PSYCHOLOGICAL AND SOCIO-PSYCHOLOGICAL FACTORS IN BEHAVIORAL SIMULATION OF HUMAN CROWDS

Jacob Sinclair, James Cook University, Australia, Jacob.sinclair@my.jcu.edu.au
Carrie Siu Man Lui, James Cook University, Australia, Carrie.Lui@jcu.edu.au

ABSTRACT

This paper aims to synthesis existing research efforts to provide an integrated view of behavioral model designs and relevant theoretical frameworks of heterogeneous agents for crowd simulations. Most existing studies considered only limited parameters by including a few selected personalities traits, emotion, and group characteristics for specific scenarios and applications. Most often, these factors are implemented with limited reference to theoretical psychology and cognitive models. This study attempts to synthesis existing research effort and outlines opportunities, challenges, and promising areas for future research for integrating psychological and socio-psychological factors in crowd behavior simulations.

Keywords: Crowd simulation, personality, cultural, emotion.

INTRODUCTION

Simulating realistic movement and behavior of crowds is important in many application areas, including computer animation, video games, pedestrian dynamics modeling and analysis, virtual reality, emergency management, building planning, and massive event management. Approaches for crowd behavior simulations can be classified into macroscopic approaches and microscopic approaches [1]. Macroscopic approaches consider the crowd as a whole, focusing on the flow characteristics of the crowd as a single entity. Example of macroscopic approaches include regression statistics models [2], route choice based on agent's utility [3], queuing models [4], fluid dynamics [5] and gas kinetics [6]. Although macroscopic approaches are adequate for simulation crowd movement en masse, these approaches ignore the heterogeneity of individuals in the crowd, therefore cannot simulate varieties of individual behaviors in crowds caused by individual differences such as cultures, personalities and emotions. With microscopic approaches, a crowd is modeled as a group of individuals. Each individual in the crowd can make his or her own decisions, respond to the environment, and interact with other individuals. Examples of microscopic approaches include rule-based approaches [7], cellular automata models [8, 9], and social force models [10, 11]. With these approaches crowd behaviors is not explicitly defined but emerges from the sum of the individuals decisions. A general surveys of crowd simulations studies can be found in [1, 12].

Agents in crowd simulations can be homogenous or heterogeneous. Homogenous agents have the same agent's attributes values [9, 13]; hence will exhibit the same behaviors when encounter the same environment and stimuli. Heterogeneous agents have different agent attributes values or profile; hence behaviors of the agents are depends on both the environment and stimuli external to each agent and also individual differences of each agent. Ideally, the behavioral model of a human-like agents should be realistic, theoretical supported, and efficient. However, in practice, there are usually tradeoffs. To make crowd simulation with acceptable high frame rate, realism aspects required high computational power are usually comprised. As a results, different crowd simulations employ different behavioral models, different agent profiles, and different theoretical grounds.

Currently, the most commonly used microscopic crowd simulation architecture is to model the crowd as a group of Belief-Desire-Intention (BDI) agents [14] within a multi-agent system. Each agent is given human-like decision-making processes to determine their own decision and behaviors. In order to study the complexity of crowd behaviors and simulation realistic virtual crowds, various studies attempts to incorporate personality, emotions and social groups in crowd simulations. However, many implementations only use one or two dimensions of the corresponding psychological models and tailor the mapping of these affective aspects directly to agents parameters in limited reference to the underlying theories and for limited scenarios.

There are needs for a psychology and socio-psychology theories driven models for agent-based crowd simulation implementations. This study aims to survey the psychology and socio-psychology theories and factors used in existing crowd simulations literatures and also theories and factors in crowd behavior studies to identify the opportunities and challenges for future crowd simulations research.

This objectives of this study are 1) to synthesis existing research efforts to provide an integrated view of behavioral model designs and relevant theoretical frameworks of heterogeneous BDI agents for crowd simulations and 2) to identify research gaps and challenges for future agent-based crowd simulations. While it is clearly impossible for this study to include all existing research efforts that have contributed significantly to the topic, we aim to cover a good mix of the systems that are representative of a diverse range of approaches and disciplines.

Theories related to collective behaviors such as Contagion theory [15], Emergent-Norm Theory [16] and Convergence Theory [17] suggest that crowd behavior is not a product of the crowd itself, but an emergent property of individuals which are

influenced by and influencing other individuals. In order to model and simulate crowd behaviors, it is important to properly design the behavioral model of the individual agent and the interactions among the agents.

In multi-agents systems, behavioral models are used to represent the theoretical characteristics of the agents. The behavioral models specify how perception of the environment and stimuli are processed to determine the best next action to achieve a specific goal. BDI model is popularly used and prevalent model in agent-based crowd simulation. BDI model is developed based on the theory of human practical reasoning [18] and theory of intentional systems [19]. An agent's behaviors are determined by its belief (information the agent has about the environment), desires (the agent's goals) and intentions (the agent's committed goal to achieve). Although there are other human agents behavioral models such as Soar [20] and Act-R[21], BDI offers the most straightforward representation for describe human reasoning and actions [14]. For this reason, this study use BDI as the foundations and aims to related psychology and soci-psychology theories of crowd behaviors for extending the BDI model to allow understanding of impacts of these affective aspects on agent's decision making, agents communications and agents behaviors. The high level simulation framework is shown in Figure 1.

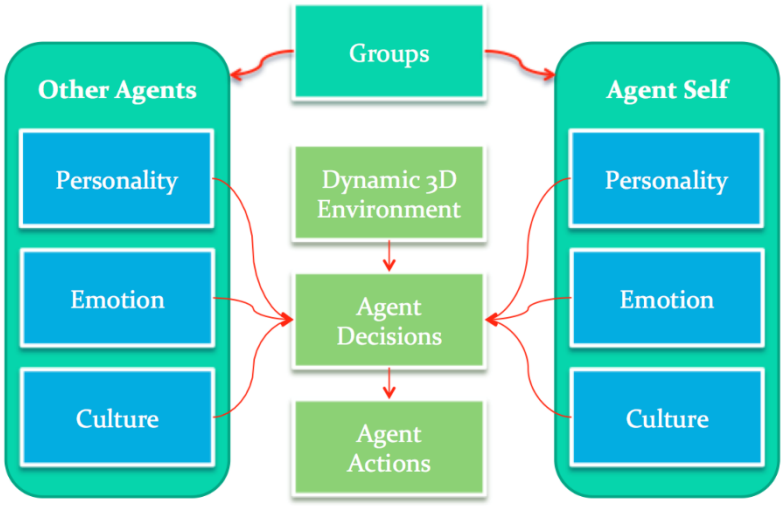

Figure 1. Psychological and Socio-psychological Aware Framework for Behavioral Crowd Simulation

In the proposed framework, psychological and socio-psychological factors such personality, emotion, and culture will affect agent's own decision making process. Among agents, the interactions among agents will be affected by culture and group norms. Emotions will be affected by agents interactions and some emotions may also propagate among agents.

PERSONALITIES IN CROWD SIMULATIONS

Personalities are always created using agents parameters as the focal point providing the ability to display realistic movement and reactions around other agents and obstacles. But personality can be developed within the path finding allowing agents to determine their path based on their personality. Guy et al. [22] provides agents with the parameters to display their personality but uses no path finding capabilities. Guy et al. [22] uses an RVO2 library which provides an easy-to-use implementation of the Optimal Reciprocal Collision Avoidance (ORCA) formulation for multi-agent simulations. This tool provides the ability to setup simulation scenarios by setting agent default parameters (max. neighbours distance, max. number of neighbours, planning horizon, obstacle planning horizon, agent radius and max speed), setting up agent starting positions, and adding obstacles specific vertices in counter clockwise order to create an invisible barrier around objects in which an agent is unable to walk through and move agents by using a preferred velocity.

Although RVO2 is a useful library to use you still are provided to generate all the agents and the environment yourself. RVO2 also doesn't provide any path finding capabilities only allowing agents to move directly to a goal using the preferred velocity. RVO2 requires the developer to incorporate a path finding solution into it and connect it to the preferred velocity allowing agents to move around the environment in a more realistic manner. It also lacks the ability to allow agents to move up or down an environment, this is because RVO2 us 2D vectors to move the agents and create the invisible wall around objects. Guy et al. [22] does not provide a path finding to the RVO2 library forcing agents to go to the goal directly without thinking of the best path or a path picked based on their personality. But by adding a path finding solution that considers the agents personality in mind, will help produce more realistic agent behaviours and decisions in crowd simulation. RVO2 was used in this project to help setup the simulations with the inclusion of a self-developed local path finding. The local path finding developed relies on the agent's personality to decide what path an agent should take to reach a set goal.

Three personality traits were implemented in developing realistic agents; these were shy, impulsive and aggressive. Guy et al. [22] had already mapped the parameters to each of these personality traits making it easier to implement them into the project.

The only difference between Guy et al. [22] is that the radius of each personality has been cut in half allowing better movement between agents in large crowds (see Table 1). But the code that includes personalities into the local path finding is original.

Table 2. The agent parameter's for each personality.

Traits	Max. Neighbour Distance	Max. Number Neighbour	Planning Horizon	Obstacle Planning Horizon	Agent Radius	Max Speed
Aggressive	15	20	31	31	0.6	1.55
Impulsive	30	2	90	90	0.4	1.55
Shy	15	7	30	30	1.1	1.25
No Personality	15	10	10	10	1.0	2.0

The local path finding used is a point to point network in which the agents move around the environment moving from point to another point that has some connection to it; in this case it is a neighbouring point. At the start of a simulation the agents move to the closest point from where they were spawned, from there on they rely on their personality to determine the next point until they are close enough to the goal. Each personality trait determines what path an agent should go down differently. An agent with an aggressive personality looks for the quickest path to the goal (see Figure 2). This has been developed into the path finding by selecting the closest point to the goal from a list of neighbouring points given to the agent from the current point the agent in which the agent is located.

```
findAggressiveRoute(){
        Array of Neighbour Points = Get all neighbourPoints from CurrentPoint;
        Float goalDistance = Get distance between the goal and the agents position;
        Float closestDistance = 10000.0;
        RVO.Vector2 position;
        Set previousPoint as the currentpoint;
        For every Neighbour Point in the array{
                Float distance = get distance between the goal and the Neighbour  Point position
                If the distance is less than or equal to the closestDistance and the blockedPoints doesn't contain the
                Neighbour Point{
                        Set distance to the closestDistance;
                        Set Neighbour Point position to the position;
                        Set Neighbour Point to the currentPoint ;
                }
                Else if the blockedPoints does contain the Neighbour Point{
                        If the distance is less than or equal to the closestDistance and the blockedFrom are not equal to
                        the previousPoint{
                                Set distance to the closestDistance;
                                Set Neighbour Point position to the position;
                                Set Neighbour Point to the currentPoint ;
                        }
                }
        }
        If the goalDistance is closer than the closestDistance && the goalBlock is false{
                Set goal as targetPosition;
        }
        Else{
                Set goal as targetPosition;
                Set goalBlock to false;
        }
}
```

Figure 2. Path finding for aggressive personality's by finding the shortest route.

Agents with an impulsive personality don't go for the quickest path to the goal but for more go for a more adventures path (see Figure 3). The path finding incorporates the impulsive personality by selecting the second best path to the goal allowing impulsive agent to take a more adventurous path, except when there is only one path to the agent's goal the impulsive agent will then take that path instead.

```
findImpulsiveRoute(){
        Array of Neighbour Points = Get all Neighbour Points from CurrentPoint;
        Float goalDistance = Get distance between the goal and the agents position;
        Float furthestDistance = 0.0;
        RVO.Vector2 position;
        Set p as the previousPoint;
        Float currentDistance = Get the distance between the goal and the currentPoint;
        For every Neighbour Point in the array{
                Float distance = get the distance between the goal and the neighbour point;
                If distance is less than currentDistance and the Neighbour Point is not equal to p{
                        If distance is greater than furthestDistance and blockPoints does not contain Neighbour
                        Point{
                                Set distance to the futherestDistance;
                                Set Neighbour Point position to the position;
                                Set Neighbour Point to the currentPoint ;
                        }
                        Else if blockedPoints does contain the Neighbour Point{
                                If distance is greater or equal to the furthestDistance and blockedFrom is not
                                equal to previousPoint{
                                        Set distance to the futherestDistance;
                                        Set Neighbour Point position to the position;
                                        Set Neighbour Point to the currentPoint ;
                                }
                        }
                }
        }
        If the goalDistance is less than or equal to 20.0f and the goalBlock is false{
                Set goal as targetPosition;
        }
        Else{
                Set goal as targetPosition;
                Set goalBlock to false;
        }
}
```

Figure 3. Path finding for impulsive personality's by finding the second best route.

Agents with a shy personality are very different to aggressive and impulsive agents as they don't focus on finding the quickest path or second best path to reach their goal by comparing the neighbouring points from the current point from where the agent is located (see Figure 4). Instead their main focus is to take a path to their goal that has the least amount of agents to interact with. The path finding determines this by how many agents are between the current point and each neighbouring point using a detector that has been place between each one. A shy agent then takes the point that has the least number of agents between them, which can end up making shy agents move further away from their goal.

EMOTIONS IN CROWD SIMULATIONS

Emotions have been used in crowd simulations in different ways, one type used is levels [23] in which a single emotion such as horror is represented by several basic emotions of horror represented in a level system starting from calm, to alarm, fear, terror, panic and all the way up to hysteria. Emotions can be affected by different situations in the environment as long as it involves the agent in some way. For example a fire breaks out on one side of a town, the people on that side are emotionally affected but the people on the other side may not be emotionally affected. Other agents can also affect agent emotions in different way such as disagreements are decisions and feeling these emotions from the agents around them.

Agent emotions were incorporated into the project differently to a level and to individual emotions. A threshold system was developed to represent multiple agent emotions for the project. The threshold system works by using two opposite emotions like happy and sad or angry and calm and setting high threshold to one and a low threshold to the other. This prevents an agent from feeling both emotions at the same time as an agent cannot be happy and sad or angry and calm. Each two opposite emotions are given value to share that can be increased or decreased based on different situations. If that value reaches or pass one of the thresholds it means that the agent feels that emotion and is affected by it.

Eight emotions were used and split into four groups of opposites to represent the thresholds used in the project. These emotions were happy and sad, stressed and relaxed, angry and calm, excited and bored (see Table 2). These emotions were

chosen as they each are an opposite of one of the emotions and because when split into their four groups when a threshold has been reach each threshold can co-exist with the other. For example an agent can be happy about something, angry towards something and bored at the same.

```
findShyRoute(){
        Array of Neighbour Points = Get all Neighbour Points from CurrentPoint;
        Float goalDistance = Get distance between the goal and the agents position;
        Float secondClosest = 999.0;
        Float neighbourDistance = 0.0;
        Int number of agents = 999999999;
        Set currentPoint as the previousPoint;
        Float currentDistance = Get the distance between the goal and the currentPoint;
        For every Neighbour Point in the array{
                Int count = get the number of agents between each the current point and neighbour point;
                neighbourDistance =  get the distance between the goal and the neighbour point;
                If goalDistance is less than 15.0 and the goalBlock equals false{
                        Set goal as targetPosition;
                }
                Else if neighbourDistance less than currentDistance and blockPoints does not contain Neighbour
                Point{
                        If count is less than numAgents{
                                Set count to the numAgents;
                                Set neighbour point to the currentPoint;
                                Set current point to the targetPoint;
                                Set the goalBlock as false;
                        }
                }
                Else if neighbourDistance is greater than currentDistance and blockPoints does not contain
                Neighbour Point{
                        If neighbourDistance is less than secondClosest{
                                Set neighbourDistance as the secondClosest;
                                If count is less than numAgents{
                                        Set count to the numAgents;
                                        Set neighbour point to the currentPoint;
                                        Set current point to the targetPoint;
                                        Set the goalBlock as false;
                                }
                        }
                }
                Else if blockedPoints does contain the Neighbour Point{
                        If neighbourDistance is less than currentDistance and blockedFrom is not equal to
                        previousPoint{
                                If count is less than numAgents{
                                        Set count to the numAgents;
                                        Set neighbour point to the currentPoint;
                                        Set current point to the targetPoint;
                                        Set the goalBlock as false;
                                }
                        }
                        Else if neighbourDistance is greater than currentDistance and blockedFrom is not equal
                        to previousPoint{
                                If neighbourDistance is less than secondClosest{
                                        Set neighbourDistance as the secondClosest;
                                        If count is less than numAgents{
                                                Set count to the numAgents;
                                                Set neighbour point to the currentPoint;
                                                Set current point to the targetPoint;
                                                Set the goalBlock as false;
                                        }
                                }
                        }
                }
        }
}
```

Figure 4. Path finding for shy personality's by finding the route with the least number of agents.

Table 3. The emotional values and the corresponding emotional thresholds.

Values	High Threshold	Low Threshold
happySad	Happy	Sad
stressedRelaxed	Stressed	Relaxed
angryCalm	Angry	Calm
excitedBored	Excited	Bored

In order for the values to increase or decrease and reach a threshold, multiple situations were created to affect the values. These are danger, time pressure, obstacle interference, points of interest and interaction with other agents. Dangerous situations are things that can threaten an agent's life such as natural disasters (fire, floods, earthquakes, etc.). Danger affects the values by increasing stressedRelaxed and angryCalm value and decreasing the happySad value. Time pressure is the attempt to reach a goal by a particular time frame. If an agent doesn't reach their goal within a certain time frame the stressedRelaxed value is increased and the time frame starts again (see Figure 5).

```
timePressure(){
        add deltaTime to timer;
        float minutes = divide timer by 60 to get number of minutes;
        if minutes is greater than 1 minute{
                increase stressedRelaxed value by 5;
                reset timer to 0;
                timer = 0.0f;
                check if the stressedRelaxed value has reached one of the
                thresholds;
        }
}
```

Figure 5. An emotional threshold affected by time pressure.

Obstacle interference is when an obstacle or barrier blocks an agent from taking that path to their goal forcing an agent to go back. When an agent is forced back from an obstacle interfering their angryCalm and stressedRelaxed value is increased bring them closer to their angry and stressed threshold. Points of interest is something that goal or object that is of high interest to an agent. In this project an agent point of interest is their goals. When an agent is interested in the goal they are moving towards their excitedBored value is increase if not it is decreased making them closer to the bored threshold. Agent interactions are when agents who are already reached an emotional threshold effect other agents around them. They affect the other agents by slowly increasing their values of that particular threshold. In large crowds agents can affect each other in large doses forcing a single agent to be affected by multiple agents at a single time (see equation 5).

```
agentInteraction(){
        List of other agents = get list of all agents in the environment;
        For each other agent in the list{
                Float range = get distance between the two agents;
                If the range is less than 5{
                        If one of the emotional values is greater than its corresponding emotional threshold {
                                Increase emotion type timer by deltaTime;
                                Float seconds = get number of seconds by timer % 60;
                                If seconds is greater than 30{
                                        Increase the emotion value of this agent;
                                        Reset timer to 0;
                                }
                        }
                        Else if one of the emotional values is less than its corresponding emotional threshold {
                                Increase emotion type timer by deltaTime;
                                Float seconds = get number of seconds by timer % 60;
                                If seconds is greater than 30{
                                        decrease the emotion value of this agent;
                                        Reset timer to 0;
                                }
                        }

                }
        }
}
```

Figure 6. An emotional threshold affected by agent to agent interaction.

When a threshold is reached an agent parameters are changed to show that emotion. But not all parameters are affected at the same time (see table 3). For example when bored it alters the planning horizon, planning obstacle horizon and speed while stressed affects all parameters. This show how strong certain emotions are to others. When a value goes back under a threshold level it undoes it changes to each parameter that was affect by it.

Table 4. How the agent parameters are affected by each emotional threshold.

	Happy	Sad	Stressed	Relaxed	Angry	Calm	Excited	Bored
Neighbour Distance	Decreased	Increased	Decrease	Decreased	Increased	Decreased	N/A	N/A
Max Number of Neighbours	Increased	Decreased	Increased	Increase	Decreased	Increased	Increased	N/A
Planning Horizon	N/A	Increased	Increased	Decreased	Increased	Decreased	Increased	Decreased
Planning Obstacle Horizon	N/A	Increased	Increased	Decreased	Increased	Decreased	Increased	Decreased
Radius	Decreased	Increased	Increased	Decreased	Increased	Decreased	N/A	N/A
Speed	N/A	Decreased	Increased	Decreased	Increased	N/A	Increased	Decreased

To make each agent emotions unique the thresholds are randomly generated based a min to max based on an agents personality. Different personalities each have different levels in which a person can tolerate and control their emotions (see table 4). For example an aggressive personality cannot control their anger compared to a shy personality and an impulsive agent can get excited quicker than an aggressive personality.

Table 5. The min and max for each emotional threshold for each personality.

	Happy	Sad	Stressed	Relaxed	Angry	Calm	Excited	Bored
Aggressive	70-80	20-30	60-70	20-30	60-70	20-30	70-80	20-30
Shy	70-80	40-50	60-70	30-40	70-80	30-40	70-80	20-30
Impulsive	60-70	20-30	60-70	20-30	70-80	20-30	60-70	20-30
No Personality	70-100	0-40	70-100	0-40	70-100	0-40	70-100	0-40

GROUP FORMATION AND COMMUNICATIONS IN CROWD SIMULATIONS

Group formations can be considered in two ways, firstly agents moving in different positions to display specific shapes and secondly the movement of a group with a leader agent directing the other members of the group and follower agents who follow the directions of the leader agent. [24] defines group by a leader and members (See Figure 7). The leader always moves first and leaves three advised tiles for the members to follow. The members would then move one by one into advised tile. A member then leaves their own advised tiles behind them to allow other members behind to follow. The problem with this is that leader can move into a position in which a member cannot reach an advised tile.

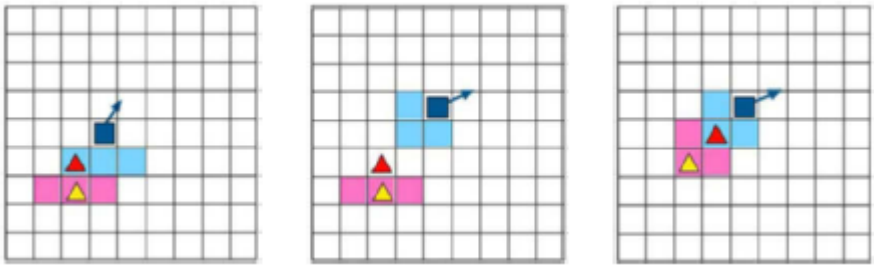

Figure 7. Leader (Square) and member's (Triangle).

Leader moves first and leaves advised tiles. Members then move into the leaders advised tiles. [25] represent group formation as a group of agents forming specific shapes. They adopt the approach that "describes emergent behaviour of agents to occupy the space corresponding to the desired shape". The shapes tested were the letters V and H. They do this by using markers in the space to represent the desired shape. The environment provides markers to allow the agent to move. Users can then spray the

marker to represent the shape they desire. The agents then move towards these markers and position themselves in the shape wanted. [26] provides "an approach to group formations that considers simulated autonomous robots". They use global behaviours to maintain the group's formation without having the behaviours place into the agents. [26] focused on queue, inverted-V and rectangle formations (See Figure 8). They also use the leader/follower in their group formations.

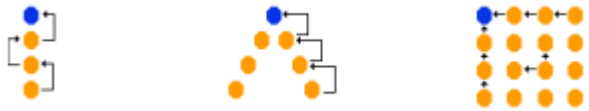

Figure 8. Group formation: queue, inverted-V and rectangle.

The leader and followers use the global behaviours to work together. They are:

- Reaching a target position – An agent tries to reach a set position, the agent will speed up to get there as quick as possible as long as there are no other global behaviours interfering with the agent.
- Reference neighbour following – Agents following a path of their reference neighbours while keeping a fixed angle and distance. This behaviour cannot be used by the leader because it does not have any references instead it is given a path to follow.
- Limited passivity – Determines the minimum movement distance of a reference agent must move ahead by before a follower can react and follow it.
- Waiting for the follower – Forces an agent to reduce its velocity when a follower agent goes beyond a maximum separation distance.
- Priority respect – Causes agents that are going to cross paths and collide avoid each other by reducing the speed of an agent.

The formations used [26] are queue, inverted V and rectangle. The queue formation is when group agents follow a leader in a line allowing all agents to pass through same positions. An inverted V formation is when all agents are positioned in the shape of a V. The leader agent is positioned at the centre point of the V. When the leader moves left or right the other agents placed in the V formation attempt realign back into the V shape. The rectangle formation positions the leader in the corner of the rectangle. When the leader move the follower agents adjust in a realistic manner to get back into the rectangle shape. All these group formations that display leader and followers all have the same problem. The leader is always in front of the members which is not realistic as some leaders can be in the centre of a group or at the back. For example a group of soldiers wandering an area, the leader can be at back giving orders while a few members can be upfront scouting ahead for the group.

Group communication and propagation discusses how agents affect other agent's parameters. One example is stress, [27] demonstrates a technique that can simulate dynamic patterns of crowd behaviours using stress modelling. [27] forces agent's to get stress from different types of factors such as danger, time pressure, areas in the environment and even from other agents. They run multiple types of test each forcing an agent to get stressed based on one or more of these stress types. It was found that agents caused each other stress when trying to evacuate an office environment when a fire broke out. It was also found that stress was communicated from one agent to another when an agent was being chased.

[28] displays communication between agent's through social interaction. [28] is a communication model used to simulate different behaviours in a riot. [28] uses IMCrowd, which is a multi-agent system that allows agent's to have local awareness and autonomous abilities to improvise their actions. The IMCrowd has a communication model that allows agents to socially interact with each other and decide what actions they should take. They test their model with two groups of agents confronting each other. The two groups would become violent and start assaulting each other. It shows when one group surrounded the opposite group the agents surrounded became suppressed, felt inferior to the others and some agents became victims.

The development of groups is very important in creating crowd simulations. Most groups developed in crowd simulation have been created with a leader who directs the rest of the group around the environment. The problem is that the leader is always in front of the group which can be seen as unrealistic as leaders can direct a group from the front, back or middle of a group. Groups [24, 26, 28-30] that require no leader when moving around an environment is not implemented often. When groups with no leader [25, 31] have been implemented they are either developed for group formations in which the goal is to try and keep in position individually with no group decision making or to walk in two large crowds of individual in which the goal is to pass the other group.

This project has been developed with groups that have a leader and groups without a leader. In order for a group dynamic to work a local path finding was used that uses a point to point network in which the agents move around the environment moving from point to another point that has some connection to it; in this case it is called a neighbouring point. At the start of a simulation the agents move to the closest point from where they were spawned, from there on they rely on their personality to determine the next point until they are close enough to the goal. Three personalities were used in this project and they were

aggressive, shy and impulsive. When it comes to a group with a leader, the leader find's the closest point from their position and determines the next points they should take to get to their goal. The group with a leader is developed with the leader deciding what path the group should take to get to their goal. The leader has the ability to be at the front of the group, at the back of the group or a free reign ability in which the leader can be anywhere within the group (See Figure 9). This is done by making the members of the group focus on a particular target while the leaders target is always a point or the goal in all three situations. When the leader is at the front of the group the other member's target is the leader which forces them to stay behind the leader this is also done by decreasing their speed so that they don't get ahead of the leader. When the leader is at the back of the group the other member's target is a point or the goal but in order to keep the leader behind them the leaders speed is reduce so that the leader is slower than the others. When the leader has free reign to go anywhere in the group the members target is a point or the goal.

```
For each member of the group{
        If the group member is the leader{
                Set the leader target as the point position;
        }
        Else if the leader is positioned at the front{
                Decrease the speed of the group member;
                Set the members target to the leaders position;
        }
        Else if the leader is positioned at the back{
                Decrease the speed of the leader;
                Set the members target to the point;
        }
        Else{
                Set the members target to the point;
        }
}
```

Figure 9. The leader has the ability to be at the front, back or have free reign within the group.

The group with no leader determines their path as a group by collaborating with each picking a path and sharing it with the group. This is done by getting each agent in the group to find the best path to the goal by selecting a point that suits the agent's personality. Then each agent of the group goes through each selected path and see's if it is better than their own based on their personality. If an agent finds that another agent in the group has a better path the agent places a vote on that path and on their own path in order have their path select. If they don't they select their own path and place a vote on it. Once each agent has voted the results are checked and the path with the highest vote is selected.

CULTURE IN CROWD SIMULATIONS

Culture is a very important aspect into human behaviour and communications but has never been incorporated into crowd simulations. One possible reason for culture not being incorporated in to crowd simulation is due to its interpersonal complexity [32]. Culture can be incorporated into crowd simulation by using Hofstede's dimensions of cultural. By using power distance, long term vs. short term and individualism vs. collectivism it can help improve crowd simulations realism.

Power distance is used to determine an agent's goal. Each goal has a minimum and a maximum power level. Agents are given a random power level from 0 to 100. An agent goes through each goal and compares its power level to the goals power. If an agents power level is between a goals min to max power the agent places a vote of one to the goal. If there is no goal that matches the power level of an agent, they then search for the closest goal to their power level and store the difference in power and the chosen goal in a secondary vote (See Figure 10). At the end of viewing all goals if the secondary vote has been used it adds one vote to the main vote of the selected goal. Otherwise it moves on to the counting the votes to see if there has been a single vote on any of the goals, if there has it becomes the chosen goal of the agent.

When agents are within a group the power distance is also used to determine a goal but for the whole group (See Figure 11). Each agent in the group goes through each goal and compares the power level of each goal to theirs. If an agents power level is between the goals min to max power then the agents places a vote on that goal. If there is no goal that matches the power level of that agent, the agent then searches for the closest goal to their power level and store the difference in power and the chosen goal in a secondary vote. At the end of viewing all goals if the secondary vote has been used it adds one vote to the selected goal in main vote. Once all agents have voted, those votes are tallied up and the goal with the highest number of votes is selected as the group's goal to achieve.

Power distance is also used within a group to determine the path the agents should take. This is done by adding an extra vote in the group's collaboration to the person who has the highest power level within the group. This shows that agents with a higher power level will have more control over a group than agents with lower power levels.

Long term vs. short term has been used to determine how long a single agent or a group of agents will spend at a single goal. If

an agent is long term it means that the agent will spend a small amount of time at the goal before heading to the next one. While a short term agents will spend a large amount of time at one goal before moving on to the next.

```
setGoalIndividuals(agent, list of goals){
        list of goalVotes;
        int difference = 1000;
        int voteChoice = 99;
        int chosenGoal = 0;
        int highestVote = 0;
        For each goal in the list{
                add a vote of 0 to each goalVote;
        }
        For each goal in the list{
                If the goal max power distance is less than the agents power distance && the goal min power distance is greater than
                the agents power distance{
                        Add 1 to goalVote;
                }
                Else{
                        If the goal min power distance is greater than the agents power distance{
                                Int powerDifference = goal min power distance – agents power distance;
                                If powerDifference is less than difference {
                                        Set powerDifference as difference;
                                        Set the goal number to voteChoice;
                                }
                        }
                        Else if the goal max power distance is less than the agents power distance{
                                Int powerDifference = agents power distance – goal max power distance;
                                If powerDifference is less than difference {
                                        Set powerDifference as difference;
                                        Set the goal number to voteChoice;
                                }
                        }
                }
        }
        If the voteChoice is not equal to 99{
                Add 1 vote to goalVote[voteChoice];
        }
        For each goalVote{
                If the goalVote[i] is greater than the highestVote{
                        Set i to chosenGoal;
                        Set the goalVote[i] to the highestVote;
                }
        }
        Return the goal[chosenGoal];
}
```

Figure 10. How an individual agent selects a goal using power distance.

```
setGoalGroups(groupMembers, list of goals){
        list of goalVotes;
        int difference = 1000;
        int voteChoice = 99;
        int chosenGoal = 0;
        int highestVote = 0;
        For each goal in the list{
                add a vote of 0 to each goalVote;
        }
        For each groupMember{
                For each goal in the list{
                        If the goal max power distance is less than the groupMember[i] power distance && the goal min power distance is
                        greater than the groupMember[i] power distance{
                                Add 1 to goalVote[j];
                        }
                        Else{
                                If the goal min power distance is greater than the groupMember[i] power distance{
                                        Int powerDifference = goal min power distance – groupMember[i] power distance;
                                        If powerDifference is less than difference {
                                                Set powerDifference as difference;
                                                Set the goal number to voteChoice;
                                        }
                                }
                                Else if the goal max power distance is less than the groupMember[i] power distance{
                                        Int powerDifference = groupMember[i] power distance – goal max power distance;
                                        If powerDifference is less than difference {
                                                Set powerDifference as difference;
                                                Set the goal number to voteChoice;
                                        }
                                }
                        }
                }
        }
        If the voteChoice is not equal to 99{
                Add 1 vote to goalVote[voteChoice];
        }
        For each goalVote{
                If the goalVote[i] is greater than the highestVote{
                        Set i to chosenGoal;
                        Set the goalVote[i] to the highestVote;
                }
        }
        Return the goal[chosenGoal];
}
```

Figure 11. How a group of agent select a goal using power distance.

In a group with a leader directing them the leader determines how long the group will wait at each goal (See Figure 12). In a group that has no leader the amount of time spent at each goal is determined by how many long and short term agents are within the group. For example if there are five agents in a group and three of them are short term and two are long term then the group will spend the amount of time of a short term agent at each goal.

```
longTermVsShorTerm(agent){
        If the agent is a long term{
                return 10.0;
        }
        Else{
                return 30.0;
        }
}
```

Figure 12. Check if an agent is long term or short term.

Individualism vs. Collectivism focuses on the group dynamic by determining whether an agent is able to accept the group decision on which path should be taken. Each agent picks a path, if that path is chosen by the group that agent is unaffected by this dimension of culture as it only affects an agent who chose a different path. This is because an agent who did pick that path has already accepted the decision while agents who didn't pick that path may not have accepted it. Collectivism agent tends to accept a group's decision as they think of the group over one's self. While individualism agents don't accept group decision's willingly as they do think of one's self over the group.

CONCLUSION

Emotional threshold method provided a unique way of implementing agent's emotions. Even though these tests revealed only half of the emotions that were implemented it still provide reasonable data in which different situations with in an environment such as danger, time pressure and agent interaction can affect agent's emotions effectively. In order to improve these methods future developments such as adding more personalities to the path planning and incorporating the ability to learn and memorise the path they have taken. To improve the emotional threshold more situations that can affect an agents emotions within an environment need to be implemented. To make agents appear more realistic, actions can be added so that when an emotional threshold is reached the agent can perform a particular action corresponding to the emotion.

Similar, the proposed framework integrated culture in crowd simulations. In order to improve these methods future developments must be considered. For culture more research into other cultural dimensions should be considered and improvement into the current cultural dimensions that have been implemented by adding more situations that can effect each one. The group's method can be improved by having the ability to leave or join a group when they reach a certain goal. The group's method can be improved by having the ability to change leaders based on a situation like group disagreements, etc.

REFERENCES

[1] N. Pelechano, J. M. Allbeck, and N. I. Badler (2008) *Virtual crowds: Methods, simulation, and control*.

[2] J. S. Milazzo, N. M. Rouphail, J. E. Hummer, and D. P. Allen (1998) 'Effect of pedestrians on capacity of signalized intersections,' *Transportation Research Record: Journal of the Transportation Research Board,* vol. 1646, pp. 37-46.

[3] S. P. Hoogendoorn and P. H. Bovy (2005) 'Pedestrian travel behavior modeling,' *Networks and Spatial Economics,* vol. 5, pp. 193-216.

[4] G. G. Løvås (1994) 'Modeling and simulation of pedestrian traffic flow,' *Transportation Research Part B: Methodological,* vol. 28, pp. 429-443.

[5] B. Piccoli and A. Tosin (2011) 'Time-evolving measures and macroscopic modeling of pedestrian flow,' *Archive for Rational Mechanics and Analysis,* vol. 199, pp. 707-738.

[6] S. Hoogendoorn and P. H. Bovy (2000) 'Gas-kinetic modeling and simulation of pedestrian flows,' *Transportation Research Record: Journal of the Transportation Research Board,* vol. 1710, pp. 28-36.

[7] C. W. Reynolds (1987) 'Flocks, herds and schools: A distributed behavioral model,' *ACM SIGGRAPH Computer Graphics,* vol. 21, pp. 25-34.

[8] A. Varas, M. Cornejo, D. Mainemer, B. Toledo, J. Rogan, V. Munoz, *et al*. (2007) 'Cellular automaton model for evacuation process with obstacles,' *Physica A: Statistical Mechanics and its Applications,* vol. 382, pp. 631-642.

[9] C. Burstedde, K. Klauck, A. Schadschneider, and J. Zittartz (2001) 'Simulation of pedestrian dynamics using a two-dimensional cellular automaton,' *Physica A: Statistical Mechanics and its Applications,* vol. 295, pp. 507-525.

[10] T. I. Lakoba, D. J. Kaup, and N. M. Finkelstein (2005) 'Modifications of the Helbing-Molnar-Farkas-Vicsek social force model for pedestrian evolution,' *Simulation,* vol. 81, pp. 339-352.

[11] D. Helbing and P. Molnar (1995) 'Social force model for pedestrian dynamics,' *Physical review E,* vol. 51, p. 4282.

[12] D. Thalmann (2012) *Crowd Simulation*. London, UK: Springer- Verlag London Ltd.
[13] D. C. Brogan and J. K. Hodgins (1997) 'Group behaviors for systems with significant dynamics,' *Autonomous Robots*, vol. 4, pp. 137-153.
[14] H.-Q. Chong, A.-H. Tan, and G.-W. Ng (2007) 'Integrated cognitive architectures: a survey,' *Artificial Intelligence Review*, vol. 28, pp. 103-130.
[15] G. Le Bon (1897) *The crowd: A study of the popular mind*: Macmillian.
[16] R. H. Turner and L. M. Killian (1957) 'Collective behavior,'
[17] N. E. Miller and J. Dollard (1941) 'Social learning and imitation,'
[18] M. E. Bratman, D. J. Israel, and M. E. Pollack (1988) 'Plans and resource-bounded practical reasoning,' *Computational intelligence*, vol. 4, pp. 349-355.
[19] D. C. Dennett (1989) *The intentional stance*: MIT press.
[20] J. E. Laird, A. Newell, and P. S. Rosenbloom (1987) 'Soar: An architecture for general intelligence,' *Artificial intelligence*, vol. 33, pp. 1-64.
[21] J. R. Anderson, D. Bothell, M. D. Byrne, S. Douglass, C. Lebiere, and Y. Qin (2004) 'An integrated theory of the mind,' *Psychological review*, vol. 111, p. 1036.
[22] S. J. Guy, S. Kim, M. C. Lin, and D. Manocha (2011) 'Simulating heterogeneous crowd behaviors using personality trait theory', in *Proceedings of the 2011 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pp. 43-52.
[23] I. Stamatopoulou, I. Sakellariou, and P. Kefalas (2012) 'Formal Agent-Based Modelling and Simulation of Crowd Behaviour in Emergency Evacuation Plans', *in Tools with Artificial Intelligence (ICTAI), 2012 IEEE 24th International Conference on*, pp. 1133-1138.
[24] C. Loscos, D. Marchal, and A. Meyer (2003) 'Intuitive crowd behavior in dense urban environments using local laws', *in Theory and Practice of Computer Graphics, 2003. Proceedings*, pp. 122-129.
[25] R. A. Rodrigues, A. de Lima Bicho, M. Paravisi, C. R. Jung, L. P. Magalhaes, and S. R. Musse (2010) 'An interactive model for steering behaviors of groups of characters,' *Applied Artificial Intelligence*, vol. 24, pp. 594-616.
[26] Lopez-Sanchez, Maite Cerquides, and Jesus (2006) 'Formation Maintenance for Autonomous Robots by Steering Behavior Parameterization,'
[27] S. Kim, S. J. Guy, D. Manocha, and M. C. Lin (2012) 'Interactive simulation of dynamic crowd behaviors using general adaptation syndrome theory', *in Proceedings of the ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games*, pp. 55-62.
[28] W.-M. Chao and T.-Y. Li (2011) 'Simulating riot for virtual crowds with a social communication model', *in Computational Collective Intelligence. Technologies and Applications*, ed: Springer, pp. 419-427.
[29] O. B. Bayazıt, J.-M. Lien, and N. M. Amato (2005) 'Swarming behavior using probabilistic roadmap techniques', *in Swarm Robotics*, ed: Springer, pp. 112-125.
[30] Q. Nguyen, F. D. McKenzie, and M. D. Petty, (2005) 'Crowd behavior cognitive model architecture design', *in Proceedings of the 2005 conference on behavior representation in modeling and simulation (BRIMS)*, pp. 55-64.
[31] C. Turkay, E. Koc, and S. Balcisoy (2011) 'Integrating information theory in agent-based crowd simulation behavior models,' *The Computer Journal*, vol. 54, pp. 1810-1820.
[32] N. Pelechano, J. M. Allbeck, and N. I. Badler (2008) 'Virtual crowds: Methods, simulation, and control,' *Synthesis Lectures on Computer Graphics and Animation*, vol. 3, pp. 1-176.