insistence
Knowledge]

EXPLANATION:

1. Globals: These are global variables that are accessible from anywhere in the program. In this case, they are defining various attributes related to escapees, deaths, and other factors in the simulation. For example, blue-escapees, cyan-escapees, etc., seem to represent the number of escapees of different colors (perhaps indicating different groups or categories). Similarly, blue-fire-deaths, cyan-fire-deaths, etc., represent the number of deaths by fire for each group.

2. Breeds: This section defines agent breeds, which are essentially different types of agents in the simulation. In this case, survivors and doors are being defined as breeds. survivors likely represent individuals trying to escape, while doors represent the physical exits they're trying to reach.

3. Patches-own: This section defines attributes specific to patches in the simulation environment. In agent-based modeling, patches typically represent the environment where agents interact. Attributes like distance1, distance2, etc., could indicate distances from reference points within the environment. The attribute "distancefire" likely represents the distance of each patch from a fire source, which could be crucial for simulating fire evacuation scenarios.

4. Survivors-own: This section defines attributes specific to the "survivors" breed. Each individual survivor in the simulation will have these attributes. Some of the attributes listed include:

- goal: The destination or objective the survivor is trying to reach.
- health: Represents the health status of the survivor.
- base-speed: Base speed of the survivor.
- speed: Current speed of the survivor, which can be affected by various factors like panic or patch pressure.
- vision: The range of vision for the survivor.
- gender: Gender of the survivor.
- age: Age of the survivor.
- mass: Mass of the survivor.
- panic: Level of panic experienced by the survivor.
- knowledge: Amount of information or knowledge possessed by the survivor.

CODE:

```
to move-normal
  ask survivors [
    let next-patch 0
    ifelse goal = door 14178 [set next-patch min-one-of neighbors [distance1]] [
      ifelse goal = door 14179 [set next-patch min-one-of neighbors [distance2]] [
        ifelse goal = door 14180 [set next-patch min-one-of neighbors [distance3]] [
          ifelse goal = door 14181 [set next-patch min-one-of neighbors [distance4]] [
            ifelse goal = door 14182 [set next-patch min-one-of neighbors [distance5]] [
              ifelse goal = door 14183 [set next-patch min-one-of neighbors [distance6]] [
                ifelse goal = door 14184 [set next-patch min-one-of neighbors [distance7]] [
                  ifelse goal = door 14185 [set next-patch min-one-of neighbors [distance8]] [
                    ifelse goal = door 14186 [set next-patch min-one-of neighbors [distance9]] [
                      ifelse goal = door 14187 [set next-patch min-one-of neighbors [distance10]]
[]]]]]]]]]]
```

```
  repeat speed [
    while [ [pcolor] of next-patch != grey] [
      ask next-patch [
        set distance1 10000000
        set distance2 10000000
        set distance3 10000000
        set distance4 10000000
        set distance5 10000000
        set distance6 10000000
        set distance7 10000000
        set distance8 10000000
        set distance9 10000000
         set distance10 10000000
      ]
      ifelse goal = door 14178 [set next-patch min-one-of neighbors [distance1]] [
        ifelse goal = door 14179 [set next-patch min-one-of neighbors [distance2]] [
          ifelse goal = door 14180 [set next-patch min-one-of neighbors [distance3]] [
            ifelse goal = door 14181 [set next-patch min-one-of neighbors [distance4]] [
              ifelse goal = door 14182 [set next-patch min-one-of neighbors [distance5]] [
                ifelse goal = door 14183 [set next-patch min-one-of neighbors [distance6]] [
                  ifelse goal = door 14184 [set next-patch min-one-of neighbors [distance7]] [
                    ifelse goal = door 14185 [set next-patch min-one-of neighbors [distance8]]
[
                      ifelse goal = door 14186 [set next-patch min-one-of neighbors [distance9]]
[
                        ifelse goal = door 14187 [set next-patch min-one-of neighbors
[distance10]] []]]]]]]]]
      ]
      if not patch-overcrowded? next-patch [ move-to next-patch ]
    ]
    if any? doors-here [
```

```
    ifelse color = blue
    [ set blue-escapees blue-escapees + 1]
    [ ifelse color = cyan
     [ set cyan-escapees cyan-escapees + 1 ]
     [ ifelse color = yellow
       [ set yellow-escapees yellow-escapees + 1 ]
       [ ifelse color = red
         [ set red-escapees red-escapees + 1 ]
         [ ifelse color = 29
          [ set beige-escapees beige-escapees + 1 ]
          [ set green-escapees green-escapees + 1 ]
         ]
       ] ] ]
     ifelse gender = "female"
    [ set female-escapees female-escapees + 1 ]
    [ set male-escapees male-escapees + 1 ]
    ifelse age = "child"
    [ set child-escapees child-escapees + 1 ]
    [ ifelse age = "adult"
     [ set adult-escapees adult-escapees + 1 ]
     [ set elderly-escapees elderly-escapees + 1
    die
   ]
End
```

EXPLANATION:

The "move-normal" procedure is pivotal for simulating the typical movement behavior of survivors toward their designated goals, usually exit doors. Let's dissect each part of the procedure:

1. Iterating Over Survivors: It begins by looping through all survivor agents present in the simulation, ensuring that each survivor's movement is considered.

2. Determining Next Patch: For each survivor, it identifies the next patch they should move to based on their designated goal. This determination is made by finding the neighboring patch with the shortest distance to the goal door.

3. Movement Loop: This section initiates a loop to facilitate the movement of the survivor toward the next patch. It repeats the movement step a certain number of times, where the number of repetitions is determined by the current speed of the survivor. While the next patch is not obstructed by obstacles (indicated by it not being grey), it clears the distances to the goal doors on that patch. It then recalculates the next patch based on the updated distances to the goal doors.

4. Moving to Next Patch: Once a suitable next patch is identified (i.e., not overcrowded), the survivor proceeds to move to that patch.

5. Escaping Through Doors: If a survivor reaches a patch containing an exit door, they are considered to have successfully escaped. The procedure updates escapee counters based on survivor attributes such as color, gender, and age. The survivor is then removed from the simulation (dies), as their goal of escaping has been achieved.

CODE:

```
to follow-crowd
 ;ask patches [set distancefire distance (min-one-of patches with [pcolor = orange] [distance myself]) ]
 ask survivors [
   let next-patch 0
   ifelse any? patches in-radius vision with [any? doors-here] [
       set goal min-one-of doors [distance myself]
       ifelse goal = door 14178 [set next-patch min-one-of neighbors [distance1]] [
         ifelse goal = door 14179 [set next-patch min-one-of neighbors [distance2]] [
```

```
          ifelse goal = door 14180 [set next-patch min-one-of neighbors [distance3]] [
           ifelse goal = door 14181 [set next-patch min-one-of neighbors [distance4]] [
            ifelse goal = door 14182 [set next-patch min-one-of neighbors [distance5]] [
             ifelse goal = door 14183 [set next-patch min-one-of neighbors [distance6]] [
              ifelse goal = door 14184 [set next-patch min-one-of neighbors [distance7]]
[
               ifelse goal = door 14185 [set next-patch min-one-of neighbors [distance8]]
[
                ifelse goal = door 14186 [set next-patch min-one-of neighbors
[distance9]] [
                 ifelse goal = door 14187 [set next-patch min-one-of neighbors
[distance10]] []]]]]]]]]
        while [ [pcolor] of next-patch != grey] [
         ask next-patch [
          set distance1 10000000
          set distance2 10000000
          set distance3 10000000
          set distance4 10000000
          set distance5 10000000
          set distance6 10000000
          set distance7 10000000
          set distance8 10000000
          set distance9 10000000
          set distance10 10000000
         ]
        ifelse goal = door 14178 [set next-patch min-one-of neighbors [distance1]] [
          ifelse goal = door 14179 [set next-patch min-one-of neighbors [distance2]] [
           ifelse goal = door 14180 [set next-patch min-one-of neighbors [distance3]] [
            ifelse goal = door 14181 [set next-patch min-one-of neighbors [distance4]] [
             ifelse goal = door 14182 [set next-patch min-one-of neighbors [distance5]] [
```

```
                ifelse goal = door 14183 [set next-patch min-one-of neighbors [distance6]]
[
                 ifelse goal = door 14184 [set next-patch min-one-of neighbors [distance7]]
[
                  ifelse goal = door 14185 [set next-patch min-one-of neighbors
[distance8]] [
                   ifelse goal = door 14186 [set next-patch min-one-of neighbors
[distance9]] [
                    ifelse goal = door 14187 [set next-patch min-one-of neighbors
[distance10]] []]]]]]]]]
        ]
 ifelse is-patch? patch-at-heading-and-distance (180 + heading) vision and is-patch? patch-
at-heading-and-distance (90 + heading) vision and is-patch? patch-at-heading-and-
distance (270 + heading) vision [
       ifelse [pcolor] of patch-ahead vision = orange or [pcolor] of patch-at-heading-and-
distance (180 + heading) vision = orange or [pcolor] of patch-at-heading-and-distance (90
+ heading) vision = orange or [pcolor] of patch-at-heading-and-distance (270 + heading)
vision = orange [
         set next-patch max-one-of neighbors [distancefire]
         while [ [pcolor] of next-patch != grey] [
          ask next-patch [set distancefire 0]
          set next-patch max-one-of neighbors [distancefire]
         ]
         if not patch-overcrowded? next-patch [ move-to next-patch ]
        ][
         if any? turtles-on neighbors [
          let avg mean [heading] of turtles-on neighbors
          ifelse avg <= 90 [set heading 90][
           ifelse avg <= 180 [set heading 180][
            ifelse avg <= 270 [set heading 270][
             set heading 0
```

```
            ] ] ]
          if [pcolor] of patch-ahead 1 = gray [fd 1]
        ] ]
End
```

EXPLANATION:

The "follow-crowd" procedure plays a crucial role in simulating survivor behavior when they opt to follow the crowd instead of moving independently toward an exit. Let's break down how it operates:

1. Iterating Over Survivors: It begins by iterating over all survivor agents present in the simulation, ensuring that each survivor's behavior is considered.

2. Determining Next Patch Based on Goal and Vision: If any patch within the survivor's vision radius contains an exit door, the survivor resets its goal to be the nearest door. This ensures that survivors prioritize escaping toward nearby exits. The survivor then determines the next patch to move to based on its updated goal, taking into account the new objective of reaching the nearest exit.

3. Moving Toward Next Patch: survivor enters a loop to move toward the next patch. Within this loop, it checks if the next patch is not obstructed by obstacles (indicated by it not being grey). If the patch is clear, it clears the distances to goal doors on that patch to recalculate the path to the exit. It then recalculates the next patch based on the updated distances to goal doors.

4. Handling Fire Avoidance and Crowd Movement: If the survivor detects fire within its vision range or in specific directions (ahead, left, right), it adjusts its movement to avoid the fire. This is done by selecting the patch with the maximum distance to fire and moving toward it. If the survivor detects fire within its vision range or in specific directions (ahead, left, right), it adjusts its movement to avoid the fire. This is done by selecting the patch with the maximum distance to fire and moving toward it. If the selected patch is not obstructed, the survivor

moves to that patch to avoid the fire. Additionally, if there are other survivors nearby, the survivor aligns its heading with the average heading of those nearby survivors. This simulates the behavior of following the crowd, which can provide safety in numbers during emergencies.

5. Movement Loop and Update of Distance to Fire: The movement loop continues until the survivor reaches a suitable patch or successfully avoids the fire. If the patch is not overcrowded, the survivor moves to it, ensuring progress toward the goal of reaching a safe location.

CODE:

```
to spread-fire
  ask patches with [pcolor = orange] [
    if any? neighbors with [pcolor = gray] [
      let spread-probability random-float 1
      if spread-probability < fire-spread-rate [
        ask one-of neighbors with [pcolor = gray] [ set pcolor orange ]
      ]
    ]
  ]
end
```

EXPLANATION:

1. to spread-fire: This line starts the definition of a procedure named `spread-fire`. This procedure governs the spread of fire in the simulation.

2. ask patches with [pcolor = orange] : This line initiates a loop over all patches that are currently colored orange, indicating they are on fire.

3. if any? neighbors with [pcolor = gray]: Within the loop, this line checks if any neighboring patches of the currently examined patch are gray, indicating they are not yet on fire.

4. let spread-probability random-float 1: This line generates a random float between 0 and 1 and assigns it to the variable `spread-probability`. This variable represents the probability of the fire spreading to a neighboring gray patch.

5. if spread-probability < fire-spread-rate : This line checks if the randomly generated `spread-probability` is less than the `fire-spread-rate` parameter. If so, it proceeds to spread the fire to a neighboring gray patch.

6. ask one-of neighbors with [pcolor = gray] [set pcolor orange]: This line selects one random neighboring gray patch and changes its color to orange, indicating that it is now on fire.

7. end: This line marks the end of the procedure definition.

5.EXPERIMENTAL RESULTS

5.1 EXPERIMENT SETUP

The toolkit used in this project is NetLogo. NetLogo is a versatile and user-friendly multi-agent modeling and simulation environment that enables researchers, educators, and professionals to explore and understand complex systems. Developed at Northwestern University's Center for Connected Learning and Computer-Based Modeling, NetLogo provides a powerful platform for creating agent-based models, where individual agents interact with their environment and other agents according to predefined rules. Its intuitive interface, featuring a combination of code editor and graphical modeling tools, allows users to easily design, implement, and visualize simulations of diverse phenomena ranging from social dynamics and ecology to economics and epidemiology. With its extensive library of pre-built models and its support for customization and extension through programming, NetLogo serves as an invaluable tool for investigating emergent behaviors, testing hypotheses, and gaining insights into the dynamics of complex systems. Moreover, its open-source nature fosters a vibrant community of users who contribute to its development and share their models and findings, making NetLogo a collaborative hub for interdisciplinary research and education. The following the steps for setting up Netlogo:

Download: Go to the official NetLogo website (ccl.northwestern.edu/netlogo/) and navigate to the download section. Choose the appropriate version for your operating system (Windows, macOS, or Linux) and download the installer file.

Install: Once the download is complete, run the installer file. Follow the on-screen instructions to install NetLogo on your computer. The installation process typically involves accepting the license agreement, choosing the installation directory, and completing the installation.

Launch: After installation, you can launch NetLogo by finding it in your list of installed applications or by double-clicking the desktop shortcut (if you chose to create one during installation).

Explore: When you open NetLogo, you'll be presented with the NetLogo interface, which consists of several components such as the Code tab (where you write code for your models), the Interface tab (where you design your simulation interface), the Information tab (where you can find documentation and resources), and the Graphics tab (where you visualize your simulation). Take some time to familiarize yourself with the interface and explore the example models provided.

Learn: NetLogo comes with extensive documentation, including tutorials and guides to help you get started. You can access these resources from the Information tab or visit the official NetLogo website for additional learning materials, user forums, and support.

Start Modeling: Once you're comfortable with the basics, you can start creating your own models. Begin by defining the agents, their behaviors, and the rules governing their interactions. Experiment with different parameters and settings to observe how your model behaves and iteratively refine it to match your research or educational objective

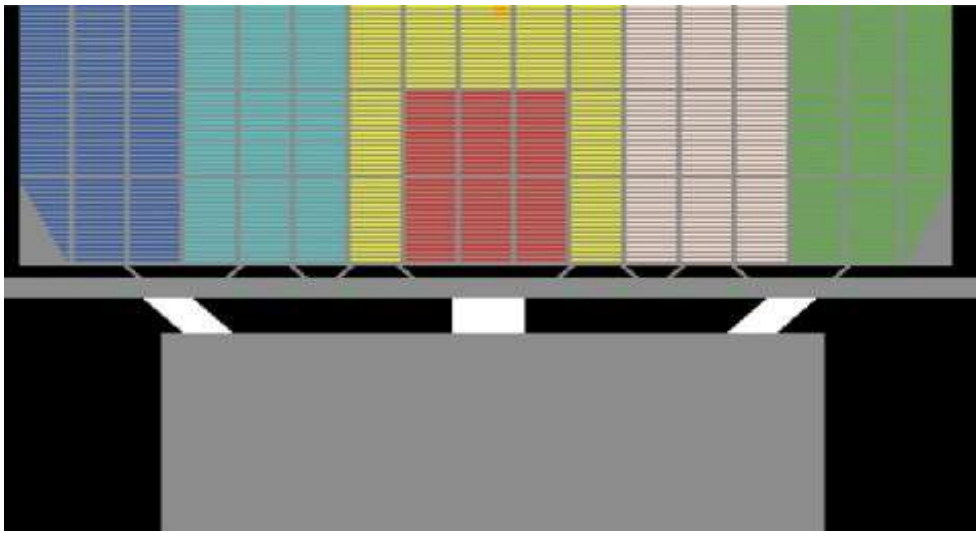

Figure 5.1 Environment setup

5.2 PARAMETERS

The below are the parameters described for the agents in the simulation

PARAMETERS	DESCRIPTION
Age Group	Normal Distribution: 3 Categories Child Adult Elderly
Gender	Male 48% Female 52%
Speed	Each agent has a base walking speed depending on their age category: Child: 038m/s Adult: Uniform distribution between 1.50m/s and 1.51m/s Elderly: Uniform distribution between 1.30m/s and 1.40m/s
Max vision	Uniform distribution 0 (vision can be extremely poor due to natural blindness or onset of smoke) and a maximum that can be set between 20 and 100
Panic level (1 to 3)	Level 1 – All agents base Level 2 – Fire in agent's vision. Agents speed increase fast pace (1.80m/s) Level 3 – Fire nearer (half the distance that the agent can see). Agent's speed increases to a running speed (2.50m/s)
Mass	Agent mass drawn from a normal distribution depending on their age category and gender Female child mean = 35kg Male child mean = 40kg Adult/Elderly mean = 57.7kg

Table 5.2.1 Parameter

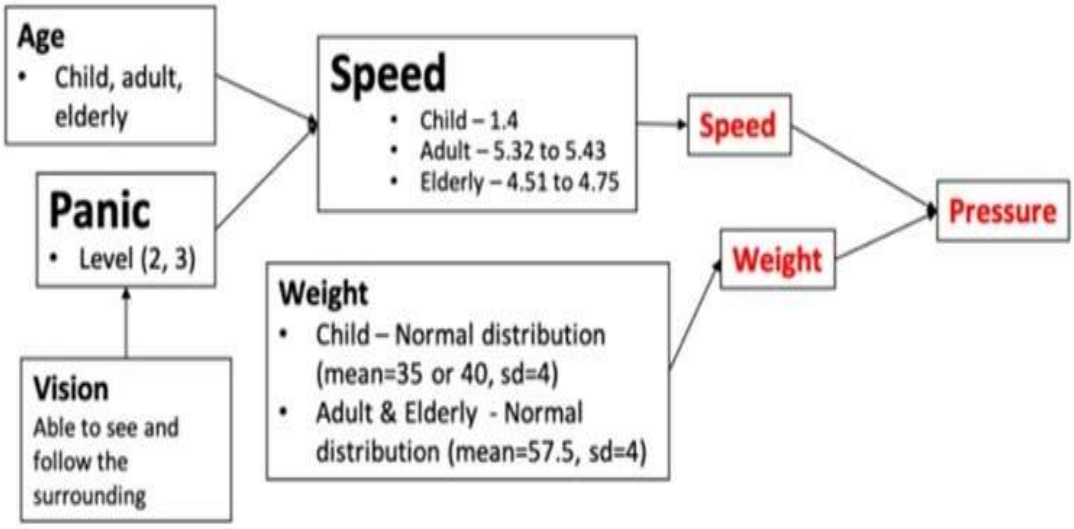

Figure 5.2 Parameters

5.3 PARAMETER FORMULAS:

Force / Pressure exerted in a patch p as :

$$F_p = \sum_{a \in A} mass_a \times speed_a ,$$

Health:

$$heath_a = mass_a \times speed_a \times threshold$$

6. DISCUSSION OF RESULTS

Figure 6.1 Spreading of fire

As the fire spreads throughout the stadium the number of available exits decreases. In the above scenario we can see that the Available Exits are 0.

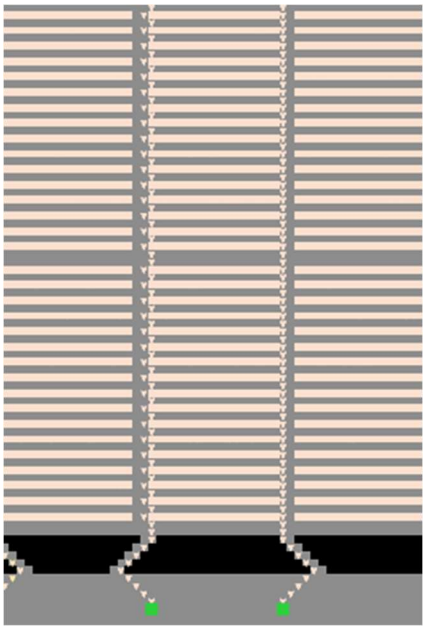

Figure 6.2 Escapese

Fire Spread: If the simulation includes a fire event, patches might represent areas that are susceptible to fire spread. In this case, patches could change color or state to indicate they

are on fire, and neighboring patches might be affected over time, simulating the spread of fire through the environment.

Emergency Exits: Patches might represent locations of emergency exits in the stadium environment. In an emergency situation, such as a fire, patches representing exits could become targets for agents trying to escape. Patches might change color or have special properties to indicate they are exits, and agents might be programmed to navigate towards these patches to evacuate the stadium.

Dynamic Environment: Patches could represent dynamic elements of the environment that change over time. For example, patches might represent movable obstacles or barriers that open up to create escape routes during emergencies. In this case, patches might change properties or positions based on certain triggers or events in the simulation, allowing agents to escape through newly opened pathways.

For a fairer comparison of results, we fix the origin location of the fire and we run a behaviour space for 100 times to confirm the results plotted.

"Smart" Strategy Without panic

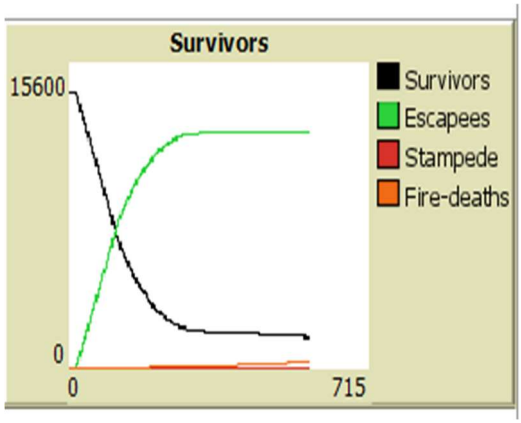

Figure 6.3 Survivors

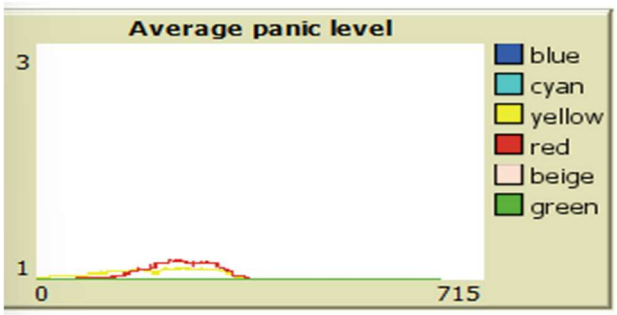

Figure 6.4 Average panic levels

Figure 6.5 Escapees by color

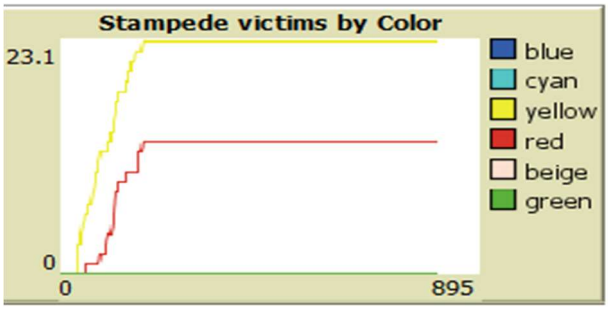

Figure 6.6 Stampede victims

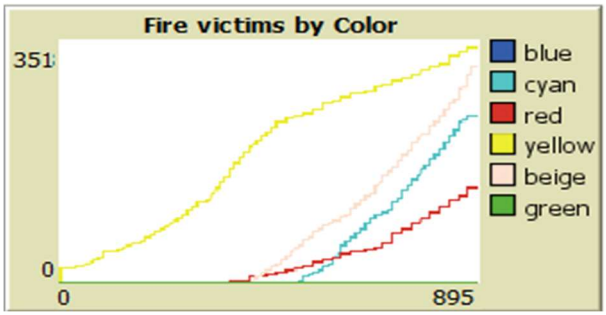

Figure 6.7 Fire victims

When there is no sense of panic, there are zero deaths resulting from stampedes - a clear contrast between Figure 6.1 and 6.2 Red agents (where the VIPs are seated) can escape scathe free with 100% survival rate (See Figure 6.2). Interestingly, less victims die in the fire in the scenario with panic due to the sense of urgency resulting in increased speed.

"Follow" Strategy

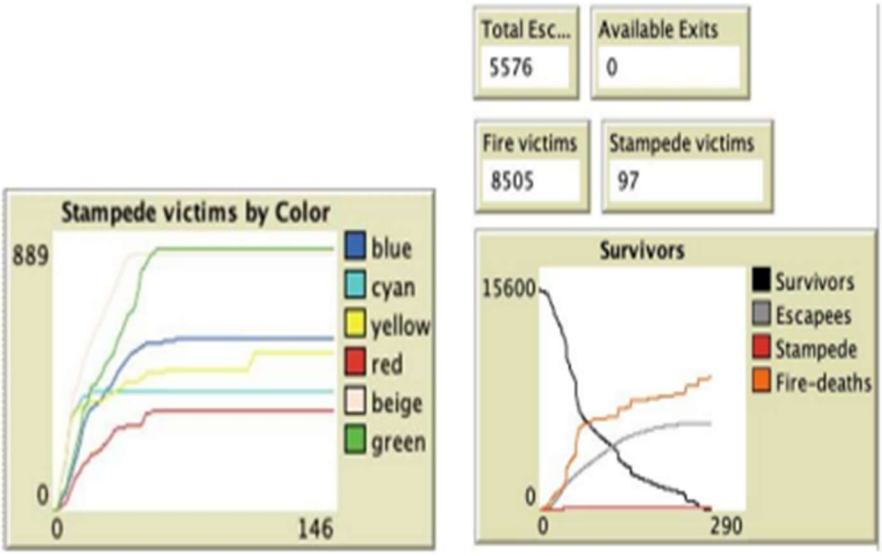

Figure 6.8 follow strategy

Cause 1: Lack of Knowledge as a Factor for Human Stampede Effect

A comparison between the "Smart" strategy and "Follow Strategy" reveals the effect of the lack of knowledge of the stadium layout and exit points on the human stampede effect. We can clearly see the much higher stampede casualty rate in the "follow" strategy.

Cause 2: Seat Location as a Factor for Human Stampede Effect

Given the unsafe layout of the stadium, the seat location is definitely one of the direct influencing factors on the Human Stampede Effect. In the "Smart" strategy, we see that the highest casualties from stampede came from the center block, even though it is not very significant. As for the one with the follow strategy we can see is the seating blocks from the sides that are seeing more stampede casualties.

Cause 3: Age as a Factor for Human Stampede Effect

We see from the results of both the "follow" and "smart" strategy that the children and elderly tend to be greater casualties of the stampede effect, possibly due to their slower movement and susceptibility to getting crushed in high density areas easily.

7. CONCLUSION

Conclusion, this project attempts to find out the significance of various factors on the human stampede effect using the unsafe layout of The Stadium as a simulation environment. Using the experiment results, it hopes to provide some form of insights into likely causes of human stampede effects and seeks to provide informed recommendations to increase survivability in crowd evacuations in such settings. We see that the results have testified to all the hypothesis causes of the human stampede effect, namely the lack of knowledge of the location layout, the seat location of the spectators and the demographics of the spectators. Amongst these factors, we see the largest impact on the human stampede effect being the lack of knowledge of the location layout. This is probably due to the direct effect of the lack of knowledge on creating the 'herding' behaviour which often leads to frantic situations of "blind" leading the "blind". Furthermore, the "following" behaviour causes the pressure in the same direction to increase by multiple times, resulting in much higher stampede casualties. The panic level in this situation is also seen to be much higher. Hence, while changing infrastructure to include more exit points may be the most ideal solution, in the event that this idea is not welcome by authorities, one can simply inform the spectators of the fastest way to exits in times of emergency situations and provides more visual aids such as exit signs along the way. These small alterations may simply lead to a much smaller stampede casualty rate as modelled in the "smart" strategy simulation.

8. REFERENCES

[1] Dennis Parker and John Handmer, Hazard management and emergency planning: perspectives in Britain, Routledge, 2013.

[2] Alexander Mintz, "Non-adaptive group behavior.," The Journal of abnormal and social psychology, vol. 46, no. 2, pp. 150, 2017.

[3] Dirk Helbing, Illes J Farkas, Peter Molnar, and Tamas Vicsek, "Simulation of pedestrian crowds in normal and evacuation situations, Pedestrian and evacuation dynamics, vol. 21, no. 2, pp. 21–58, 2019.

[4] Noor Akma Abu Bakar, Khalid Adam, Mazlina Abdul Majid, and Mario Allegra, "A simulation model for crowd evacuation of fire emergency scenario," in 2017 8th International Conference on Information Technology (ICIT). IEEE, 2017, pp. 361–368.

[5] Yasser M Alginahi, Muhammad N Kabir, and Ali I Mohamed, "Optimization of high-crowd-density facilities based on discrete event simulation," Malaysian Journal of Computer Science, vol. 26, no. 4, pp. 312–329, 2013.

[6] JH Wang and JH Sun, "Principal aspects regarding to the emergency evacuation of large-scale crowds: a brief review of literatures until 2010," Procedia engineering, vol. 71, no. 4, pp. 1–6, 2014.

[7] Uri Wilensky and William Rand, An introduction to agent-based modeling: modeling natural, social, and engineered complex systems with NetLogo, Mit Press, 2015

[7] Daniel Thalmann, Soraia Raupp Musse, and Marcelo Kallmann, "Virtual humans' behaviour: Individuals, groups, and crowds," Tech. Rep., 1999.

[8] Nuria Pelechano, Jan M Allbeck, and Norman I Badler, "Controlling individual agents in high-density crowd simulation," 2007.

[9] Nuria Pelechano, Kevin O'Brien, Barry Silverman, and Norman Badler, "Crowd simulation incorporating agent psychological models, roles and communication," Tech. Rep., PENNSYLVANIA UNIV PHILADELPHIA CENTER FOR HUMAN MODELING AND SIMULATION, 2005.

[10] Linbo Luo, Suiping Zhou, Wentong Cai, Malcolm Yoke Hean Low, Feng Tian, Yongwei Wang, Xian Xiao, and Dan Chen, "Agent-based human behavior modeling for crowd simulation," Computer Animation and Virtual Worlds, vol. 19, no. 3-4, pp. 271–281, 2008.

[11] Adriana Braun, Soraia Raupp Musse, Luiz Paulo Luna de Oliveira, and Bardo EJ Bodmann, “Modeling individual behaviors in crowd simulation,” in Proceedings 11th IEEE International Workshop on Program Comprehension. IEEE, 2003, pp. 143–148.

[16] Yuan Chunmiao, Li Chang, Li Gang, and Zhang Peihong, “Safety evacuation in building engineering design by using buildingexodus,” Systems Engineering Procedia, vol. 5, pp. 87–92, 2012.

[17] Junji Kiyono and Naoto Mori, “Simulation of emergency evacuation behavior during a disaster by use of elliptic distinct elements,” in 13th world conference on earthquake engineering, Paper, 2004, number 134, pp. 1–6.

[18] Saeed Alighadr, Abdolhossein Fallahi, Junji Kiyono, Nabilashuada Rizqi Fitrasha, and Masakatsu Miyajima, “Emergency evacuation during a disaster, study case:“timche muzaffariyye–tabriz bazaar” iran”,” Lisbon: Conference proceeding of, 2012.

[19] Teresa Onorati, Alessio Malizia, Paloma Diaz, and Ignacio Aedo, “Modeling an ontology on accessible evacuation routes for emergencies,” Expert systems with Applications, vol. 41, no. 16, pp. 7124–7134, 2014.

[20] Xiaoxia Yang, Hairong Dong, Qianling Wang, Yao Chen, and Xiaoming Hu, “Guided crowd dynamics via modified social force model,” Physica A: Statistical Mechanics and its Applications, vol. 411, pp. 63–73, 2014.

[21] Ahmed Abdelghany, Khaled Abdelghany, Hani Mahmassani, and Wael Alhalabi, “Modeling framework for optimal evacuation of large-scale crowded pedestrian facilities,” European Journal of Operational Research, vol. 237, no. 3, pp. 1105– 1118, 2014.

[22] Fazilah Haron, Yasser M Alginahi, Muhammad N Kabir, and Ali I Mohamed, “Software evaluation for crowd evacuation-case study: Al-masjid an-nabawi,” International Journal of Computer Science Issues (IJCSI), vol. 9, no. 6, pp. 128, 2012.

[23] Khaled Nassar and Ahmed Bayyoumi, “A simulation study of the effect of mosque design on egress times,” in Proceedings of the 2012 Winter Simulation Conference (WSC). IEEE, 2012, pp. 1–8.