Evacuation from complex floor plans using agent-based modelling

Sami Ali Choudhry, Balsharan Bedi, Eisa Adil, Shiv Sondhi

School of Computer Science
University of Windsor
Windsor, Canada
(bedib, adile, choud116, sondhis) @uwindsor.ca
110010312, 110012666, 105187035, 105216319

***Abstract*—Different building floor plans necessitate specific and tuned evacuation strategies. Therefore, these evacuation scenarios must be simulated beforehand to assess if the scenarios involved are adequately prepared to handle an emergency. These factors include (but are not limited to) the number of exit gates, barriers in the room and number of people being evacuated at a time. Our simulation helps test floor plans through a variety of configurable parameters in which the barriers can be replicated like floor plans (e.g. doors and walls). These simulations can help in better decision-making not only while constructing a building, but also while devising escape strategies during emergencies after construction. In this paper, the workings of this simulation are rigorously explained, and then a handful of scenarios are graphically analyzed, and conclusions are then drawn from them.**

***Index Terms*—Evacuation, Floor plans, Agent-based Modelling, Crowd Control, Simulation**

I. Introduction

In case of an emergency, evacuation of people is the most challenging task. People panic in such situations leading to consequences like trampling, stampedes and emotional trauma. There is a need to simulate the behaviour of people in such emergencies. We have created an agent-based model to understand the behaviour of crowds in an emergency evacuation. This model can be adjusted to represent complex floor plans that can help model better evacuation strategies.

II. Relevant Literature Review

Agent-based models (ABMs) are a kind of micro-scale model that simulate the simultaneous operations of agents in their environment. In this, the goal is to search for analytical insight into the collective behaviour of the agents obeying simple rules. Individual agents act based on some decision-making rules. An ABM consists of numerous agents, decision-making heuristics, learning rules, interactive topology and environment. One such ABM proposed by Collins et al. considers the behaviour of an individual as an important factor in evacuating the building [1]. Evacuating a particular space is closely related to the physical features associated with that space. So, this model focuses more on the topology of the building when evacuation needs to be done. The graph model is used effectively to represent the building that needs to be evacuated. Network analysis is then performed in order to find out the escape doors from where the crowd could take an exit. Such a microscopic model can also give an idea about the route the evacuees can take in case of an emergency. Kasereka et al. proposes a model that can be created and used to evacuate people considering the disorganization in the environment, panic in the people, and evacuating maximum people in the shortest time. Unlike all the other models, the model proposed here used the number of people evacuated as the key parameter for evaluating the result [2]. Trivedi et al. take the psychological factors into consideration during the evacuation in an emergency situation. Such dangerous situations are generally responsible for stampedes as the people are in a state of panic. So, analysis of these factors is done carefully in order to create an agent-based model for evacuation. It also identifies the possible difficulties in the environment that pose a challenge for the people during evacuation. In this, the rule-based roadmap approach is used in which all the important nodes in the surroundings are identified, and each node is associated with the rule [3]. Zhang et al. present an agent-based model that can make their own decisions when people need to be evacuated from a building. The model can simulate the action when evacuation needs to be done, people rush towards the exit, and changing the exit when required. This model performance was good as it was similar to the actions of the real people in case of an emergency [4]. Zhiyong Lin et al. study a multi-agent model that is designed for evacuation during an emergency situation in the city. When such a situation takes place, the simulation is used to find out the exit paths for the people as well as find places where people can take shelter. This agent-based model was made on Repast S, and the result was analyzed using the number of people to be evacuated, knowledge of the people, and the environment [5].

III. Proposed Methodology

To build a successful model for our agent-based simulation, we used a simple, agile approach. Once we had defined our starting objectives and the desired outcomes of the simulation, we began constructing a basic model. During this phase, we also defined the interactions between several components of our model and introduced new parameters to it. We revised and reconstructed our objectives a few times as we learnt more about the simulation environment and how our agent was behaving. This helped us create new obstacles and understand our model better.

Once our model was finalized, we ran a few simulations in different arrangements and collected important data for each simulation. The appropriately represented data helped us draw conclusions from our simulations, which can be extrapolated to real-world scenarios. These findings and suggestions are discussed later. Below, we summarize the methodology we followed.

- Define the objectives.
- Define the goal of the agent and components of the environment.
- Define the interactions between the agent and its environment.
- Introduce changes in the environment - experiment with different parameters such as barrier layouts.
- Run simulations and collect simulation data.
- Plot results and draw conclusions.

In Section IV below, we discuss the creation of our simulation model. Then, in Section V we present our data findings in graphical form and talk briefly about their real-world consequences. Finally, in Section VI and VII we summarize our research and present some final thoughts.

IV. Experimental Process

Our model consists of a number of agents confined in a room that they must evacuate from. We provide exit gates that allow the agents to evacuate and barriers that disrupt their paths. Below, we describe the planning process for setting up our model.

A. *Defining the objectives*

In this study, our main objectives are to simulate the behaviour of people when evacuating from a room during an emergency. We wanted to observe their behaviour as all agents rush to the exits - does anybody get left behind? How long does it take for the agents to evacuate? After working on such questions, new questions arose: is the time taken affected by the number of exit gates? How do the agents' behaviour change when barriers are added in their paths? We continued to make changes to our simulations with these new objectives in mind. Finally, we decided to compare the various simulation scenarios that we had run and see which floor patterns were the most efficient and which weren't.

The aforementioned questions arise from common evacuation patterns. There are often multiple exits and many people evacuating. Some evacuations from real emergencies tend to turn into stampedes and panicked running around. Most rooms also contain walls, tables or other obstacles within them, which may disrupt people's paths. Therefore, to summarise, our final objectives were:

- Identify the effects of panic during evacuation.
- Consider environmental factors during an emergency evacuation.
- Use multiple configurations of environments to study agent behaviour.
- Find shortcomings with various evacuation strategies.
- Compare various evacuation strategies and scenarios.

B. *Simulation set-up*

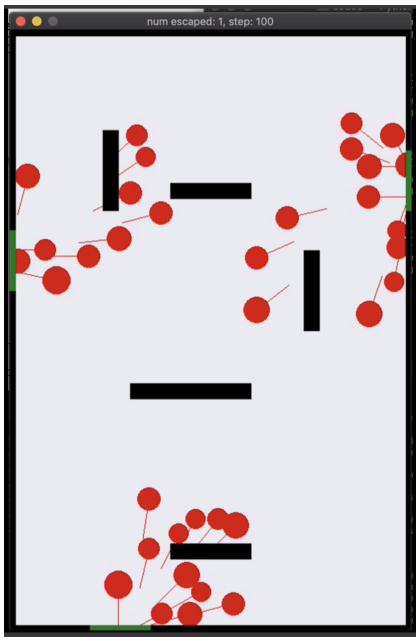

Fig. 1. Set-up for our simulation using PyGame

Our set-up is defined by a number of factors. The programmatic logic was created using Python 3, and the simulation was displayed using the PyGame library (Fig 1). Our codebase contains a switch to enable the simulation to run without the PyGame display, so the code is not dependent on PyGame. However, having the display is more intuitive as it allowed us to observe the agents while they make their decisions.

Each simulation variation has a few properties associated with itself. A few important properties are described below:

- ***Number of agents***: The number of agents is important to observe the panic during an evacuation. A very high number of agents introduce a lag in the PyGame simulation, and a meagre number allows all agents to evacuate very easily. Therefore, we conducted most of our simulations with 40-50 agents. For the sake of comparison, we also added a few simulations with a higher number of agents.
- ***Agent Speed***: The speed of the agents defines how fast they can move across the room. Faster agents would evacuate faster and slower ones would evacuate slower. In the current version of our project, we have kept the speed of all agents equal and set it at around 5 PX/s (pixels/second).
- ***Agent Size***: Each agent is of a random size within certain limits. The agent size plays a minor role in determining who escapes first when the agents are huddled around the exit gate.
- ***Agent Mass***: Each agent has an associated mass, which is directly proportional to its size. The mass has implications when the agent interacts with its surroundings.
- ***Agent Position***: Probably the most important attribute of an agent, at least from a programming point-of-view is

its position in the room. The position not only defines what region is already occupied by this agent, but it also controls the goal of the agent.

We have talked about an agent's goal before and even about multiple exit gates in the room. There is a subtle difference between the two. The gates are part of the agent's environment, which we will talk about soon. Quite separately, each agent has a goal associated with it. In layman terms, the goal of the agent is to successfully escape from the room, but programmatically, the goal of the agent is defined as the exit that is nearest to its current position at any given time. Therefore it is reasonable to have several gates in the model; however, more often than not, an agent has just one goal associated with itself. Another parameter provided in our code is that of the room dimensions. We could build a room of any size for our simulations. Considering the physical screen size and some other aesthetic factors, we decided upon a room having the dimension of (15PX x 10PX).

There are some essential points to note about our set-up. The first point has to do with the dimensions mentioned above, namely PX and PX/s. These dimensions are used to define the key attributes of our system. However, our program logic creates a level of abstraction over the PyGame environment. Therefore, one PX is actually 50 pixels in the PyGame window. This abstraction helps because it allows us to think of the room in terms of a smaller, more manageable grid. Having a room-height of 15 units is easier to work with than a room-height of 750 units (15PX versus 750 pixels). Secondly, it is important to note that in our model, since we have multiple agents, for each induvidual agent, the remaining agents are part of its environment. For instance, if we simulate with 40 agents, the environment of each agent is made up of the room walls, the barriers, the exit gates and 39 other agents.

C. *Components of the simulation environment*

1) ***Agents****:* The agents mimic human behaviour. Each agent wants to achieve a common objective, which in this case, is to exit the room as the project is evacuation from complex floor plans using agent-based modelling. Each agent has an associated vector that defines the agent's desired direction and its internal velocity. During debugging, we draw this vector in the simulation window to watch the agents' directions change. This description does not deal with the single agent, but with the 39 other agents referred to above. Each of these 39 agents obstructs the path of the individual agent. More than obstructing its path, they are all competing with the single agent to get out of the room.

2) ***Barriers and Walls****:* Barriers are simple wall-like structures that disrupt the agent's path - they create obstacles in the path of the agent. Walls are the range of boundary which provides the ground for the agent to move on. The given agent cannot move or pass through a barrier or the walls. However, it can move around the obstacles. We created several different barrier layouts (as shown in Fig. 4) to make it harder for the agents to evacuate. The layout can be passed as an argument while running the program.

3) ***Exit Gates****:* Gates enable agents to escape and exit the room. Exit gates in this particular project are green-coloured cut-outs on the wall, which allow the agent to exit the vicinity. They are represented as coloured sections of the walls through which the agent can pass. Also, the number of gates in the room is parameterized in the simulation beforehand.

4) ***Interactions****:* Interactions are the key to agent-based modelling. This is one of the most important steps when it comes to portraying the results during the simulation. They are usually a specified set of rules which defines how the agent will interact with its surrounding. There can be many types of interactions which in this case are:

- Agent-agent interaction: When an agent approaches another agent and forces an overlap of positions, the agents exert a force on each other, like the real world. This force pushes both agents away from each other. The resultant force from this interaction is affected by the incoming velocities of the agents, their masses and the angle at which they collide.
- Agent-barrier interaction: When an agent approaches a barrier or a wall, both objects exert a force on the other. However, since barriers and walls are immovable, the agent is pushed away off the walls. The resultant force from this interaction is the difference between the normal force (perpendicularly away from the wall) and the tangent force (a diagonal force determined by the angle that the agent's path makes with the wall)
- Agent-gate interaction: When an agent approaches a gate in the wall, it will pass right through it. This event also triggers a counter that keeps track of the number of escaped agents.

Since each interaction (except for agent-gate) generates a resultant force on the agent, the agent's velocity is affected by all these forces. The final velocity of an agent is calculated by summing its internal velocity with all the resultant velocities of the forces acting on it.

V. Results and Discussion

In Fig. 2, we use no barriers and only try scenarios in which we variate the number of exit gates in our simulation. These experimental tests were really helpful in the beginning because we could see agents hypothetically exiting rooms that do not have any internal hindrances. It is evident from our results that an increase in the number of gates leads to an increased number of agents passing within a stipulated time step. In our simulation, the agents knew about the nearest exit. They gravitated towards them by calculating the mathematical Euclidean distance. However, this can not be the case in a real-world situation. This means that while planning evacuation systems, not only should the number of exit gates be maximized based on the floor capacity, but the agents should be thoroughly educated about their presence. This would lead them to propel towards their nearest exit without spending too much time thinking and balance the load on all exit gates.

In Fig. 3 (a), we try a scenario in which the agent's velocity is doubled. For this experiment, we use no barriers and keep

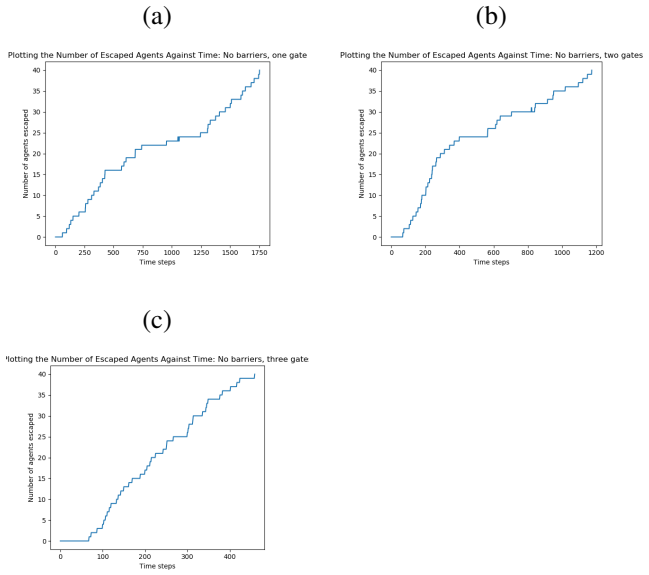

Fig. 2. No barriers with: (a) 1 gate, (b) 2 gates, (c) 3 gates

3 gates, so our result can be compared to Fig. 2 (c). It can be seen from the graphical results that more agents pass through the gates in lesser time if the agents move faster. Drawing a similar inference from the results in Fig. 2, we can say that if agents were better educated about the position of their nearest exit gates, they would move much faster, and this, in turn, would lead to a quicker evacuation. In Fig. 3 (b), we increase the number of agents in the room to 100 from 40. For reference, we can compare it again to Fig. 2 (c). As expected, it takes much more time for agents to pass through, mainly because they collide with each other trying to go out. This finding implies that rooms should have appropriate limits on how many people can occupy the floor area based on the results obtained from the simulation.

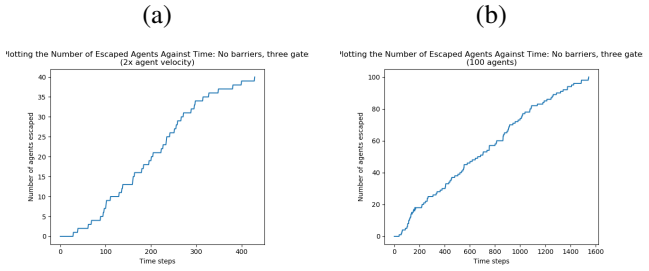

Fig. 3. No barriers & 3 gates with: (a) 2x velocity, (b) 100 agents

We finally add barriers to our simulation based on the three heuristically constructed barrier layouts (BLs), as proposed in Fig. 4. In Fig. 5, we couple these different barrier layouts with a different number of gates to derive results drawn in different complex floor plan settings. While deriving these barrier layouts, we found out that if barriers are located directly in front of the gate, it becomes difficult for the agent to get around the barrier. Therefore, for this experiment, barriers were constructed in such a way that the agents can all pass through. It can be seen in the graphical results that barriers that have a larger footprint and those that occupy more space in front of the gate tend to significantly slow down the agents from passing through.

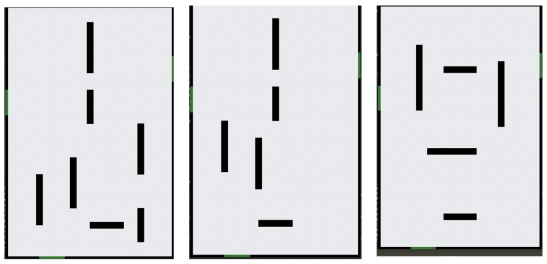

Fig. 4. Barrier layouts 1, 2 and 3 (from left to right)

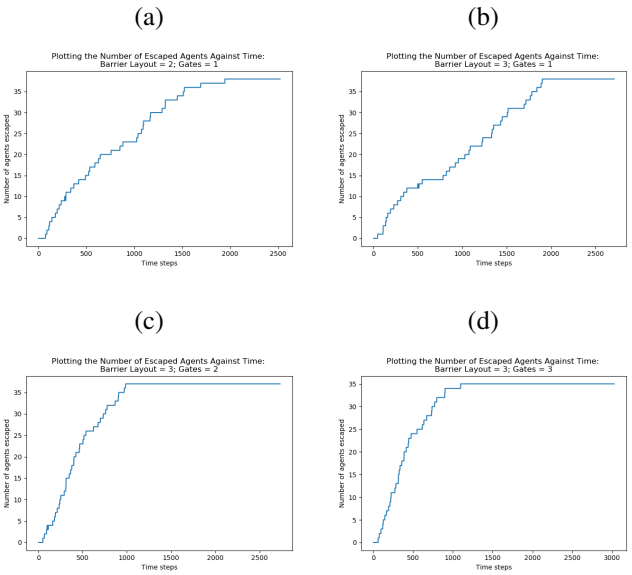

Fig. 5. (a) BL 2 & 1G (b) BL 3 & 1G (c) BL 3 & 2G (d) BL 3 & 3G

VI. Conclusion

We created an evacuation simulation program that is parameterized by the number of agents, internal velocity of agents, number of gates and different barrier layouts. This simulation software can be used to map actual floor plans from buildings to calculate the evacuation time of people. Based on the results, our findings show that before constructing a building, building engineers need to variate and determine the number of emergency exit gates for a proposed floor plan in order to facilitate an efficient and safe evacuation using a simulation (as shown in Fig. 2). Exit gates should be maximized to split the load of all gates, the size of exit gates should be enlarged so many people can get through at a certain time period without colliding, and people should be educated about their nearest exit gates. This education would, in turn, lead to an increase of velocity in people who are evacuating as they would know beforehand where to run to. This is demonstrated

in Fig. 3 (a) when we double the velocity of our agents. For an already constructed building, event organizers need to determine the number of people that can safely gather in order to facilitate an efficient evacuation. In Fig. 3 (b), we showed the impact a drastic increase in the number of agents makes to the simulation. After we tried various barrier layouts in Fig. 5, it was seen that barriers should not obstruct the gate. If barriers are located directly in front of the gate, it becomes difficult for the agent to get around the barrier. Therefore, this simulation can be used to adequately judge the hindrance that barriers can cause at the time of a dire emergency. A possible solution is to teach the agent how to diverge from its current direction to get around barriers.

In short, to drive down evacuation time, floor planners need to decrease the number of agents, decrease barriers and increase the number of exit gates. Further, people should be adequately educated about the emergency gates of places they gather in, and evacuation simulations like this should be used with different scenarios while planning the building.

VII. Future Work

In the future, we can work on making the simulation more robust by making the agents more intelligent so that all agents escape rather than get stuck in barriers due to the lack of intuition of getting around them. We could also give agents different speeds and track their escape time individually, in order to potentially find blind spots in the room from where evacuation is relatively harder. These speeds could be dictated by their mass and physical disabilities too. Further, several more combinations of parameters can be tried in this simulation, such as changing the room size, adjusting the length of the exit gates and making more elaborate and realistic barriers that map actual floor maps.

Acknowledgment

We would like to thank Dr. Ziad Kobti for giving us the opportunity to work on this project and for his insightful discussions on various related topics. In these trying times, this project would not have been possible without the help and constant support of various faculties and organizations at the University of Windsor.

References

[1] Andrew J. Collins et al. “Agent-Based Pedestrian Evacuation Modeling: A One-Size Fits All Approach?” In: *Proceedings of the Symposium on Agent-Directed Simulation*. ADS ’15. Alexandria, Virginia: Society for Computer Simulation International, 2015, pp. 9–17. ISBN: 9781510800984.

[2] Selain Kasereka et al. “Agent-Based Modelling and Simulation for evacuation of people from a building in case of fire”. In: *Procedia Computer Science* 130 (2018). The 9th International Conference on Ambient Systems, Networks and Technologies (ANT 2018) / The 8th International Conference on Sustainable Energy Information Technology (SEIT-2018) / Affiliated Workshops, pp. 10–17. ISSN: 1877-0509. DOI: https://doi.org/10.1016/j.procs.2018.04.006. URL: http://www.sciencedirect.com/science/article/pii/S1877050918303569.

[3] Shrisha Trivedi Ashutosh Rao. “Agent-Based Modeling of Emergency Evacuations Considering Human Panic Behavior”. In: *IEEE Transactions on Computational Social Systems* 5 (Mar. 2018), pp. 277–. DOI: 10.1109/TCSS.2017.2783332.

[4] Y. Zhang et al. “Study on the Agent-Based Evacuation Simulation Models for Large Public Buildings”. In: *2009 Second International Conference on Intelligent Networks and Intelligent Systems*. 2009, pp. 522–525.

[5] Zhiyong Lin et al. “Multi-agent modeling of city emergency evacuation”. In: *2011 International Conference on Multimedia Technology*. 2011, pp. 3570–3574.