Pereira Miguel

Emotional Contagion Influenced By Authority In The Context Of Panic Crowd Simulation

Graduation Work 2022-2023

Digital Arts and Entertainment

Howest.be

Contents

ABSTRACT

Simulating crowd-based reactions to panic-inducing hazards has been a popular research topic for the past decades, as it has proven to have several applications both in the context of strategic planning in public safety and in the context of the entertainment industry **[1]**. Multiple simulation models have been developed where the individuals within a crowd behave in personalized emotional manners, reacting to outside conditions according to their personal psychological traits, but this paper proposes an expansion on said models. Specifically, the proposed approach takes as basis two recent simulation models, Dynamic Behavior **[2]** and the Third-Party **[3]** models. The proposed approach expands on the concept of contagious emotions which trigger from both hazards and other agents (from the first model), and uses authority as one of several reacting conditions (from the second). This approach therefore applies an emotion-based profiling method reminiscent of the OCEAN personality model **[4]**, a memory system for individual spatial awareness and a CA-SIRS **[5]** inspired logic for emotional contagion methodology, along with a mix of BOID-like **[6]** behaviors and Flow Fields **[7]** for agent movement steering. This proposed technique can achieve a rich yet simple and processing-light simulation, which can prove itself useful both in the context of safety-planning, and in the context of video-game coding.

INTRODUCTION

The simulation of crowds within the context of panic situations has been a well-documented topic for the past decades, as their usage has proven to be beneficial in strategic planning for emergency prevention regarding public safety. Multiple models and techniques have been developed, attempting to achieve realistic and natural behavior replication with the technological resources available.

These models have taken different approaches to calculate the movement and intention of individuals, but most share the common ground of taking an agent-based approach (also known as ABM), where the units of a crowd are considered as independent individuals, capable of making autonomous decisions. Some of the most relevant ABM models include the BOID **[6]**, which pioneered this approach, CA-SIRS (Cellular Automata Standardized Infection Ratio) **[5]**, HiDAC (High-Density Autonomous Crowds) **[8]**, Context Steering **[9]**, Third-Party **[3]** and Diversity Behavior **[2]**. All these models are further expanded on in the "Related Work" section.

In the last 15 to 20 years, several ABMs have started implementing psychological profiling methods as to aid in the replication of more personalized human behavior, such as the the Big-Five Personality Traits model **[4]** (also known as OCEAN) and the Belief-Desire-Intention model (also known as BDI). The adoption of these profiles has allowed simulation models to achieve more varied and faithful behavior to a microscopic and macroscopic level, as it has allowed programmers to replicate how different people may (and normally do) react differently to similar stressful conditions and panic-inducing hazards.

Two recently developed simulation models that focused on agent psychological profiling as to achieve emotional contagion were Third-Party and Diversity Behavior. Both models take a cellular automata approach to crowd simulation, based on the OCEAN method, but differ wildly in their approach to agent intention calculation. The first introduces figures of authority which are immune to outside influences, but affect the remaining crowd heavily, which is in itself divided into following/leading roles. The second expands on the relationship between agent's emotional-profiles and their reactiveness to not only hazards, but also surrounding agents – assimilating a single emotion at a time, which dictates the agent's steering until an internal timer runs out.

Inspired in these two models, I came up with a pair of questions to guide my research process:

- Can a panic simulation model be developed, inspired in Third-Party approach, where leadership is an adaptative parameter, influenced by other traits and factors, as well as authority?
- Is it possible to expand on the Diversity Behavior model as to allow for adaptive emotions, reactive to emotional triggers?

Taking both these questions into consideration, the model proposed by this paper was developed – a cellular automata inspired approach to panic crowd simulation, in which agents' behaviors are calculated through adaptive emotions, reacting not only to hazards and other agents, but also figures of authority. Each agent in this model is initialized with an OCEAN profile, which defines: how likely they are to get influenced by others' emotions; how likely they are to react to a hazard by panicking or evacuating orderly; how effectively they can learn and remember their surroundings; how curious they are to explore different rooms while calm; and how fast they are to listen to authority figures. Whatever emotion an agent assumes defines what type of actions they are able to execute (for example, while calm they can either wander around their current room or look for a new room). Unlike the before mentioned Diversity Behavior model, this proposed approach allows for emotions to switch automatically, given that enough outside influence has been provided to the agent.

RELATED WORK

As mentioned in the introduction, there have been several papers written around crowd simulation, from as early as 1987 to today. Although the goal of this paper focuses on emotion-based models applied in panic situations, multiple models that don't fit that description will be mentioned in the following pages, including non-programming models, as they all serve a contextual purpose for the proposed approach.

1. BOID

The BOID model was proposed by Craig Reynolds in 1987 **[6]** and it was one of the first agent-based simulations to be developed. His work has since then become a cornerstone of crowd simulation, being quoted in several papers on the topic, and being applied in several disparate contexts, such as in the 1998 game "Half-Life".

This model was proposed with the goal of avoiding individual hand animation of models – both for optimization's sake, and to reduce the workload of animators (as animating organic-looking flocks manually is very time-consuming, as well as prone to unintentional collisions). It has, since then, been expanded on substantially for other non-animation related purposes – like this paper's topic, panic simulation. As such, its application was mostly envisioned with flocks of birds and other formation-prone agents in mind, contextualizing its name as a shortened version of "bird-oid object".

The BOID model treats agents in a similar fashion to particles, leading each agent to calculate their own desired steering direction every frame, according to its pre-defined target, while still adjusting to the position of the remaining agents. To achieve this, agents can be assigned a blend of a myriad of individual steering behaviors (like wandering and fleeing/seeking a target), but these will still be taken into consideration along with three other factors: Separation, Alignment and Cohesion. Each of these three factors lead all the agents within a flock to behave as single unit while still considering their individual steering desires, therefore simulating some resemblance of individuality along with uniformity. As illustrated bellow, separation corresponds to the avoidance of nearby agents; alignment corresponds to the rotation towards the average facing direction of nearby agents; and cohesion corresponds to the seeking of the average position of nearby agents.

In the model proposed by this paper, although most of the orderly movement applies a FlowField-based logic (described in the following pages), a behavior reminiscent of BOID is still applied. Specifically, depending on the action, multiple separation and cohesive forces are applied to the agents, to both avoid hitting walls, and to avoid colliding with each other while still following a similar goal. Regardless, even if it hadn't directly influenced the proposed approach, BOID's relevancy in the field of agent-based crowd simulation makes it a crucial piece of literature to understand the concepts explored further in this paper.

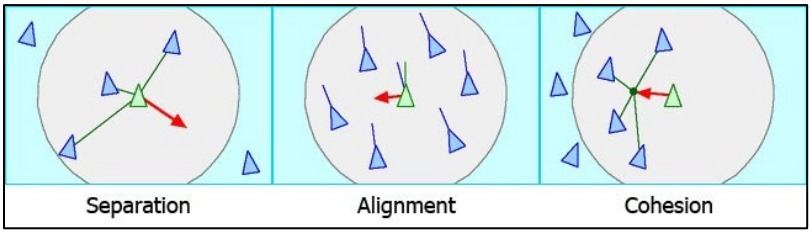

Fig. 01 – Main BOID steering behaviors

2. CA-SIR

As previously mentioned, a cellular automata (or CA) approach to crowd simulation has been quite popular for the past decades. CA's ease of implementation and environmental-partitioning logic make it an optimal technique for non-macro scale simulations.

The origins of Cellular Automata stem all the way back to the 1940s, as it was first conceptualized by Stanislaw Ulam and John von Neumann, and later expanded during the 1950s and 1960s **[10]**. Although with no practical application yet, the goal of the original cellular automata theorical model was to achieve a "self-reproducing machine" **[11]**, simulating the biological processes of evolution witnessed in nature. This "machine" would be composed of elements named cells, each doted with a single value and a given number connections to other cells. The values of said cells would change at given intervals according to the influence of their neighbors, following a set of predefined rules. The original theorical model proposed by Neumann was limited to twenty-nine possible states/values and four possible neighboring connections, one in each cardinal direction.

This model would later be expanded on, within the context of the two-dimensional application most recognize as the iconic CA propagation example – the "Game of Life". This was a self-playing game developed by the British mathematician John Horton Conway, in 1970 **[12]** (illustrated bellow). The game depicted an infinite space partitioned into cells by a squared-grid, each allowing for only 2 states – alive or dead – and 8 neighboring connections, horizontally, vertically and diagonally. The defined propagation ruleset consisted of 2 simple rules:

- Cells would die, unless 2 or 3 of its neighbors were alive (no more, no less);
- Dead cells would remain dead indefinitely, unless exactly 3 of their neighbors happened to be alive, in which case they would revive.

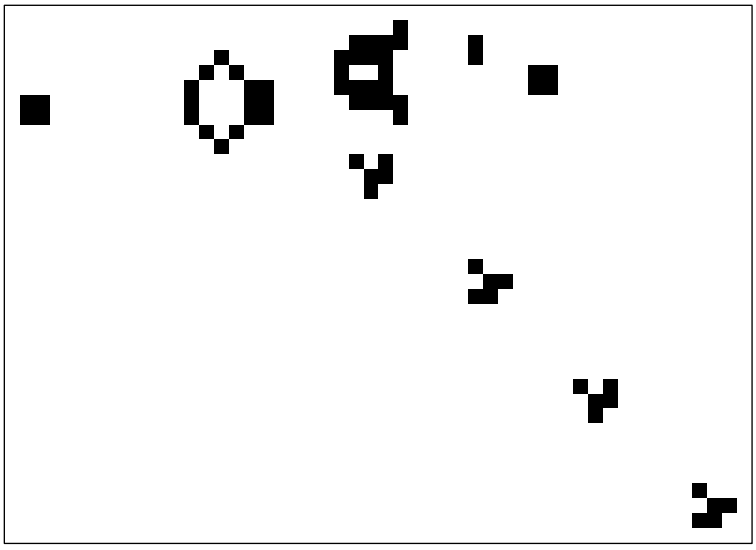

Fig. 02 – Screenshot from "Game of Life"

SIR, or Standardized Infection Ratio, is a specific contagion equation first documented in 1927 by Kermack and McKendrick **[5]**.

The equation itself consists of "N = S + I + R", where N stands for the total population, S stands for the susceptible crowd, I for the infected crowd, and R for the recovered crowd (and therefore also immune). The illustration bellow (taken from **[13]**) showcases the possible transitions between these states, where A represents the infection possibility between a susceptible and an infected agent represents the recovery rate, or the average infection duration, and C represents the rate in which agent can lose immunity after recovering from an infection.

SIR can be expanded to account for many other factors, such as mortality and birth rate and their correlations with infection, but for the context of the simulation models discussed in this paper, susceptibility, infection and recovery are the only relevant key-states.

Using SIR's infection logic along with the cellular automata approach described before, one can achieve a seemingly realistic infection simulation within a static environment – which, in turn, can be expanded on for the context of emotion propagation. Due to this reason, CA-SIR approaches are common-place in emotional contagion simulations, including both fundamental reference papers this research is based on (mentioned in the introduction). The model proposed in this paper also applies a CA-SIR inspired method for calculating the contagion and emotion effect time-spawn, as further described in the Study Case section.

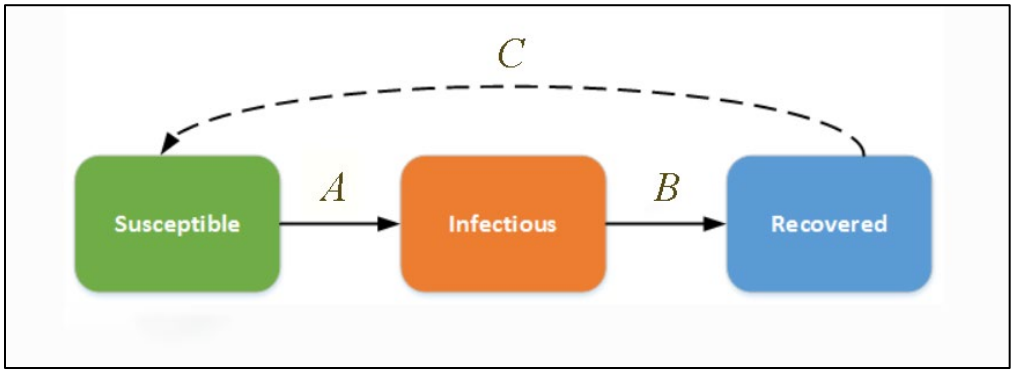

Fig. 03 – The 3 stages of contagion, as proposed by the SIR model

3. OCEAN

The OCEAN model, also known as the Big 5 Personality Traits, is a psychological profiling method proposed and defined by several independent researchers, but later expanded and further organized into the form we know today by Lewis Goldberg in 1990 **[4]**.

This method proposes the division and categorization of human's emotional expression into five traits – Openness, Conscientiousness, Extraversion, Agreeableness and Neuroticism. The picture bellow, taken from an article by the company MindHelp, explains the difference between these traits with clear examples.

Although this model was not developed with crowd simulation in mind, it still stands today as the foundation of many emotion-based simulation approaches, including the one proposed by this paper. The relevancy and usefulness of this model in the field of simulation, although reductive to today's standards, stems from it serving as a credible way to simplify emotional responses in panic situations into easily quantifiable variables. For the proposed implementation, every agent is initialized with a struct based on this model, with individual values for each trait – which then affect how the agent acts, perceives their environment, and affects / gets affected by others.

Fig. 04 – The 5 human traits, as proposed by OCEAN model

4. BDI

Another psychological model often applied to emotion-based simulation is BDI, also known as the Belief Desire Intention model. This method was proposed by the American philosopher and professor Michael E. Bratman in 1987 **[14]**. A BDI architecture, as its name entails, is defined by the application of individual belief-sets, desires, goals, intentions, plans and events to each agent, as illustrated bellow, to the right.

Beliefs stand for the information the agent perceives from their environment, which can be mutable and not correspond to the actual reality of the environment as well. Desires correspond to the motivations of said agent, and the final objectives they intend on accomplishing. Goals are the desires chosen for active pursuit, as desires might be non-interchangeable or even contradictory. Intentions stand for the desires the agent is currently committed to achieve, therefore a goal already mid-execution. Plans are the arrays of direct actions the agent engages in to achieve their intentions. Lastly, events are reactions to the agent's actions, which might both mutate the environment and the beliefs/goals of the agent themselves.

The agent architecture used in the model proposed by this paper follows a similar logic to that of BDI, separating the accumulated emotional influence agents perceive (similarly to Beliefs), their assumed predominant emotion (somewhat like Desires), and the action they take (just like Intentions).

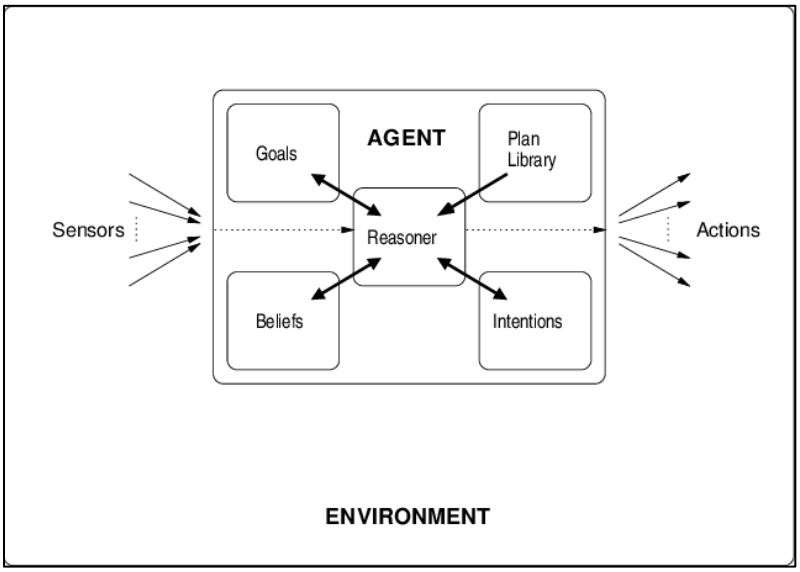

Fig. 05 – BDI's agent's decision-making architecture

5. Third Party

The Third-Party model was proposed in 2018 by Yan Mao, Shanwen Yang, Zuning Li and Yongjian Li **[3]**. Its development stemmed from the lack of literature the authors found when it came to authority presence in panic simulations – as simulations such as the one in question often replicate situations which, in real life, would most likely be affected by trained personnel such as fire-fighters or police.

To simulate this effect, the model they propose initializes each agent with an OCEAN emotional profile, and it further assigns them with one of four static roles: third-party authority agents, group leaders, members and isolated agents.

Leaders and members share an intra-group emotional relationship, as the leaders' actions directly influence the actions of the remaining members, regardless of their emotional profiles. Groups can influence each other as well, along with isolated agents, allowing members to switch groups and isolated agents to be assimilated. Lastly, third-party agents serve as figures of authority, which cannot be influenced nor integrated, but can affect members, leaders and isolated agents alike. Third-party agents do not possess an emotional profile, as they mostly serve as a guiding trigger. These intra/inter-group dynamics are exemplified by the first image bellow, taken from the paper at hand.

The way in which agents calculate their individual desired steering is guided by a personal goal, which is concluded through their perception of the environment according to their personality traits. This goal then mutates according to the influence of both their group, surrounding groups they might encounter and the intention of the nearby authority figures. This architecture is illustrated in the second image bellow, also taken from the original paper.

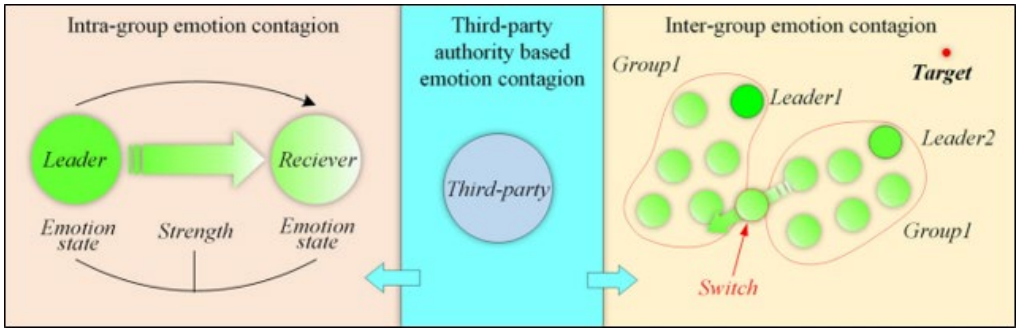

Fig. 06 – Contagion dynamics within Third-Party model

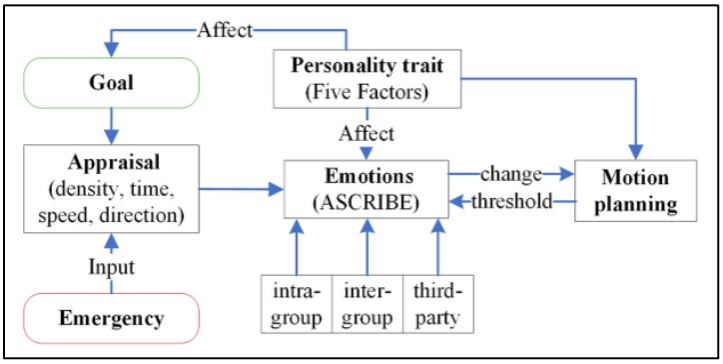

Fig. 07 – Agent decision-making logic in Third-Party model

6. Diversity Behavior

The Diversity Behavior model was also proposed in 2018 by Yan Mao, Zuning Li, Yongjian Li and Wu He **[2]**. Its goal was to develop simulated behaviors that would better account for human unpredictability and reaction variety. As the paper best puts it, “In most of the existing crowd escape simulation systems, human behaviors are often limited to taking flight or running away. However, due to the personality and the current emotion of each individual, they may behave in various ways”. This model therefore focuses on handling reactions such as surrounding hazards due to curiosity, as well as freezing with anxiety-induced indecision.

To achieve this, the Diversity Behavior approach applies not only an OCEAN profile to each agent, but an OCC one as well. The OCC model is a psychological taxonomy proposed by Andrew Ortony, Gerald Clore and Alan Collins in 1988 **[15]** which, similarly to OCEAN, proposes 22 emotional categories for the representation of the human psyche. This model also proposes a set of variables which are directly linked to emotion intensity – affecting both individual emotions, and the global emotional availability of the individual.

With these combined profiles, it then applies the CA-SIRS emotion contagion model (described previously), as well as the effects of Yerkes–Dodson Law **[16]** on each agent's behaviors, as to simulate authentic individuality in movement. This mentioned law dictates that although performance quality inherently increases with mental arousal, this positive relationship has a limit - as performance actually starts decreasing once a certain arousal cap has been met (as shown in the illustration to the right, taken from the original paper).

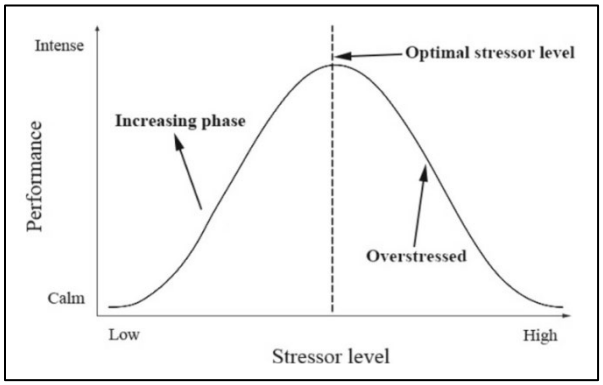

Fig. 08 - Yerkes-Dodson Law's emotional performance over stressor level

Ultimately, this model’s decision-making architecture is not too different from that of the previous model (Third-Party), but although it is much less focused on group dynamics, it offers a much richer myriad of desires and goals handling. This architecture is showcased by the illustration bellow, also taken from the original paper.

Although this model can provide diverse individual behaviors, it does have one downfall (which the model proposed by this paper solves), which is, as the paper itself specifies, "(in this model) once a certain emotion is activated, the agent will no longer be affected by other emotions until its recovered".

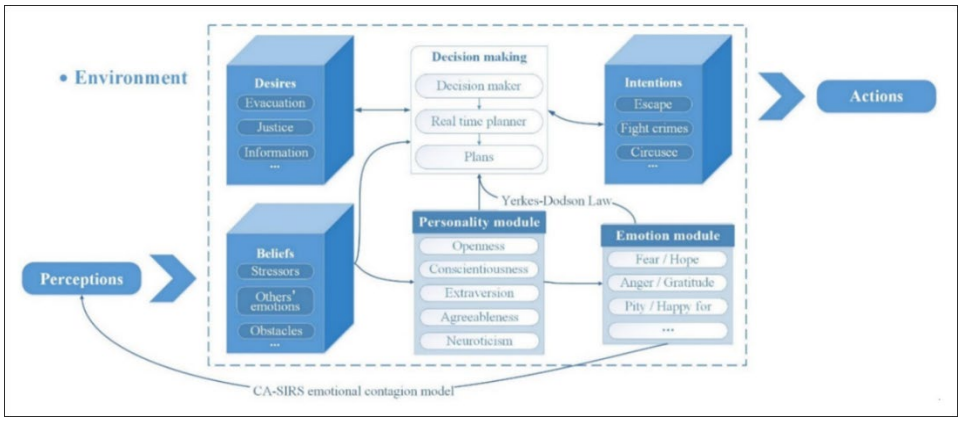

Fig. 09 – Agent's decision-making logic in Diversity Behavior model

7. Flow Fields

Flow Fields are one of the most well-known approaches to pathfinding, due to their simple nature and how optimized their implementations can be when applied to large crowds with common targets **[7]**. For this reason, Flow Fields are often thought about within the context of their common usage in the RTS videogame genre. Nonetheless, this technique has been applied in many other fields, including procedural texture-art **[17]**.

A Flow Field implementation would consist of dividing the intended environment into a grid (usually bidimensional), where each intersection-square is labeled a "cell". Each cell keeps track of what other cells are they connected to – aka, their neighbors. One can consider a cell's neighbors only its horizontal and vertical connections, or the diagonals connections as well. Cells must possess an ingrained direction as well, towards the shortest path to the target position. All agents within said cell should be able to access this direction – as to avoid redundant agent-specific steering calculations. Cells that collide with non-traversable obstacles, such as walls, should be set as invalid.

The way each cell calculates its direction is through a "cost-based" logic. The more layers of neighbors a cell must go through to get the cell where the target position lies (ignoring invalid cells), the higher the cost. Once all the costs are calculated, the direction should point from the center of the cell to the cheapest neighbor's center.

The two main downfalls of this pathfinding method are: the fact it relies on the presupposition all agents share a common goal (which is a huge limitation), and the fact it generally creates very predictable and unnatural steering behaviors – as the crowd will always rotate in 45º offsets, considering the minimum direction change an agent can achieve is when moving towards a diagonal neighboring cell.

Regardless of these few limitations, Flow Fields were a key component to development of this paper's Case Study. A big portion of all agents' steering is partially influence by a group of 6 static Flow Fields, as better explained in the "Case Study / 3. Steering Agents' Movement" section.

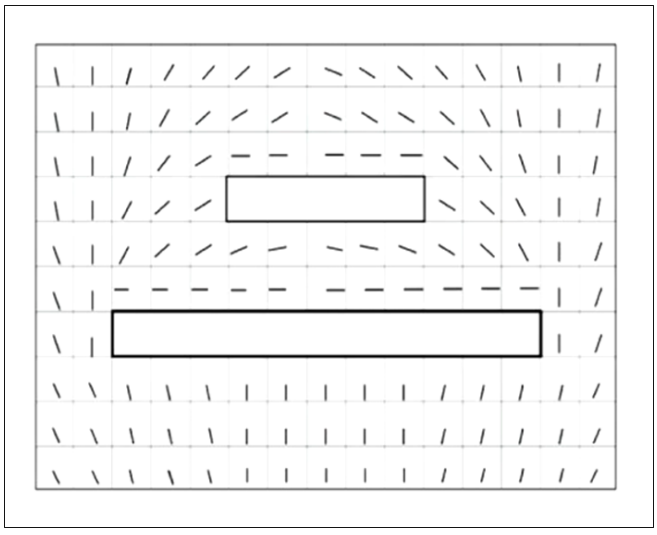

Fig. 10 - Flow Field cells' directions

8. HiDAC

HiDAC, short for High Density Autonomous Crowds, is a simulation model proposed by Nuria Pelechano, Jan M. Allbeck and Norman I. Badler in 2007 **[8]**. Its design addresses the issue of movement fluidity and realism in dense crowds, as its name entails, since the existing approaches at the time failed at doing so.

As the original paper mentions, most crowd simulation approaches can be divided into three groups: Social Forces models, Cellular Automata models and Rule-Based Models, and all have limitation when it comes to high-density scenarios. Social Forces, such as BOIDS, tend to lack individual personality due to their particle-like approach (as mentioned before in the BOID section). Cellular Automata fall under the limitations of environment partitions, such as discussed in the CA-SIRS section, which can become apparent in dense crowds. And lastly, Rule Based are lackluster when it comes to collision-detection without immersion-breaking caveats, such as waiting rules.

In HiDAC, agents are assigned different psychological profiles, similar to the OCEAN model, as well as physiological profiles, which correspond to the agent's locomotion capacity (like their energy level). The most relevant psychological trait used in the specific implementation described in the original paper is politeness – as less polite agents will push others, as illustrated bellow, while more polite ones will line up and wait for pathways to de-clutter.

Both these profiles are then used to calculate reactions depending on the perceived environmental conditions the agent faces (both static and dynamic). Physical and geometrical algorithms are also applied, so that a direct collision-response behavior is only applied in the case of a nearby obstacle detection, applying tangential forces instead to smoothly steer around distant obstacles.

The "pushing" behavior described in this approach is one I took inspiration from while developing the proposed model. Although not using the same physics-based logic as the one described, I opted to keep the inter-agent separation forces quite low in specific conditions, to allow agents to collide and push each other. The pushing method I used is quite simple, opting between one of the two colliding agents to move slightly in the opposite direction of the collision. The method to define agent priority withing collision is described in more detail in the "Case Study / Steering Agent Movement" section. The implementation of these simple collisions added a simple-to-implement yet visually cohesive layer of realism to the simulation.

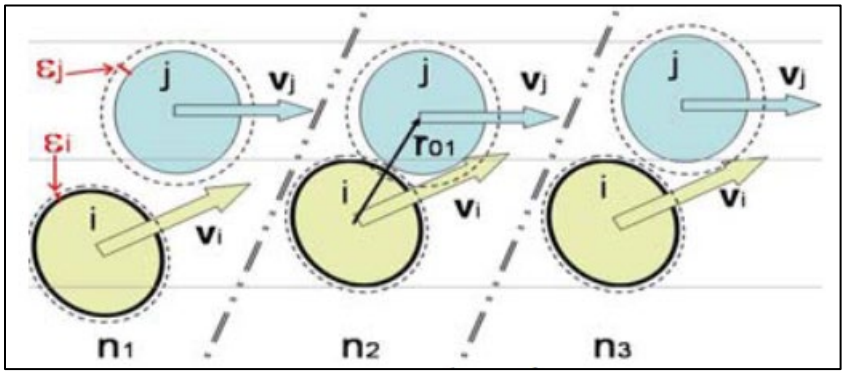

Fig. 11 – Collision response behavior, as described in HiDAC

9. Context Steering

Context Steering is an expansion of the previously mentioned BOID model, proposed by Andrew Fray in 2015 **[9]**. This expanded model was developed as an optimization of regular singular desired-velocity implementations, not only with resource management in mind, but also quality and fidelity of the final resulting movement in a macroscopic scale. The original application context of this model was the development of opponent vehicle movement AI for the game "Formula 1 2011", as enemy vehicles required a much more fine-tuned behavior than a regular BOID approach could offer, to avoid immersion-breaking collisions and unoptimized and/or unconventional steering routes.

The paper which proposes this model explains how regular flocking implementations lack individual context for each unit's desired action, other than their final velocity, making it hard to avoid bizarre blends of multiple steering behaviors with just prioritization and weights. To tackle this, the only solution within the limitations of older BOID models would be to expand the behavior at hand to be redundantly aware of both its environment and other non-directly applied steering behaviors, therefore making them more dependent on expensive and/or asynchronous operations like ray-casts and pathfinding algorithms, and ultimately also increasing coupling.

The presented solution to this issue is, instead of depending on a single desired velocity from each unit's behavior, basing agent's actions on context maps – containers of pre-defined directions and how strongly each one of them is desired/undesired. This added information increases the weight of the calculated data, but it avoids the before-mentioned expensive operations and coupling, and allows for more refined group steering-blending. The images bellow showcase both visualization of the direction containers (to the left) and the application of this technique in the original's paper context, with containers for both danger and interest of each car NPC's direction (to the right).

This method also allows for post-processing techniques to be applied as to improve the quality of the final movement, like blurring – taking the last update's context map and blending it with the current one, to avoid jarring flip-flops of direction/velocity.

Although I initially intended on applying a variant of this technique on the proposed model's implementation, it became clear during the Case Study's development that doing so would've not been beneficial. The fact I use Flow Fields for most target-based steering behaviors, as mentioned before, would've made the usage of Context Steering redundant, since the only application it would have had would be to avoid agent collision – which BOID-inspired separation forces already take care of (as mentioned in the "Related Work / BOID" and the "Case Study / 3. Steering Agent Movement" sections).

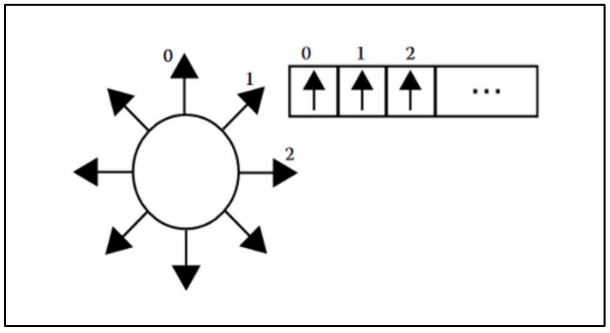

Fig. 12 – Agent direction container

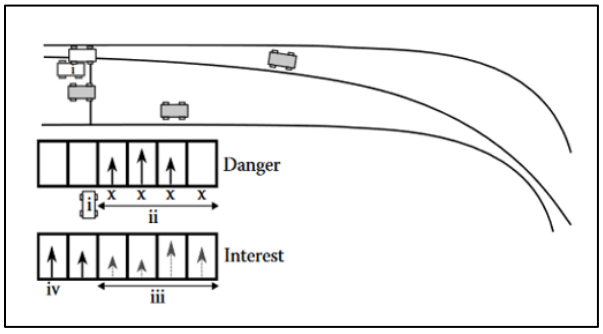

Fig. 13 – Danger/interest containers applied in agent steering calculation

CASE STUDY

1. Choosing Framework / Engine

From very early on in my research progress, I realized the framework I'd use to develop the simulation model I had been theorizing should share two traits: supporting 2D graphics and supporting C++ code, with the minimum possible extra functionalities. Long compilation times were also a characteristic I strived to avoid, since I knew from the start that developing this Case Study would require a lot of value tweaking and code restructuring – as testing out different techniques and models, even ones that would later be removed for not improving simulation quality, was a fundamental requirement.

The four engines I pondered about using were Unity, Unreal Engine 5.0, my personal 2D engine and the C++ engine provided by my supervisor, Kevin Hoefman. Out of those, I picked the C++ engine provided by Kevin Hoefman, since all other options had limitations which made them less fitting to the project at hand, specifically:

- Unity is mostly designed with C# in mind, and therefore implementing C++ code would require extra effort;
- Unreal Engine's C++ compilation times are known to be timely, and the engine also possesses a lot of functionalities that would be redundant to the scope of my Case Study;
- And my personal engine, although functional, is still a work in progress, and I preferred not potentially having to tweak or even debug the engine as I developed my model.

The provided engine is not only simple to get a grasp of and straight-forward to use, but its visual limitations are also appropriate to the type of minimal 2D graphics my project required.

2. Setting Up Scene

The scene used for the Case Study's simulation is a small floor (1024 by 768 pixels long), composed of 5 rooms in total. Four out of five of those rooms have at least two entrances, and all rooms have significant size-variance compared to each other, while respecting a general realistic uniformity. Each room is limited by thick walls (30 pixels wide), and the entrance points vary slightly in width (between around 60 and 120 pixels wide). At the top middle room, a single hazard is instantiated (the red square), occupying around 50% of the room's area with its radius of effect (the dark red circle around it). The exit (the green rectangle) is instantiated at the bottom right corner of the scene, at the end of the tightest yet longest corridor. The room layout forces most agents to run down this exit corridor while alert, creating a bit of realistic clogging and chaos, while still splitting up the crowd to a healthy degree with the multiple entrances for each room (including this corridor).

This scene layout was the result of a brainstorming and sketching process, of which my goal was to design a space which would combine both tight corridors and wider rooms, where certain rooms would be more isolated than others (as the middle room, with only one entrance), and where there would be multiple paths towards the exit. The sketch picked from this brainstorm process, along with a screenshot of the scene can be seen bellow.

All the walls that compose the scene are instances of a specific class, which recognizes collisions between a given rectangular shape and agents, pushing the agents slightly backwards. The hazard is a static entity, from which only the center shape has collision – as the area of contagion can (and should be) crossed by agents. Moving hazards have also been coded and could potentially be supported by the proposed model with minor changes to the agent steering logic (as explained in the following section, "Case Study / 3. Steering Agent Movement"), but have not been implemented into this final scene due to general time restrictions.

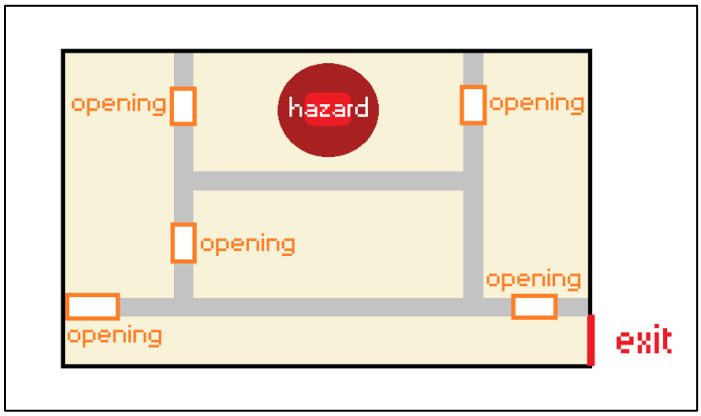

Fig. 14 – Scene layout inspiration sketch

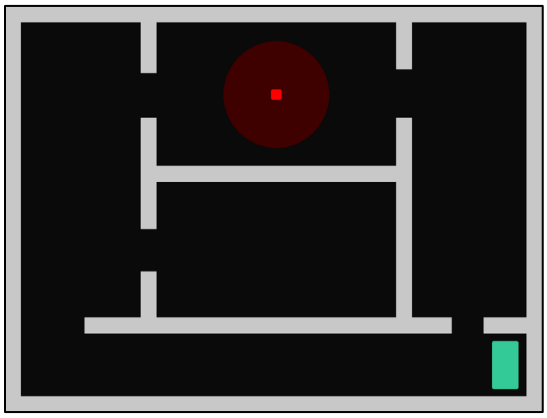

Fig. 15 – Empty scene screenshot

3. Steering Agents' Movement

For optimization purposes, considering the simulation handles 101 agents, six different static Flow Fields are initialized at start-up – one directed towards the exit and the other five directed towards the center of each room. These Flow Fields are responsible for dividing the scene into a given number of cells (for this implementation specifically, diving it into 44 columns and 33 rows), and set the cost of each cell depending on the distance towards the given target. Walls are given an invalid cell cost, making them uncrossable. The hazard-affected area is given a very high cost, higher the closest the cell is to the center of the hazard itself, making them still crossable but undesirable. The top middle room's Flow Field is the only one that ignores the hazard's area of effect while calculating its cells' costs, since the target overlaps the hazard itself.

To properly visualize these Flow Fields, a small interface was added to the top of the scene, so the player can switch out which cell-grid is rendered out. This interface is composed of six buttons, one for each flow-field. Clicking these buttons leads to the rendered cells' costs to be switched, as to match the button's number, as well as highlight the respective target room or exit. For the rooms, a dark-blue backdrop is added behind the grid, and for the exit, its rectangle visuals become a brighter green. Two extra buttons were also added to the top corners of the scene, so that the user can reset the simulation with and without the single authority agent (which is explained more in-depth in the "Case Study / 7. Authority Agents" section).

Each agent is initialized with pointers to these Flow Fields, so that they can easily request a given direction towards a room without the aid of more costly individual-based path-finding algorithms. Nonetheless, the agents don't only use these Flow Fields to move around the scene, applying a couple of individual low-cost performance-wise steering behaviors as well – namely Seek, Flee and Wander. These steering behaviors are implemented as individual classes, of which each agent initialized their own individual instance on startup and scene reset. The agent's movement speed can change according to the taken agent, switching between hard-coded "stroll", "walk" and "sprint" values.

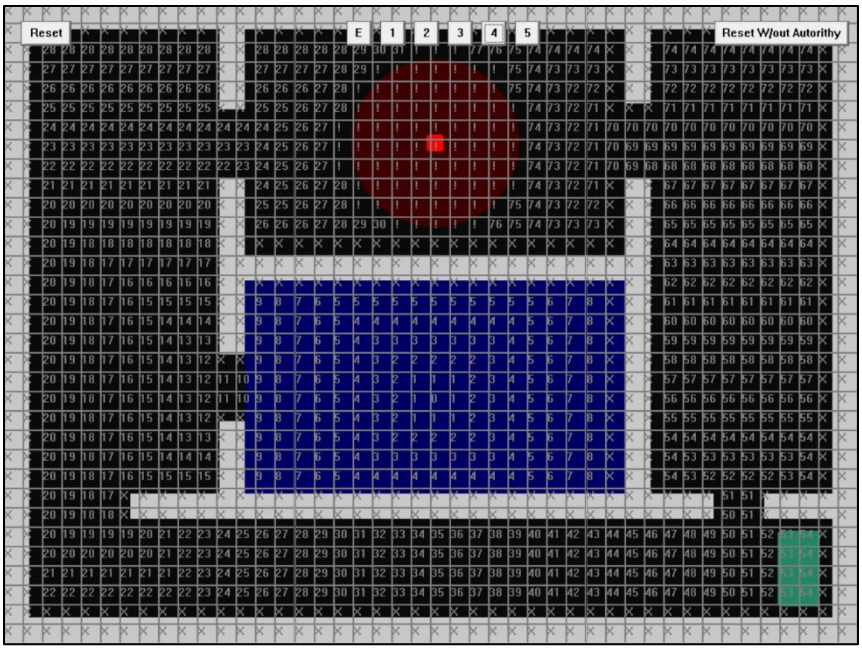

Fig. 16 – Cell grid for the middle room target, with the Flow-Field switching interface visible (on the top middle of the scene)

Extra forces are also taken into account when calculating the final agent steering, namely BOID-like separation forces which are applied to each agent, as to avoid each other and walls. The intensity in which these forces are applied is dependent on the action being executed, along with the emotional profile the agent possesses. More details on this emotional profile dependency are given in the following section, "Case Study / 4. Emotional Profiles". The mentioned separation forces are calculated by combining the Flee direction from each nearby agent, or each nearby wall, and then adding them up keeping into account the distance percentage to each target. With this, a normalized vector of the most optimized path to flee all targets at once is achieved. This logic is illustrated in a simple fashion on the illustration bellow, in which the final direction is portrayed by a yellow line, and the flee force from each target (the flee direction scaled by their distance) are portrayed as red arrows of different sizes.

Due to the adjustment of these forces' intensity, agents are still able to collide with each other on several occasions – such as when a regular agent is panicked, or a 3rd party agent is pushing through a mass of agents to change rooms). Due to this reason, agents are capable of pushing others, as well as being pushed, in case their movement leads them to overlap another agent – in a HiDAC inspired fashion. This pushing behavior is quite subtle, as the pushed agent is only coded to move in the opposite direction of the impact force enough to no longer overlap with others. The pushing itself follows a priority-logic, as to pick which agent is moved out of their way on collision – but this prioritization is further explained in the following section, since it's heavily influenced by the agent's emotional profile and current emotion-driven action.

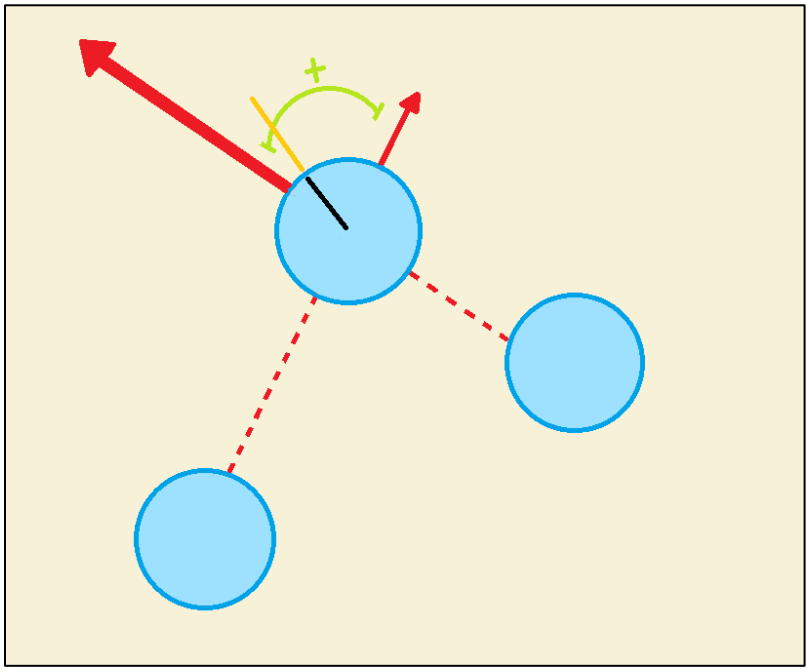

Fig. 17 – Inter-agent separation force calculations

4. Emotional Profiles

Each agent in this simulation is initialized with a 5 float struct (each ranging from 0.f to 1.f) correspondent to their OCEAN profile. These structs are crucial for the agents' action-picking and emotional reaction logic, since it's their application within steering logic that makes agents' behaviors feel unique and individual.

In this application, three emotions are depicted – Calmness, Fright and Panic. Each agent gathers emotional intensity for each of these emotions, most of the times simultaneously, through the influence of the people around them. This process is also affected by their OCEAN profile, but that's a detail I'll further explain in the section "Case Study / 6. Emotional Contagion Method". Only when the accumulated intensity of an emotion crosses a given threshold (which can be set as different values for each emotion) does it get assumed by the agent. If two emotions are currently surpassing their threshold, the assumed one will always be the one with the highest value. Since short periods of transition between emotions in which the highest valued emotion quicky switches back and forth are common, there's a fixed one second switching cooldown – as to avoid jarring "emotional flickering".

Each emotion can have multiple triggerable actions, which must depend on other individual conditions of the agent. That is the case for Calmness, as calm agents can both Wander (randomly move around their current room) and Explore (seek out new rooms curiously). For visual simplicity's sake, the other two emotions only have one triggerable action each – Evacuating for Fright, and Running Chaotically for Panic.

Wandering is implemented as a fully autonomous steering behavior. It picks a random direction within a given angle limit every frame and seeks it out. This angle limit isn't the only thing taken into consideration since, to make the randomness "smoother", that angle is applied within a temporary circle placed at a given distance from the current agent location, towards their current direction. This allows for both the distance of said circle and its radius to be tweaked, as to get different versions of similar wandering behaviors.

As for the other three actions, Evacuating and Exploring are implemented as simple seeks to the nearby cheapest Flow Field cell, and Running Chaotically is a mix of partially using the Flow Fields to change between rooms randomly, fleeing from the hazard (if panicked through hazard contact) and sprinting randomly with a tweaked Wander behavior.

This last "mixed" behavior's logic is illustrated bellow, where the Flee direction/ strength is represented by the red arrow, the evacuate by the green arrow, and the random wander is by the orange arrow. The yellow arrow is the final steering direction, calculated from the addition of the other three, taking into account the intensity of each of them.

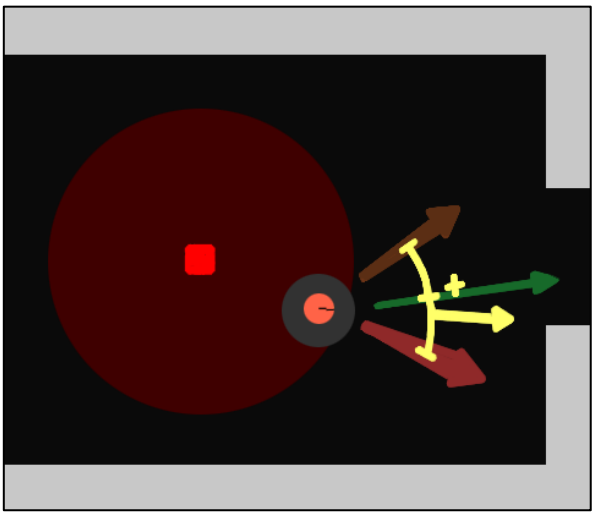

Fig. 18 – Running Chaotically's direction calculations

The way calm agents switch between the two available actions is also dependent on their OCEAN profile.

Every agent starts off as calmly wandering and, after a while, switches into exploring – until a new room is reached, that is, at which point they'll start wandering again. The time it takes for each agent to get "bored" of their current room is what differs from individual to individual, as this timer is recalculated every time the Room Changing action is triggered, being set as a random value between a minimum and a maximum cooldown, both defined by the agents Openness value during initialization. Specifically, in this implementation, the minimum cooldown can range between 1 (at 1.0f Openness) and 6 seconds (at 0.0f Openness), and the maximum cooldown can range between 6 (at 1.0f Openness) and 12 seconds (at 0.0f Openness).

As mentioned before, the inter-agent collisions are also mostly sorted out by the involved agents' OCEAN profile and current action.

The first decisive factor are the actions that the involved agents are executing. Actions themselves have an inherent "urgency" value, which allows them to be organized priority-wise. Panicked agents will always push others (aside from authorities), followed by evacuating agents, then agents calmly changing room and, lastly, wandering agents.

If both colliding parties are executing the same action and are both regular agents and not authorities, the OCEAN profile is used as the decisive factor. The higher an agents' Extroversion level, the more priority they should take – therefore getting pushed around less often.

Fig. 19 – Inter-agent collision priority

The agent's render color is defined by the current action they are executing – as light blue signifies Wandering, yellow stands for Switching Rooms, green for Evacuating, and red for Running Chaotically. That is not the only emotional visual aid present in the implementation though, as many other agent-based emotion-related values need to be easily observed to properly analyze the simulation. As such, the user can click on an agent to observe it, opening a widget to the right side of the window, which showcases all the agent-unique information. This includes the current emotion, the current action, the accumulated value for each emotion, each OCEAN trait's value portrayed as a bar and the memorized rooms along with the agent's memory time.

The last two listed pieces of information illustrated in the widget are further explained in the following section, "Case Study / 5. Individual Memory". The agents' emotional profile is used for many other decision-making situations, which are further described in the "Case Study / 5. Individual Memory", the "Case Study / 6. Emotional Contagion Method" and the "Case Study / 7. Authority Agents" sections.

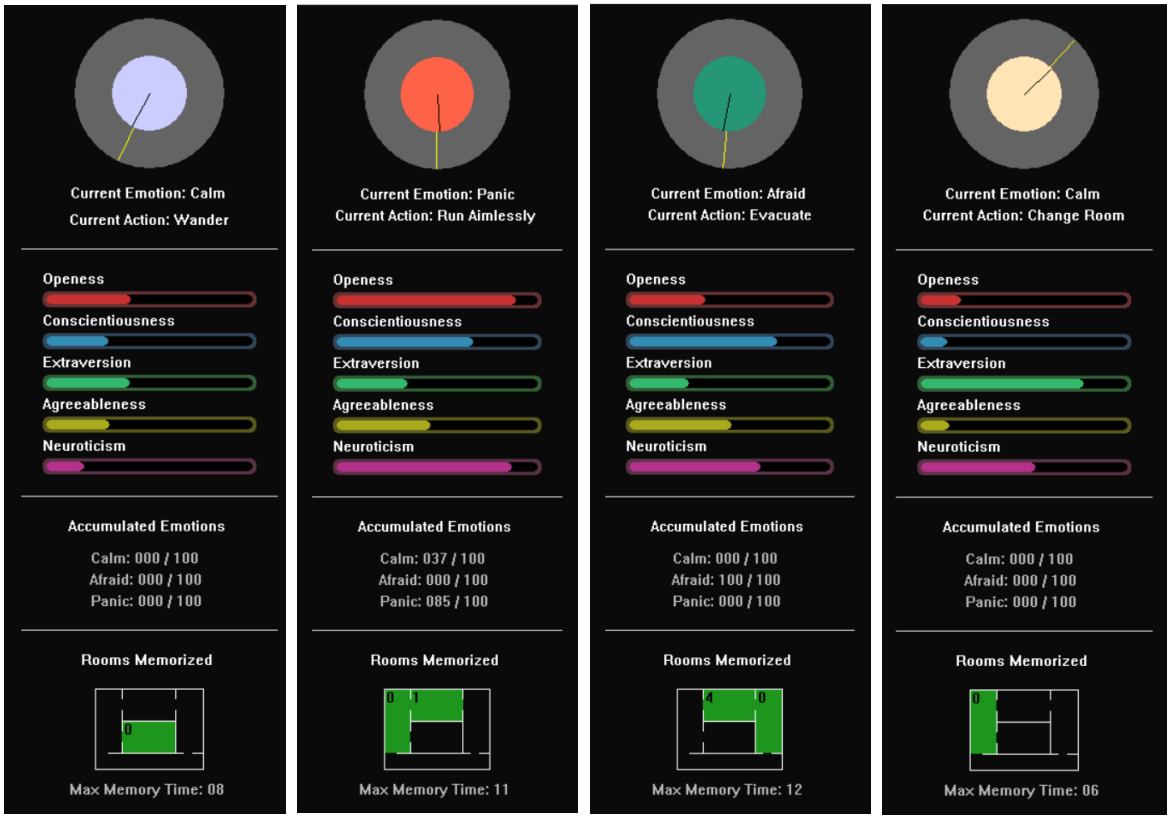

Fig. 20 – Different agents' information widgets

5. Individual Memory

The agents' decision-making structure is also affected by a memory-simulating logic. This logic is mostly based around a single container of Boolean/Float pairs, initialized with one item for each room. The first element of each pair corresponds to if a room has been memorized or not, and the second corresponds to the amount of time passed since the agent's been in said room. Every time an agent walks into a new room, they turn the first element of the pair correspondent to said room into to true, and start counting the time for the room they've came from, along with all the remaining pairs set to true (aka, all the rooms they still remember but aren't currently on). If any room's timer exceeds the agent's max memory, they are forgotten (being set to false/0.0f) again.

When initialized into the scene, agents have no recollection of any of the rooms in the floor, except the one they are spawned into. This entails their memory container is initialized with all pair elements set to false/0.0f, except for the one corresponding to their spawn room. From then on out, while steering, agents are only able to access the Flow Fields correspondent to the rooms they saved in memory, along with they're current room's neighbors. This limitation forces the agents to act with reductive environment information, often leading them to make decisions that would otherwise be non-sensical if they understood the floor's full layout.

While calmly exploring, agents will prioritize rooms they still don't have saved in memory. If all neighbors are already known, the target is picked randomly between them, preferring rooms they've visited longer ago. While evacuating, agents will check if any of the rooms they currently remember possesses the exit and, if so, will proceed to sprint towards it, with the aid of the Exit Flow Field. If they haven't encountered the exit yet, they'll sprint through different rooms until they do so, using a similar logic to the calm exploration.

The memory time each agent is endowed with is defined by their OCEAN profile on start-up, namely by their Conscientiousness level. The more conscientious an agent is, the more likely they are to remember a room for longer. For this implementation, the minimum possible time is set to 3 seconds (at 0.0f conscientiousness), and the maximum 6 seconds (at 1.0f conscientiousness).

The information of the agents' memory time, known rooms and the time passed since each room as been visited by them is all illustrated and observable through the before mentioned widget, as illustrated bellow.

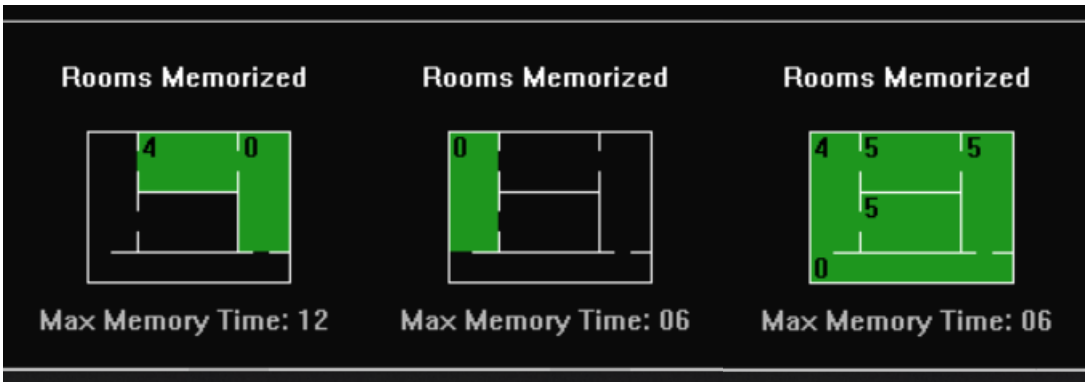

Fig. 21 – The memory-related portion of several agents' information widgets

6. Emotional Contagion Method

There are two triggering methods from which agents can acquire emotional influence: from being in the vicinity of each other, and from being within the hazard's range of effect. Both methods share almost identical mathematical solutions for propagation calculation, taking the agents' OCEAN profile into account, but differ in application.

For inter-agent influence, each individual is responsible for calculating how the emotions of the close-by crowd influence them, every frame. They do so by incrementing the "accumulated emotional intensity" (mentioned in "Case Study / 4. Emotional Profiles") correspondent to each nearby agent's active emotion. The closer an agent is, the stronger their influence should be. To achieve this, only agents closer than a given distance are observed (for this application, within a 150 pixels radius), and the emotional influence value is multiplied by the distance percentage of each agent (their current distance to the influenced agent divided by the maximum distance). The emotional influence value itself is calculated by multiplying the emotion's fixed Spreading Rate (which is hard-coded pre-simulation,) by the frame's elapsed time, and then multiplying it with the agent's Agreeableness level.

Lastly, Panic and Fright assimilation are also dependent on the agent's Neuroticism level, so that the more neurotic an agent is, the more likely they get panicked instead of afraid (since fright leads to evacuation, while panic leads to running aimlessly). To implement this, in the case of these two emotions, the influence value of each frame is also multiplied by the agent's Neuroticism level, or the invert of said value for the case of Fright (1.0f minus the level).

The hazard's influence, as mentioned before, behaves very similarly to the contagion described up to this point. The only few differences are that the influence added each frame is multiplied by the hazard's danger level (in this case, the hazard's danger level is set to 5.0f), the Openness value is ignored, and the emotion the hazard emits is adjusted according to each agent's OCEAN profile, namely their Neuroticism (once again). The more neurotic an agent is, the more panic the hazard will cause them – while still causing a certain amount of fright nonetheless.

Similar fixed values to the described Spreading Rates are used for emotional-stabilization calculations. Independently of what emotion is assumed, all accumulated emotional influence is decreased steadily every frame, making agents able to passively switch back from intense emotions if their perceived triggers are no longer close. An example of this can be an agent that gets panicked while having close contact with a single panicked individual, but later returns to being calm once surrounded by non-alarmed wandering agents.

The speed at which accumulated emotional influence decreases is defined by the Attenuation Rate of each emotion, multiplied by elapsed time and by the opposite percentage of their Neuroticism level (in other words, 1.0f minus the Neuroticism level). The rates for each emotion used for the described application are as follows:

- Calm – 0.1f Spread / 0.3333f Stabilization;
- Fright – 0.9f Spread / 0.3333f Stabilization;
- Panic – 0.75f Spread / 0.3125f Stabilization.

7. Authority Agents

Aside from the 100 regular agents that fill up most of the scene, a single agent, whose architecture is inspired by the Third Party Agent model (described in “”), is initialized in the described implementation.

This single agent always has memories of every room in the scene, does not possess an OCEAN profile, nor accumulates emotional influence like others. Their actions are independent of emotions and are instead chosen according to a container similar to the room memory information, a single Boolean variable labeled “IsAlert” and a few simple extra conditions. The mentioned container is used as to keep track of what room the agent has made sure have no more uninformed or panicked agents in, once they start helping everybody evacuate.

Although they don’t get influence by others emotionally, they do influence others emotionally – and quite heavily so. Any regular agent that gets near an alert authority will get influenced as if approaching a frightened agent, but their influence is increased exponentially. Specifically, the final influence value from each frame is generally increased by 50 percent. On the other hand, Neuroticism is, once again, applied to simulate individualized reactions, making it so that the more neurotic an agent is, the less likely they are to be receptive towards an authority figure – adding a cut of up to 50 percent on the influence’s final value.

The authority agent is always initialized in the most centered room of the scene, with its main decision-making variable set to false. This variable is only turned to true once the agent either encounters the hazard directly or notices a distressed agent in their vicinity (panicked or frightened). Until this happens, the agent just travels between all rooms, taking no time to wander and avoiding coming back to a room twice.

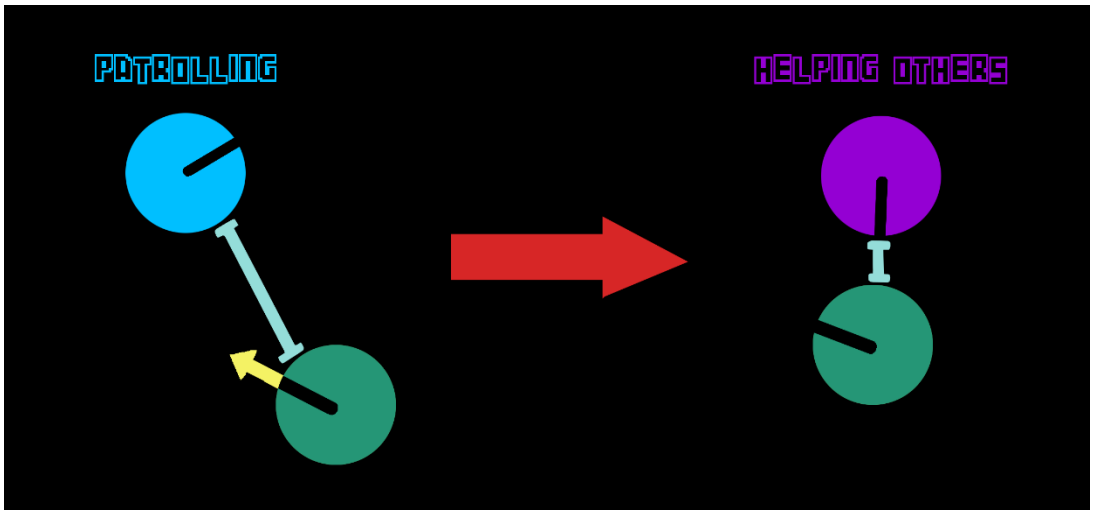

Fig. 22 – Authority agent becoming alert through proximity with frightened agent

Once the agent does become alert, they'll start switching between three possible actions: Warning Others, Changing Room and Evacuating.

The first action is automatically executed once turned alert, and it consist of going over every agent in the current room, checking which of the non-evacuating ones is the closest, and running up to them.

Once the room is fully clear, its index is saved to the before mentioned container dedicated to the evacuated rooms, and the second action starts being executed. This second action is similar to the regular agents' Explore New Room behavior, but it can be interrupted at any time if any agent in the old room turns back to being calm or panicked (updating the container as well, if that does happen).

The third action is the same as any other agents' evacuation behavior, but it only gets triggered once the authority is the last missing person in the floor. If every room is marked as checked, but the authority agent is still not the only one left in the scene, they'll reset the container and start going over each room again, until the straggling agent is found.

The addition of this guiding agent really improves the general flow of the simulation, as isolated agents are way less likely to fail to notice the hazard, and panicked agents become less abundant. Regardless of these improvements, the widget also includes a button for the sole purpose of resetting the simulation without the authority agent, so the user can better visualize the natural contagion dynamics between regular agents.

Fig. 23 – Screenshot of the reset button without authority, within the simulation interface

RESULTS

The developed case study achieved satisfying results considering its intended purpose – as it manages to portray believable human-like dynamics within a hundred-agent crowd while in an enclosed space. Not only are all the portrayed agents' actions heavily dependent on individual-based psychological traits, making each actor feel unique, but the movements of the crowd as a whole feel reasonably natural as well – cluttering in tight spaces; allowing agents to be oblivious of the danger of their situation if isolated and far away enough from the hazard; and recreating swarms of fleeing individuals moving between several rooms without any issues.

The introduction of authority agents also proved to have a positive effect in the simulation quality, as the difference between the time a crowd takes to evacuate with and without a single instance of this agent-type is significantly large, as it should be realistically. Over 25 tests without authority present, agents took a median of 139.5 seconds to evacuate the scene (being the minimum value 67.2 seconds, and the maximum 431.4). With authority present, in 100 tests, the median diminished to 58.5 seconds (being the minimum value 29.7 seconds, and the maximum 138.6).

Aside from the overall positive results, this simulation method also proved to be quite light processing-wise, as tested in a plugged-in laptop running a 64-bit operating system, with x64-based Intel® Core™ i7-7700HQ CPU @ 2.80GHz 2.81 GHz processor, with 16.0 GB of available RAM. Running the simulation at the intended 50 FPS, it utilizes around 0.1GB of memory and 5% to 6% of the Intel® HD Graphics 630 GPU. Considering the small size of the scene, no tests were ran with more agents, since there would physically be no space for said agents to steer within the confines of the rooms.

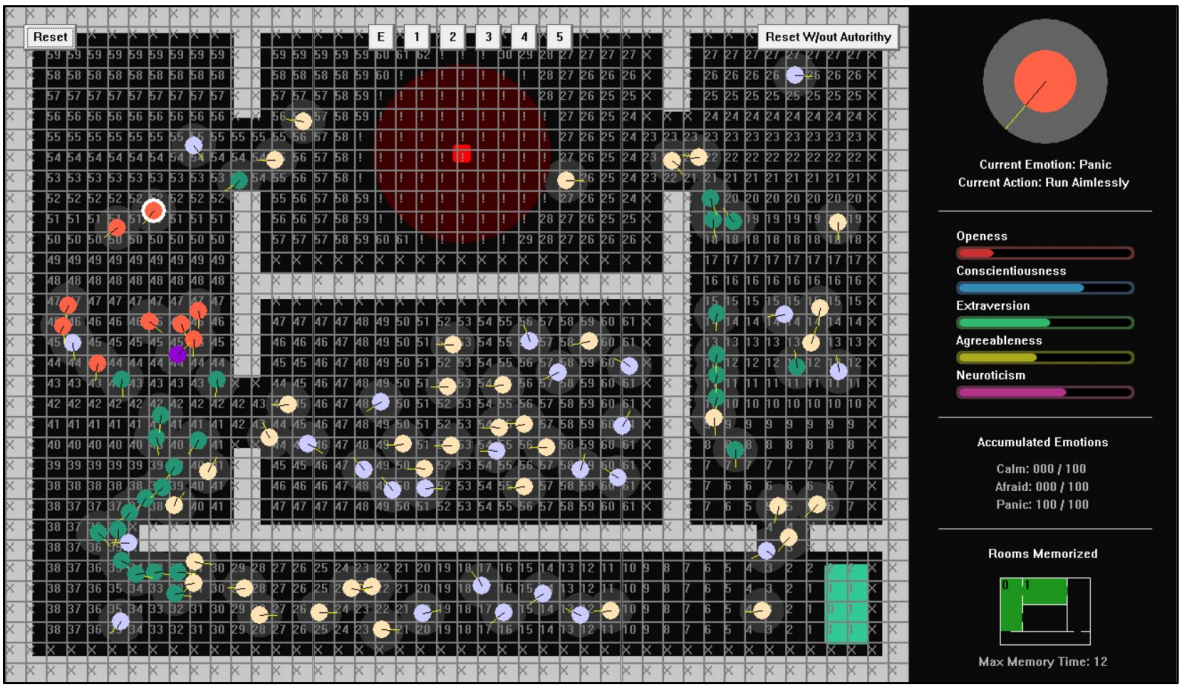

Fig. 24 – Screenshot of the developed simulation

CONCLUSION & FUTURE WORK

During these last decades, the simulation of emergency situations, such as natural accidents and human-made dangers, has proven itself to be a useful and practical tool for strategic planning of accident preventive methods. Furthermore, the usage of simulation models such as these extends further than public health/safety, as it can be applied in a myriad of different digital mediums for entertainment purposes, such as video-games.

This paper proposes an approach to the simulation of these situations, inspired in both seminal and state-of-the-art findings, with a focus-goal on replicating the relationship of dynamic emotion contagion and its relationship with authority figures. This approach proved to achieve the proposed goal in a satisfying manner, as described in the "Results" section, but it can still be expanded on quite a lot in the future.

The first clear improvement would be implementing more emotions for each agent to handle – such as "Disgust", "Curiosity" and "Laziness" – as well as more complex dependencies between them – such as considering the used "Panic" and "Fright" secondary emotion to a primary emotional spectrum of "Stress". This could lead to more realistic agents, with more individualized yet expressive personalities.

The second possible improvement would be to expand on the agents' available actions – specifically allowing some agents to surround the hazard out of curiosity, or flee slower, looking back at the hazard once in a while. These actions could potentially even lead to the aggravation of the hazard, such as, in the case of a fire, allowing agents to catch on fire and unintentionally increase the hazard-affected area. Some of the mentioned behaviors were initially intended to be included in the Case Study, and some were even partially developed, but got later refactored due to time constraints.

Lastly, for the sake of improving the simulation tool on a user standpoint, the generation of hazards in real-time would be an interesting yet achievable goal – allowing users to put down and delete hazards at will, while the simulation is already running.

BIBLIOGRAPHY

[1] YANG, S., LI T., GONG X., PENG B. and HU J. 2020. **"A Review On Crowd Simulation And Modeling"**. Graphical Models, Vol. 111, 101081.

[2] MAO Y., LI Z., LI Y. and HE W. 2018. **"Emotion-based diversity crowd behavior simulation in public emergency"**. *Vis. Comput., 10.1007/s00371-018-1568-9.*

[3] MAO Y., YANG S., LI Z. and LI Y. 2018. **"Personality trait and group emotion contagion based crowd simulation for emergency evacuation"**. *Multimedia Tools Appl., 10.1007/s11042-018-6069-3.*

[4] GOLDBERG, L. R. 1990. **"An alternative "description of personality": The Big-Five factor structure"**. *Journal of Personality and Social Psychology, 59, 1216-1229. DOI: 10.1037//0022-3514.59.6.1216.*

[5] KERMACK, W. O., MCKENDRICK, A. G. 1927. **"A contribution to the mathematical theory of epidemics".** *Proceedings of the Royal Society of London. Series A, Containing Papers of a Mathematical and Physical Character.*

[6] REYNOLDS, C. 1987. **"Flocks, Herds, and Schools: A Distributed Behavioral Model"**. *Computer Graphics, Vol. 21, No. 4.*

[7] PENTHENY, G. 2013. **"Crowd Simulation Through Steering Behaviors and Flow Fields"**. *https://gdcvault.com/play/1018262/The-Next-Vector-Improvements-in*

[8] PELACHANO, N. , ALLBECK, J. M., BADLER, N. I. 2007. **"Controlling Individual Agents in High-Density Crowd Simulation"**. *Proceedings of the 2007 ACM SIGGRAPH/Eurographics Symposium on Computer Animation, 99-108.*

[9] FREY, A. 2015. **"Context Steering - Behavior-Driven Steering at the Macro Scale"**. *Game AI Pro 2, CRC Press.*

[10] NEUMANN, J. V. 1951. **"The General And Logical Theory Of Automata"**. *In L. A. Jeffress (Ed.),* Cerebral mechanisms in behavior; the Hixon Symposium *(pp. 1–41). Wiley.*

[11] 2010. **"John von Neumann's Cellular Automata".** *Embryo Project Encyclopedia* (2010-06-14). ISSN: 1940-5030 http://embryo.asu.edu/handle/10776/2009.

[12] GARDNER, M. 1970. **"The fantastic combinations of John Conway's new solitaire game 'life'"**. *Mathematical Games. Scientific American. Vol. 223, no. 4. pp. 120–123. doi:10.1038/scientificamerican1070-120. JSTOR 24927642.*

[13] BILL & MELINDA GATES FOUNDATION. 2022. **"SIR and SIRS models"**. https://docs.idmod.org/projects/emod-generic/en/2.20_a/model-sir.html

[14] GEORGEFF, M., PELL, B., POLLACK, M., TAMBE, M., WOOLRIDGE M. 2003. **"The Belief-Desire-Intention Model of Agency"**. *Lecture Notes in Computer Science, Proceedings of the 5th International Workshop on Intelligent Agents V, Agent Theories, Architectures, and Languages.*

[15] ORTONY, A., CLORE, G.L., COLLINS, A.M. 1988. **"The Cognitive Structure of Emotions"**. *Contemporary Sociology, Vol. 18, No. 6 (Nov., 1989)*

[16] YERKES, R. M., DODSON, J. D. 1908. **"The Relation Of Strength Of Stimulus To Rapidity Of Habit-Formation"**. *Journal of Comparative Neurology and Psychology, 18, 459-482*

[17] TYLLER HOBBS. 2022. **"Flow Fields"**. https://tylerxhobbs.com/essays/2020/flow-fields