Evac Sim: Fall 2020 CSS600

Justin Downes and Chris Smith

December 2020

Abstract

Simulating large scale evacuation is cost-prohibitive in terms of realism and required man-hours. Agent-based simulation supports laying out a floorplan, modeling behaviors and at least capturing some flavor of how a real evacuation might play out. This research uses NetLogo to vary floorplans and capture the average escape times for the agents.

1 Introduction

2 Background

2.1 Previous Work

The literature abounds with previous efforts in this area. [1] [3] [2] [5] [7]

This paper is key as it is extremely similar and a NetLogo implementation. We should know this paper and incorporate into our paper [6]

3 Methodology

The simulation itself is straightforward, and makes novel use of the Behavior Space facility, which we wrapped in a Python driver making use of pytest which is laid out in scripts/tests/test$_{f}ire_{s}im.py$

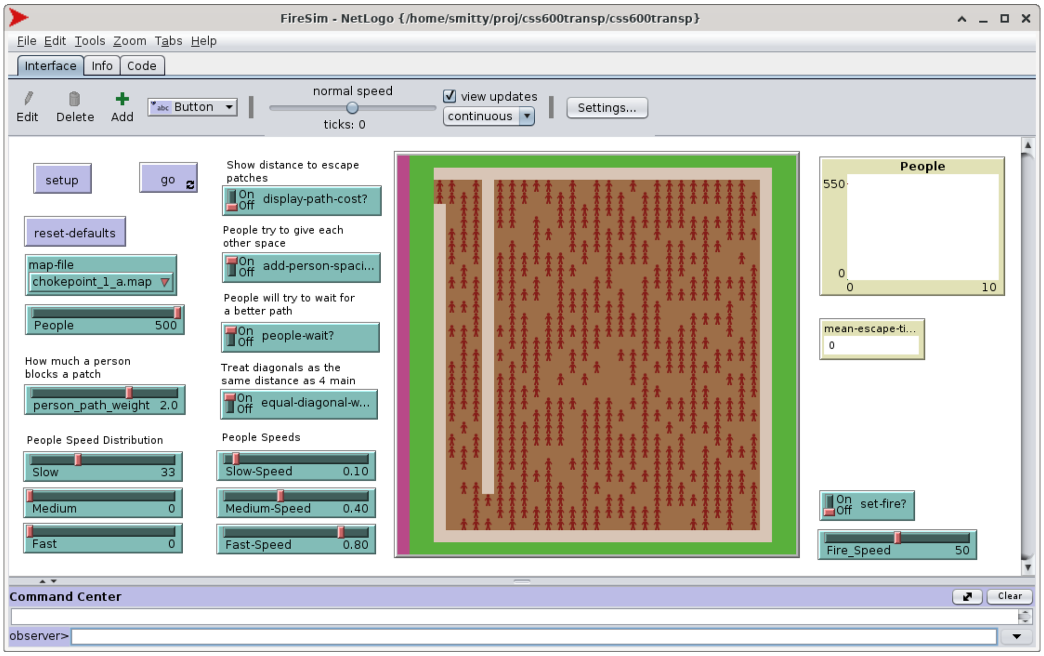

Figure 1: FireSim UI

3.1 Environment

For the map files in use, there were 10 runs against the map files, capturing mean-escape-time for the agents.

map-file	Name of map file to set up.
People	Number of people. Held constant at 500.
person$_{p}ath_{w}eight$	Agent blockage factor for patch
Slow	Percentage moving at this rate, which we set to 100
Medium	Set to 0
Fast	Set to 0
Slow-Speed	0.3 patches
Medium-Speed	0.75 patches
Fast-Speed	1.0 patches
add-person-spacing?	true
equal-diagonal-weight?	true
display-path-cost?	false
people-wait?	true
set-fire?	false
Fire$_{S}peed$	50
mean-escape-time	output

need to talk about patch parameters

3.2 Movement Mechanisms

This section discusses agent and fire movement algorithms.

The cost algorithm, where P_s is a safety patch, where A_w is the person path weight, a weight that a person adds to a patch due to blocking (this is configurable by the user)

$$cost(P) = distance(P_s, P) + Agent(P) * A_w$$
$$Agent(P) = \begin{cases} 1, & \text{Agent Present} \\ 0, & \text{Agent Not Present} \end{cases} \tag{1}$$

This can be configured through equal diagonal weight flag. if False the Distance is the Manhattan distance [1]. If true the distance is the Chebyshev Distance [2]

$$distance(P_1, P_2) =$$
$$equal - diagonal - weight? = \begin{cases} max(|x_1 - x_2|, |y_1 - y_2|), & True \\ |x_1 - x_2| + |y_1 - y_2|, & False \end{cases} \tag{2}$$

[1]https://www.sciencedirect.com/topics/mathematics/manhattan-distance
[2]https://en.wikipedia.org/wiki/Chebyshev_distance

Patch to move to algorithm, where A_p is the patch for a given agent, P_x & P_y are the X & Y coordinates for a given patch

$$neighbors_4(P) = \{patch(P_x - 1, P_y - 1), patch(P_x + 1, P_y - 1), \\ patch(P_x - 1, P_y + 1), patch(P_x + 1, P_y + 1)\} \quad (3)$$

$$move(A) = min(cost(neighbors_4(A_p))) \quad (4)$$

Person will wait for a better patch. This is configurable by the user. If it is on then a person will wait for a patch that is less cost than its current

$$people - wait? = \begin{cases} min(cost(A_p), move(A)), & True \\ move(A), & False \end{cases} \quad (5)$$

Add person spacing algorithm, people try to avoid each other, configurable through the $add-person-spacing$? flag and the $person\_path\_weight$, A_w, parameter. here, A_w, is scaled by a factor of 10 since it is not the weight of being in the same square as another but of being next to another person.

$$add-person-spacing? = \begin{cases} cost(P) \sum Agent(neighbors_4(P)) * A_w/10, & True \\ cost(P), & False \end{cases} \quad (6)$$

[4] [3]

3.3 Experiments

This section describes our experiment harness that we built using NetLogo's Behavior Space functionality.

Behavior Space supports supplying simulation arguments via an XML document, invoking the NetLogo engine via a script pointing to the XML, and then capturing the results via the standard output.

This lends itself to scripting via Python [4]. The goal had been to extract the initial Behavior Space XML content directly from the .nlogo file, and then craft a SQLAlchemy [5] model on the fly that would support storing

[3]http://www.cs.us.es/ fsancho/?e=131
[4]https://www.python.org/
[5]https://www.sqlalchemy.org/

results in an RDBMS, e.g. SQLite [6]. That proved out of reach due to the advanced nature of SQLAlchemy, so a hand-crafted model was used, which makes alterations to the underlying model more maintenance intensive.

While we stored the results in a local SQLAlchemy file, a mere update to the connection string would allow storing results in an enterprise RDBMS to good effect.

Another Open Source tool that was used extensively was PyTest [7]. Billed as a unit testing framework, PyTest supports breaking the problem down into granular fixtures and then combing them in a Lego-like fashion that lends itself to the problem space. For example, while Behavior Space allows stepped alteration of numerical parameters in a model, swapping out map file names is not directly supported. Implementing a Python function to generate the map file names and then re-writing the XML was far more convenient than having distinct experiments for each map.

And then Python's data science facilities are well-known [8], supporting arbitrary visualization pipelines.

3.3.1 Experiments Based on Layouts

The length of the exit passage was increased gradually

3.3.2 Experiments Replicating Emergent Crowd Behaviors

- mainly from this paper [1] .

if we can show that we achieve similar results even though we use a simplified pathing algorithm and abm environment i think that would be insightful

4 Results

The results showed that adding more length to the chokepoint pipeline did not result in a linear increase in the exit time. Rather, the Actors resembled a fluid dynamics problem, reaching a uniform flow and making good their escape.

[6] https://sqlite.org/index.html

[7] https://docs.pytest.org/en/stable/

[8] https://www.scipy.org/index.html

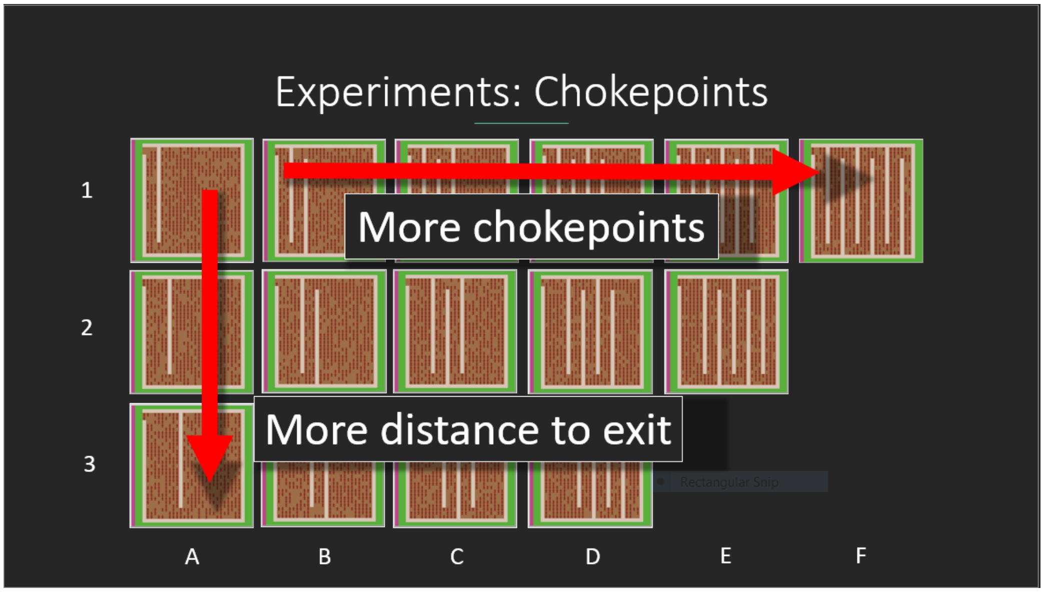

Figure 2: Chokepoint Overview

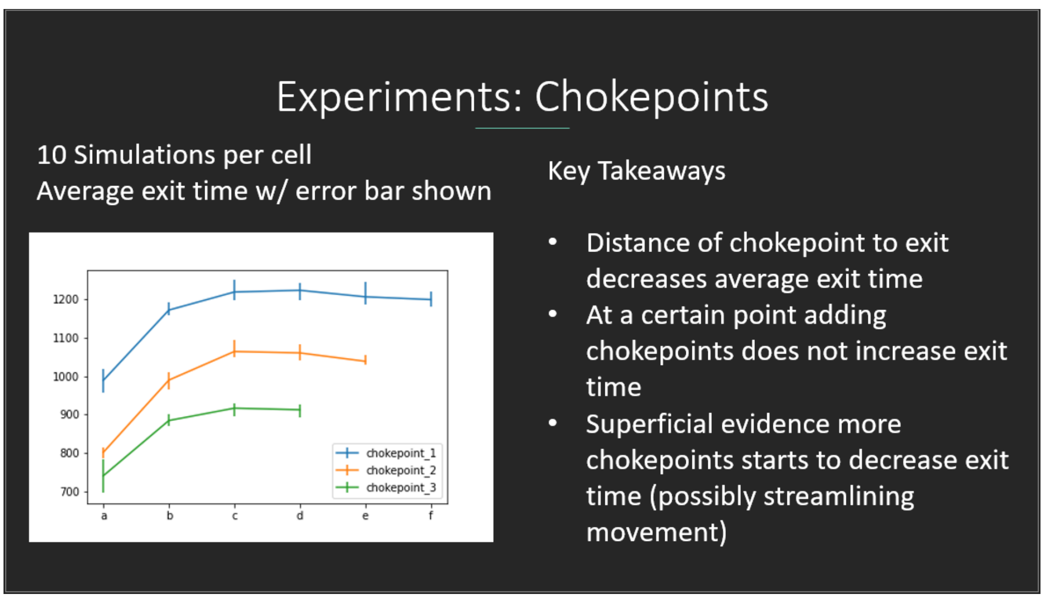

Figure 3: Chokepoint Results

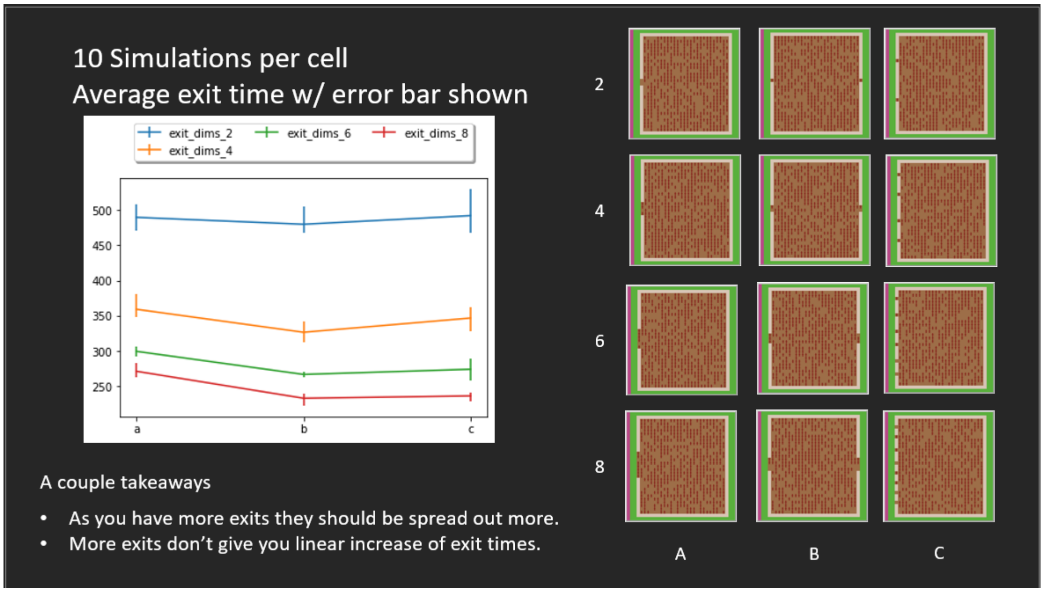

Figure 4: Exit Dimensions Overview/Results

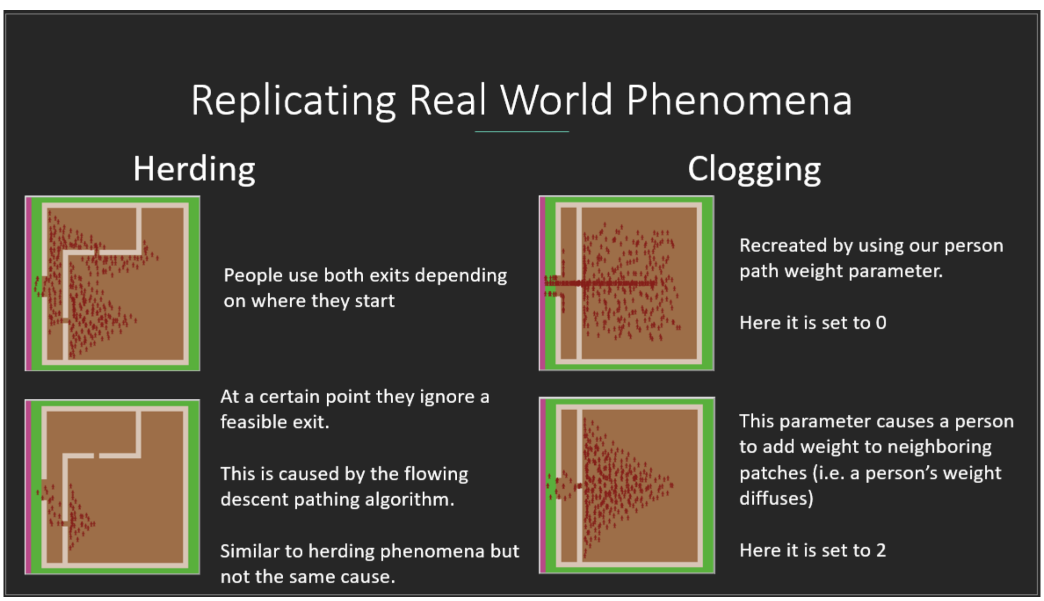

Figure 5:

More exits is better, especially if not on the same side, but there is not a linear increase with additional exits.

need to talk about patch parameters

4.1 Movement Mechanisms

This section discusses agent and fire movement algorithms.

The cost algorithm, where P_s is a safety patch, where A_w is the person path weight, a weight that a person adds to a patch due to blocking (this is configurable by the user)

$$cost(P) = distance(P_s, P) + Agent(P) * A_w$$
$$Agent(P) = \begin{cases} 1, & \text{Agent Present} \\ 0, & \text{Agent Not Present} \end{cases} \tag{7}$$

This can be configured through equal diagonal weight flag. if False the Distance is the Manhattan distance [9]. If true the distance is the Chebyshev Distance [10]

$$distance(P_1, P_2) =$$
$$equal - diagonal - weight? = \begin{cases} max(|x_1 - x_2|, |y_1 - y_2|), & True \\ |x_1 - x_2| + |y_1 - y_2|, & False \end{cases} \tag{8}$$

Patch to move to algorithm, where A_p is the patch for a given agent, P_x & P_y are the X & Y coordinates for a given patch

$$\begin{aligned} neighbors_4(P) = \{&patch(P_x - 1, P_y - 1), patch(P_x + 1, P_y - 1), \\ &patch(P_x - 1, P_y + 1), patch(P_x + 1, P_y + 1)\} \end{aligned} \tag{9}$$

$$move(A) = min(cost(neighbors_4(A_p))) \tag{10}$$

Person will wait for a better patch. This is configurable by the user. If it is on then a person will wait for a patch that is less cost than its current

$$people - wait? = \begin{cases} min(cost(A_p), move(A)), & True \\ move(A), & False \end{cases} \tag{11}$$

[9]https://www.sciencedirect.com/topics/mathematics/manhattan-distance

[10]https://en.wikipedia.org/wiki/Chebyshev_distance

Add person spacing algorithm, people try to avoid each other, configurable through the $add{-}person{-}spacing?$ flag and the $person\_path\_weight$, A_w, parameter. here, A_w, is scaled by a factor of 10 since it is not the weight of being in the same square as another but of being next to another person.

$$add{-}person{-}spacing? = \begin{cases} cost(P) \sum Agent(neighbors_4(P)) * A_w/10, & True \\ cost(P), & False \end{cases} \tag{12}$$

[4] [11]

4.2 Experiments

This section describes our experiment harness that we built using NetLogo's Behavior Space functionality.

Behavior Space supports supplying simulation arguments via an XML document, invoking the NetLogo engine via a script pointing to the XML, and then capturing the results via the standard output.

This lends itself to scripting via Python [12]. The goal had been to extract the initial Behavior Space XML content directly from the .nlogo file, and then craft a SQLAlchemy [13] model on the fly that would support storing results in an RDBMS, e.g. SQLite [14]. That proved out of reach due to the advanced nature of SQLAlchemy, so a hand-crafted model was used, which makes alterations to the underlying model more maintenance intensive.

While we stored the results in a local SQLAlchemy file, a mere update to the connection string would allow storing results in an enterprise RDBMS to good effect.

Another Open Source tool that was used extensively was PyTest [15]. Billed as a unit testing framework, PyTest supports breaking the problem down into granular fixtures and then combing them in a Lego-like fashion that lends itself to the problem space. For example, while Behavior Space allows stepped alteration of numerical parameters in a model, swapping out map file names is not directly supported. Implementing a Python function to generate the

[11] http://www.cs.us.es/ fsancho/?e=131
[12] https://www.python.org/
[13] https://www.sqlalchemy.org/
[14] https://sqlite.org/index.html
[15] https://docs.pytest.org/en/stable/

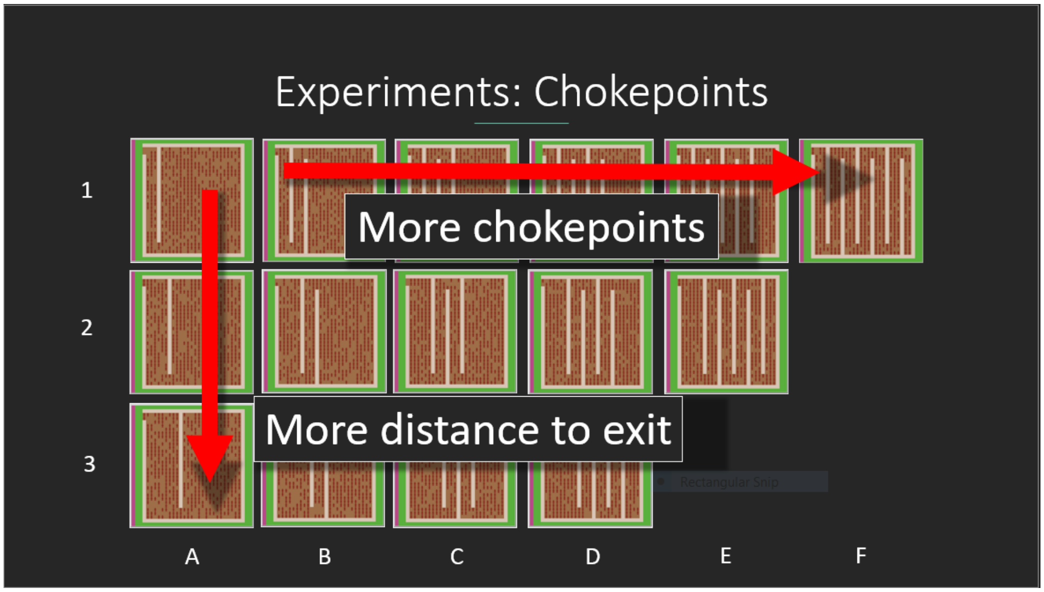

Figure 6: Chokepoint Overview

map file names and then re-writing the XML was far more convenient than having distinct experiments for each map.

And then Python's data science facilities are well-known [16], supporting arbitrary visualization pipelines.

4.2.1 Experiments Based on Layouts

The length of the exit passage was increased gradually

4.2.2 Experiments Replicating Emergent Crowd Behaviors

- mainly from this paper [1] .

if we can show that we achieve similar results even though we use a simplified pathing algorithm and abm environment i think that would be insightful

[16] https://www.scipy.org/index.html

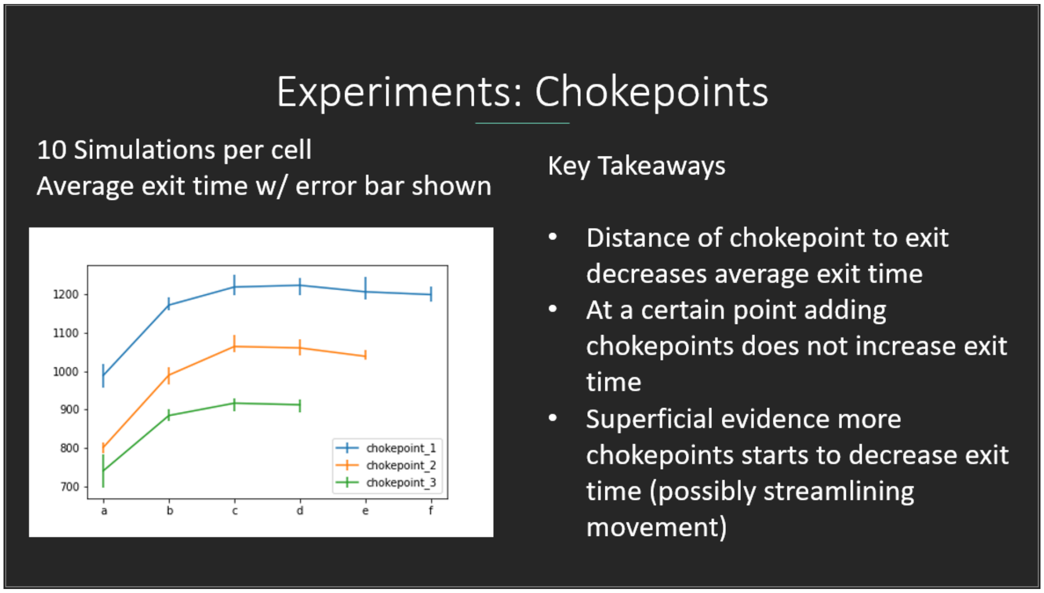

Figure 7: Chokepoint Results

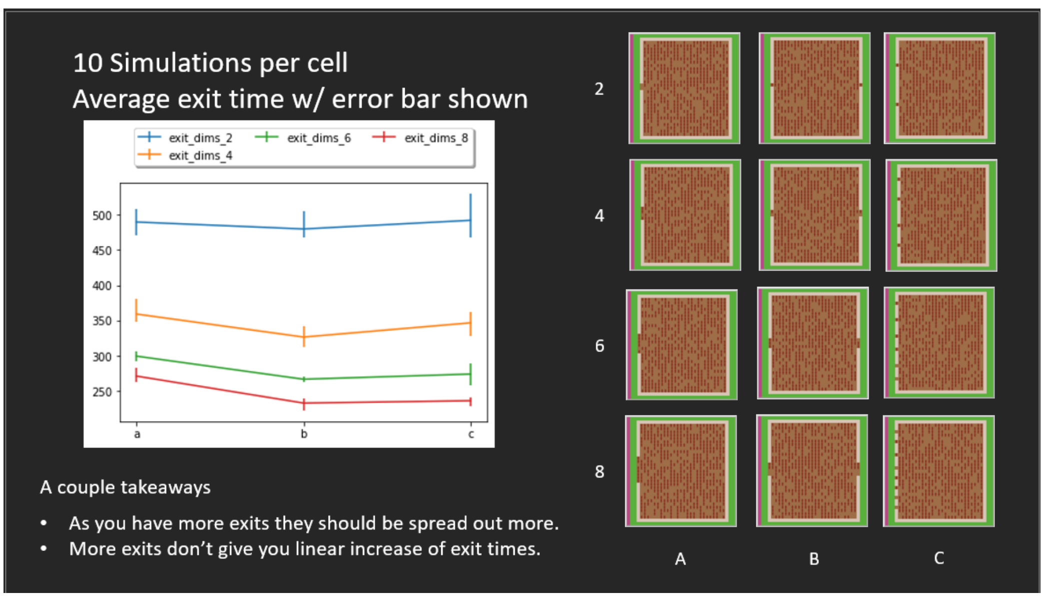

Figure 8: Exit Dimensions Overview/Results

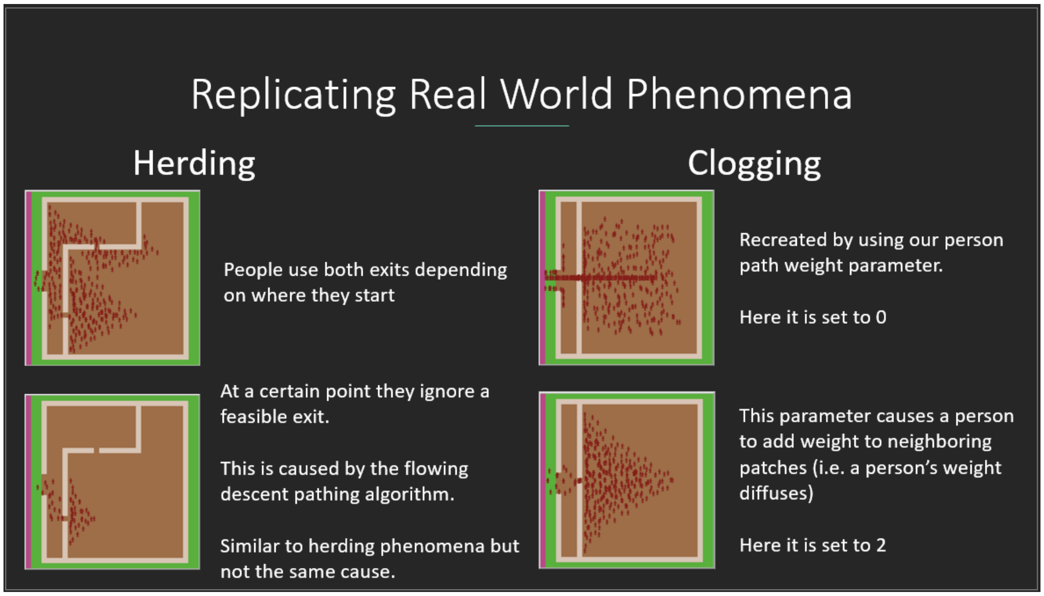

Figure 9:

5 Results

The results showed that adding more length to the chokepoint pipeline did not result in a linear increase in the exit time. Rather, the Actors resembled a fluid dynamics problem, reaching a uniform flow and making good their escape.

More exits is better, especially if not on the same side, but there is not a linear increase with additional exits.

chokepoint1	a	966.4	1006.0	963.0	995.6
	b	1155.6	1190.0	1149.0	1181.6
	c	1207.0	1243.0	1198.0	1234.9
	d	1217.5	1234.0	1210.0	1231.1
	e	1202.4	1243.0	1200.0	1226.0
	f	1183.1	1212.0	1179.0	1206.3
chokepoint2	a	790.8	833.0	781.0	819.8
	b	975.7	1025.0	970.0	1015.7
	c	1036.3	1071.0	1017.0	1071.5
	d	1039.3	1071.0	1034.0	1065.5
	e	1034.8	1055.0	1024.0	1051.2

chokepoint3	a	717.6	758.0	704.0	751.9
	b	886.9	919.0	880.0	911.2
	c	905.2	952.0	900.0	937.3
	d	902.3	925.0	894.0	919.2
exitdims2	a	480.6	516.0	472.0	502.5
	b	482.9	522.0	474.0	510.0
	c	481.3	524.0	473.0	510.6
exitdims4	a	348.4	368.0	342.0	367.3
	b	319.2	344.0	317.0	336.3
	c	334.3	365.0	328.0	358.8
exitdims6	a	292.0	302.0	287.0	301.7
	b	262.0	285.0	258.0	277.9
	c	265.8	290.0	265.0	280.3
exitdims8	a	264.7	279.0	263.0	276.2
	b	230.4	243.0	228.0	240.3
	c	231.4	250.0	231.0	242.7

6 Conclusion

The simpler the layout, with more and diversified exit options, the more optimal the evacuation results.

References

[1] João E. Almeida, Rosaldo J. F. Rosseti, and António Leça Coelho. Crowd Simulation Modeling Applied to Emergency and Evacuation Simulations using Multi-Agent Systems.

[2] E.D.Kuligowski and S.M.V.Gwynne. The need for behavioral theory in evacuation modeling.

[3] Angelika Kniedl, Dirk Hartmann, and Andre Borrmann. Using a multi-scale model for simulating pedestrian behavior.

[4] Maysam Mirahmadi and Abdallah Shami. A Novel Algorithm for Real-time Procedural Generation of Building Floor Plans.

[5] Fangqin Tang and Aizhu Ren. Agent-based evacuation model incorporating fire scene and building geometry.

[6] Eileen Young. Prioritevac: An agent-based model of evacuation from building fires.

[7] Jibiao Zhou, Yanyong Guo, Sheng Dong, Minjie Zhang, and Tianqi Mao. Simulation of pedestrian evacuation route choice using social force model in large-scale public space: Comparison of five evacuation strategies. 14(9):e0221872–e0221872.