Project Report

on

# Mixed Crowd Simulation in Virtual Environment

Under the guidance of Dr. Rahul Kala


Indian Institute of Information Technology, Allahabad

August – December 2016

Submitted By –

Sahil Agarwal (IIT2014120)

Abhishek Deora (IIT2014141)

Apoorv Dwivedi (IIT2014142)

Vipul Agarwal (IIT2014146)

Parshant Singh (IIT2014157)

# Candidates Declaration

We hereby declare that the work presented in this project report entitled “Mixed Crowd Simulation in a Virtual Environment”, submitted end-semester report of 5th Semester report of B.Tech. (IT) at Indian Institute of Information Technology, Allahabad, is an authenticated record of our original work carried out from August 2016 to December 2016 under the guidance of Dr. Rahul Kala. Due acknowledgements have been made in the text to all other materials used. The project was done in full compliance with the requirements and constraints of the prescribed curriculum.

Place: Allahabad

Date: 18-09-2016

Sahil Agarwal (IIT2014120)

Abhishek Deora (IIT2014141)

Apoorv Dwivedi (IIT2014142)

Vipul Agarwal (IIT2014146)

Parshant Singh (IIT2014157)

# Certificate

This is to certify that the above statement made by the candidates is correct to the best of my knowledge.

Place: Allahabad

Date: 18-09-2016

Dr. Rahul Kala

# Abstract

Mixed Crowd Simulation is currently one of the most important and heavily researched topics today because of its major application in the robotics and architecture industry. With heavy research going on making robots more suitable for daily needs, Mixed Crowd Simulation determines how the robots will interact with a general human crowd while maintain their desired role.

We implemented the Social Force Model along with Probabilistic RoadMap to get desired path for an individual and Finite State Machine to maintain sub-goals and desires of the individual. Our simulation works with a $O(N^2)$ complexity while maintaining the desired speed of all individuals. It also makes sure that the distance travelled by an individual does not increase by a huge factor as the environment gains more agents which is a general human tendency.

# Table of Contents

# Introduction

## Problem Statement

To design a system able to simulate a crowd of pedestrians interacting in the given scenario conditions. We present an agent-based model with the aim of achieving the following objectives:

- Simulation of scenario based crowds in normal circumstances.
- Providing a unique model that uses a simple and fast system and supports the concept of sub-goals and goals unlike existing crowd simulations with one single goal for all the crowd members.

## Background

Creating crowds involves the creation of number of individuals exhibiting a range of actions. Typically, an animator creates a library of motions for each character which a very time consuming task. The complexity of this task increases exponentially if the crowd has a high density. Crowd simulation is the procedural animation of large groups of individuals with simple automation of the motion of each individual. The crowd density can easily be increased without a drastic increase in the time consumption as compared to simple animations.

Existing crowd systems simulations are used to simulate in the major areas like entertainment, urban modelling, safety or some other emergency situations. In general, these applications require a simulation of the crowd in a typical circumstances or certain situations like battles or certain case of emergencies like fire escape etc.

We wish to simulate the crowds under a non-emergency situation i.e. how a crowd will behave in a normal situation, with the aim of achieving normal crowd simulation with large number of individuals. We also wish to add the feature of agents having multiple goals (and hence tasks) so that the simulation is closer to the real world.

## Overview

The Simulation of a large number of independent agents acting and moving through a shared space relies on the solution to many sub-problems: determining what an agent wants to do, how it will achieve its purpose and how it responds to unforeseen challenges. A full crowd simulator can be regarded as the union of solutions to each of these sub-problems.

Each of these problems typically admits various solutions. For example, the problem of determining how an agent reaches its goal can be mapped to global motion planning. To solve this, one could use different algorithms like road maps, navigation meshes etc. Selecting one depends on the developers and is a non-trivial choice since nearly all of them have similar time complexities.

We have implemented the simulation in three basic steps. First, a goal is selected for each agent depending upon its finite state machine. Then a base plan to or a path is generated to reach that goal. Lastly, the plan is adapted to local and dynamic conditions and the agent moves towards the goal. Each sub-problem can make queries into the environment to support its computation hence reducing complexity of program design and hence the overall complexity of this project.

# Literature Review

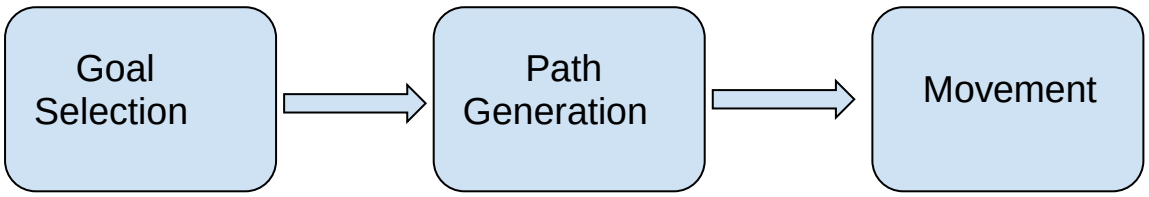
Crowd simulation is very important aspect of the field of computer graphics and robotics. In the thesis “Crowd simulation of pedestrians in a virtual city” published by Flora Ponjou Tasse et al. [1], they proposed an agent-based model for simulating behaviour of real crowd in virtual cities. Implementation of this crowd simulation include battle situations, emergency situations, etc. The Autor suggest use of Probabilistic RoadMap Model for calculating paths between locations.

One of the major practical uses of robots in social settings is to act as guide for individuals in general activities. This includes informing people when they stray out of path, like when in a commercial centre. In the paper “Cooperative social robots to accompany groups of people” by Anais Garrell et al. [2], they introduced a prediction model that anticipates the position and then tries to help by making people stay in the group. While this model is quite useful, we realized that social force model will be more practical for our scenario.

Robots in social scenarios can become bothersome. They attract attention which can lead to congestion. This can lead to problems in narrow spaces. In the paper “Simulation-Based Behaviour Planning to Prevent Congestion of Pedestrians Around a Robot” by Hiroyuki Kidokoro et al. [2], they have tried to address these problems by making the robot anticipate the pedestrian congestion and avoiding these kinds of situations. The paper also mentioned the use of Social Force for dynamic force calculations to steer the agent away from each other and building and towards the main goal.

# Methodologies

Our main purpose is to propose a crowd simulation system for addressing the problems as stated before. For this we describe a crowd as a large group of many agents where each agent has its own set of goals. The set of goals for any agent is decided by a behavioural finite state machine and each scenario in our simulations has multiple finite state machines with each agent being assigned one of them randomly. Agents should be able to perceive information from their environment and should be able to reach to their respective goals without colliding with other agents, buildings or other static obstacles just like in real life.




## Simulation

**Agent**: To simplify things, we have represented each agent as a spherical body and agents of different FSM's are represented by different colours. For different scenarios, we have chosen different agent radius.

**Static Obstacles**: Our simulations also have static obstacles. Static obstacles are not goals, their main purpose is to provide hindrance in collision avoidance with other agents. They are represented by cyan circles each characterized by their position and radius.

**Building**: Buildings are represented by red circles. Each building is defined by position of its centre and its radius. At a time, building can either act as a sub-goal or as a static obstacle for an agent.

Crowd simulation problem can be mainly into related sub-problems like goal selection, path generation and basic motion (avoiding collisions).

## Goal Selection - Finite State Machine

For the purpose of defining different behaviours of each agent, we have used a Finite State Machine(FSM). Each Automaton has a few states which defines the general tendencies of an agent in a given scenario which in turn govern what goal the agent seeks, how it intends to achieve that goal, and also models agent's fundamental characteristics such as general motion. The transitions from one state to another signifies a change in the agent's behaviour and hence its goal. Since It has been concluded that finite state machines are computationally fast and accurate [3], we used an FSM as a specification for our agent in the given environment. Our FSM works on a few elements such as condition, target and transitions. [4]

**Condition: -** The condition test along with target state is a Boolean type test which determines if a pre-defined target has been achieved. If the current target is satisfied, it invokes the transition function of the FSM which updates the behaviour of our agent. Figure 1 has the pseudo-code for the condition test. Here, we take input as current position of the agent, the FSM and the current behaviour function. If the condition is satisfied, we update the behaviour using fsm.transition.

**Target: -** The target state or behaviour is used to define the current behaviour and hence goal of our agent. We have defined our target as the next goal in terms of location coordinates which as per our scenario signifies the agent's current mood.

**Transitions: -** Transitions or goal-selections is invoked whenever the agent passes the condition test. Target transitions can be random when there are multiple possibilities after the agent reached one of its sub-goal or an automatic return. Lastly, in case of emergencies, target state can be directed to a panic like behaviour for real world simulations. Figure 1 has the pseudo-code for fsm.transition. We assign the next state of the FSM here and return the updated behaviour due to the new state.

```
function conditon-test:
        INPUT:  current-position specifies the co-ordinates of the agent's location
                fsm defines the current Finite State Machine of the agent
                current-behaviour defines the current instrunction-set of the agent
        RETURNS: new instruction set for the agent
        if current-position is at fsm.target then
                updated-behaviour <- fsm.transition()
                return updated-behaviour
        else
                return current-behaviour

function fsm.transition:
        INPUT: None
        RETURNS: new instruction set for the agent
        fsm.target <- next state of the Finite State Machine
        return fsm.target
```

*Figure 1 Pseudo - Code for the Finite State Machine*

## Path Generation

In this segment, we compute a collision free path for satisfying a goal generated according to the goal selection module. A path is generated consisting of a sequence of intermediate goals leading to the main target. For the purpose of path generation in our project we have used probabilistic roadmaps technique [5] [6] [3].

**Probabilistic roadmap**: Figure 2 is the general representation of the Probability RoadMap Model whereas figure 3 has the pseudo code for PRM. In this technique, we generate a graph which consists of randomly chosen points from our environment as nodes. Two nodes in the graph are only connected if they are not very far apart from each other (this distance is fixed) and if there are no static obstacles between them (generatePoints function in figure 3). The weight of each edge is the Euclidean distance between two nodes which it connects. After this, we add our initial position (start position) and the goal as vertices in the graph and make edges between these two and other nodes which satisfy the conditions described before (generateEdges function in figure 3). Once we have got this graph, we apply Dijkstra's algorithm to generate the shortest path between the start and goal positions and store it. Advantage of using PRM is that it is computationally less expensive than most other techniques. Moreover, it provides us with the randomness required to depict an agent's behaviour in real world scenarios to a large extent.

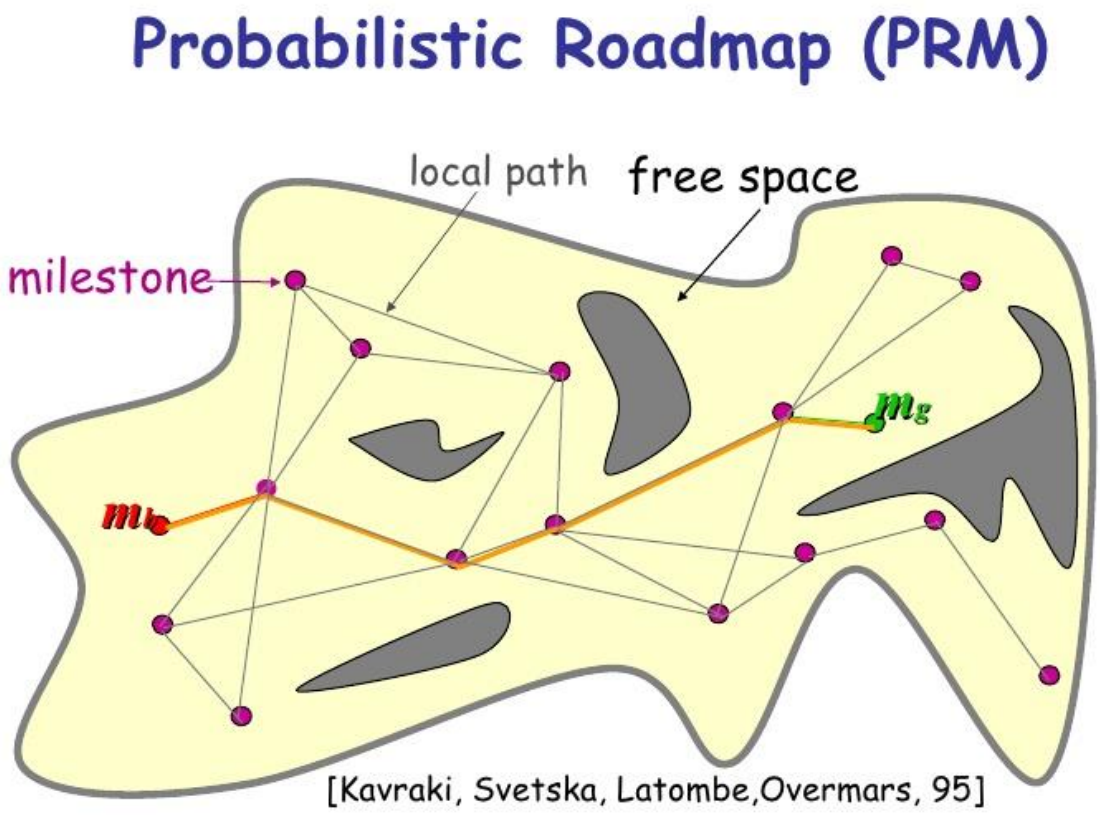



*Figure 2 Probabilistic RoadMap Example*

```
function generatePoints:
    Input: Two integers boundary and numOfPoints list of obstacles
    Output: List of vertices
    point <- stores a random point in euclidean space
    for 1 to numOfPoints :
        while point is not inside an obstacle and point is not in vertices :
            point := generate a random point inside boundary
        end while
        Add point to vertices
    end for


function generateEdges:
    Input: list of vertices and obstacles
    Output: list of edges and weights
    num <- stores num of vertices
    edges <- stores edges
    weights <- stores weights of edges
    for i from 0 to num:
        for j from 0 to num:
            d <- distance(vertex i, vertex j)
            if i != j and d is less than a given distance:
                if edge does not intersect any obstacles:
                    Add vertex j  to edges i
                    weight[(vertex i, vertex j)] <- d
                end if
            end if
        end for
    end for

function prm:
    Inputs: start, goal, boundary, obstacle
    Output: path
    numOfPoints <- stores the no of points to be generated
    vertices <- generatePoints(boundary, numOfPoints, obstacle)
    Add start and goal to vertices
    (edges, weights) <- generateEdges(vertices, obstacle)
    path <- dijkstra(vertices, edges, weights, start)
```

*Figure 3 Pseudo Code for PRM*

## Basic Movement

Navigation of the agent on a generated path is handled by this module with the help of Helbing's social force model [7]. According to this model, the acceleration of an agent is defined by: -

$$\frac{dv_i(t)}{dt} = f_{target} + \sum_{j \neq i} f_{ij} + \sum_{b} fiB$$

$$\frac{dv_i(t)}{dt} \rightarrow Social\ Force$$

$$f_{target} \rightarrow attractive\ force\ of\ the\ intermediate\ goal$$

$$f_{ij} \rightarrow repulsive\ force\ from\ another\ agent$$

$$f_{iB} \rightarrow repulsive\ force\ from\ a\ border\ of\ a\ building$$

*Equation 1 Social Force Model*

Pedestrians in real life want to reach to their goal while avoiding obstacles and collision with other pedestrians. There are many ways to achieve this. Equation 1 uses the model devised by Helbing et al. [2000]. The model which uses attractive and repulsive forces between the agent and its environment to achieve this objective. At every time step of our simulation, each agent experiences an attractive force towards the main goal (Equation 2) which helps it to achieve a desired velocity. It also experiences repulsive forces from buildings, static obstacles (Equation 4) and all the other agents (Equation 3) close to it. We have used circular specification of social force model in our project.

$$\boldsymbol{f_{target}} = \frac{v_i^d\, \boldsymbol{e_i} - \boldsymbol{v_i}}{\tau_i}$$

$$v_i^d \rightarrow the\ desired\ speed\ of\ i$$

$$e_i \rightarrow s\ the\ unit\ vector\ from\ the\ target\ to\ the\ position\ of\ i$$

$$\tau_i \rightarrow constant\ time\ that\ ensures\ that\ agent\ speed\ is\ not\ too\ low\ even\ after\ a\ collision$$

*Equation 2 Attractive force of the intermediate goal*

$$\boldsymbol{f_{ij}} = [A_i \exp\left({r_{ij} - d_{ij}}/{B_i} + kg_{ij}\right)\boldsymbol{n_{ij}} + Kg_{ij}\Delta v_{ji}\boldsymbol{t_{ij}}$$

$A_i \rightarrow Weight\ of\ a\ repulsive\ force\ on\ agent\ i\ (2000)$

$r_{ij} \rightarrow Sum\ of\ their\ radii$

$d_{ij} \rightarrow Distance\ between\ i\ and\ j$

$B_i \rightarrow Fall\ off\ distance\ of\ agent\ i\ (0.5)$

$k \rightarrow Pushing\ Weight\ (120000)$

$$g_{ij} \rightarrow \begin{cases} 0 & d_{ij} > r_{ij} \\ r_{ij} - d_{ij} & otherwise \end{cases}$$

$n_{ij} \rightarrow the\ unit\ vector\ pointing\ from\ j\ to\ i$

$t_{ij} \rightarrow the\ tangential\ direction$

$\Delta v_{ji} \rightarrow a\ change\ in\ velocity\ (v_j - v_i).t_{ij}$

*Equation 3 Repulsive Force from other agents*

$$\boldsymbol{f_{iB}} = [A_i \exp\left({r_i - d_{iB}}/{B_i} + kg_{iW}\right)\boldsymbol{n_{iW}} + Kg_{ij}(\boldsymbol{v_i}.\boldsymbol{t_{iW}})\boldsymbol{t_{iW}}$$

$A_i \rightarrow Weight\ of\ a\ repulsive\ force\ on\ agent\ i\ (2000)$

$r_i \rightarrow Radius\ of\ agent\ i$

$d_{iB} \rightarrow Distance\ between\ agent\ i\ and\ building\ B$

$B_i \rightarrow Fall\ off\ distance\ of\ agent\ i\ (0.5)$

$k \rightarrow Pushing\ Weight\ (120000)$

$$g_{iB} \rightarrow \begin{cases} 0 & d_{iB} > r_i \\ r_i - d_{iB} & otherwise \end{cases}$$

$n_{iW} \rightarrow the\ unit\ vector\ pointing\ from\ Building\ j\ to\ agent\ i$

$t_{iB} \rightarrow the\ tangential\ direction$

$v_i \rightarrow Velocity\ of\ agent\ i$

*Equation 4 Repulsive force from a building or obstacle*

Figure 4 is the generic hierarchical flow chart of our program. We first initialize the environment such as obstacles, buildings, etc. After basic initializations, paths to and from buildings are calculated using Probabilistic Roadmap Model. The next step is to initialize agents, randomly allocate them with respective FSM's for the scenario, update start state location, and provide with a valid list of sub-goals by running the FSM. Lastly, we simulate the environment and if the final goal is reached, we delete that agent from the list.

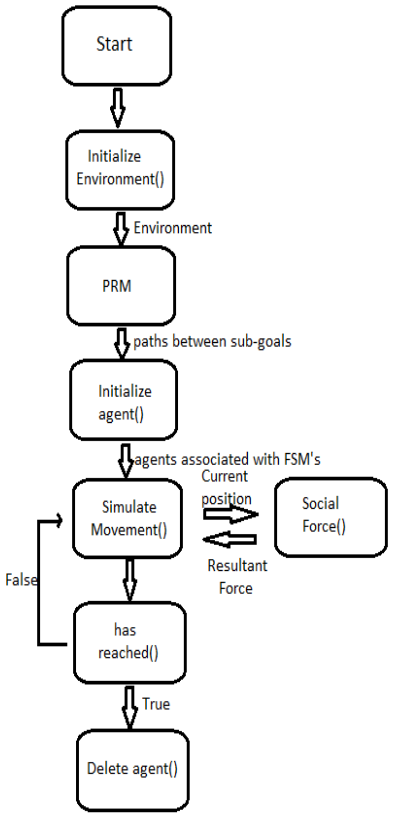



*Figure 4 Program Flow Chart*

## Screenshots and Scenarios

Figure 5 shows scenario 1. We have simulated an Admin Building at the beginning of new session. We have four different types of agent and each type of agent is given a different colour. Each FSM has a few states and a few of these states are optional and the rest are compulsory.

The first type of agent is an old student i.e. who was admitted in a previous year (green). Their FSM has the following states - Entry, Exam-Cell, Council of Wardens' Office, Bank, Accounts-Student, Exit1 and Exit2.

The second type of agent is a new student i.e. who will be joining the institute this year (orange). Their FSM has the following states - Entry, Reception, Exam-Cell, Dean-Affairs, Accounts-Student, Bank, Council of Wardens' Office, Exit1 and Exit2.

The third type of agent is an institute faculty (blue). Their FSM has the following states - Entry, Director, Dean-Academics, Accounts-Priority, Bank, Exit1, Exam-Cell and Exit2.

The last type of agent is a staff member employed by the institute (magenta). Their FSM has the following states - Entry, Reception, Dean-Affairs, Accounts-Priority, Exit1 and Exit2.

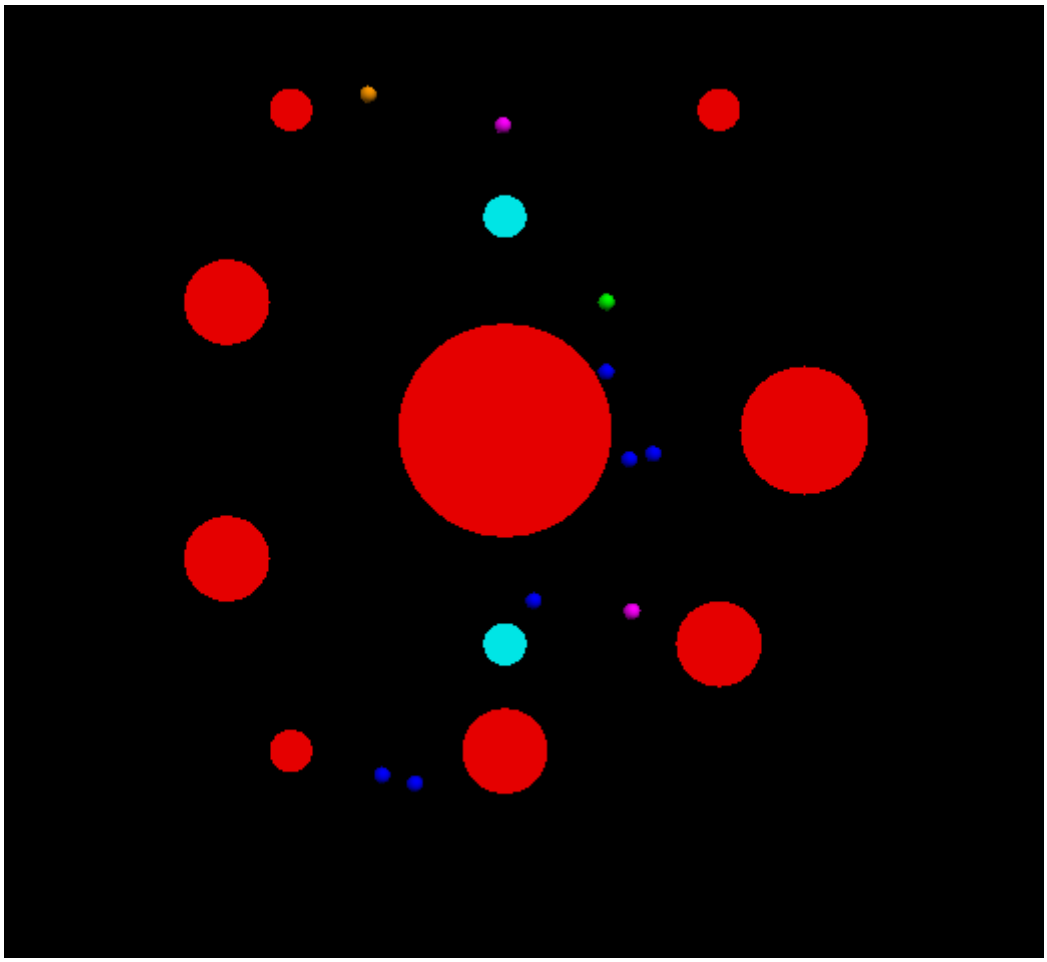


*Figure 5 Scenario 1 - Admin Building during new Session*

Figure 6 shows scenario 2. We have simulated the college campus during a fest. We have two different types of agents and each type of agent is given a different colour. Each FSM has a few states and a few of these states are optional and the rest are compulsory.

The first type of agent is an internal student i.e. who is part of the conducting institute (green). Their FSM has the following states – Pocket-Gate, HQ, Ground and CC3.

The second type of agent is an external student i.e. who is not part of the conducting institute (orange). Their FSM has the following states - Main-Gate, Admin, CC3, Ground and Cafeteria.

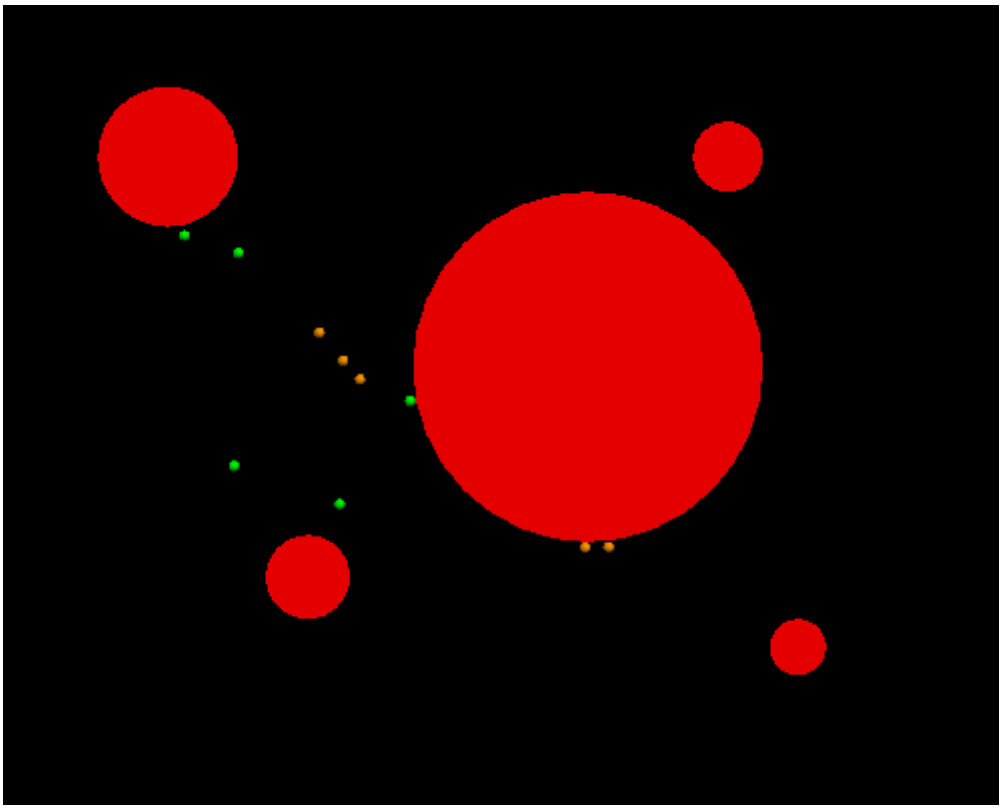


*Figure 6 Scenario 2 - College during a cultural fest*

Figure 7 shows scenario 3. We have simulated a train station. We have four different types of agent and each type of agent is given a different colour. Each FSM has a few states and a few of these states are optional and the rest are compulsory.

The first type of agent is a passenger entering from gate 1 (green). Their FSM has the following states - Entry1, Waiting Room 1, Waiting Room 2, Waiting Room 3, Waiting Room 4, Platform 1, Platform 2 and Platform 3.

The second type of agent is a passenger entering from gate 2 (orange). Their FSM has the following states - Entry1, Waiting Room 1, Waiting Room 2, Waiting Room 3, Waiting Room 4, Platform 1, Platform 2 and Platform 3.

The last type of agent is a passenger exiting from gate 1 (blue). Their FSM has the following states - Platform 1, Platform 2, Platform 3, Waiting Room 1, Waiting Room 2, Waiting Room 3, Waiting Room 4, Exit 1.

The last type of agent is a passenger exiting from gate 2 (magenta). Their FSM has the following states - Platform 1, Platform 2, Platform 3, Waiting Room 1, Waiting Room 2, Waiting Room 3, Waiting Room 4, Exit 2.

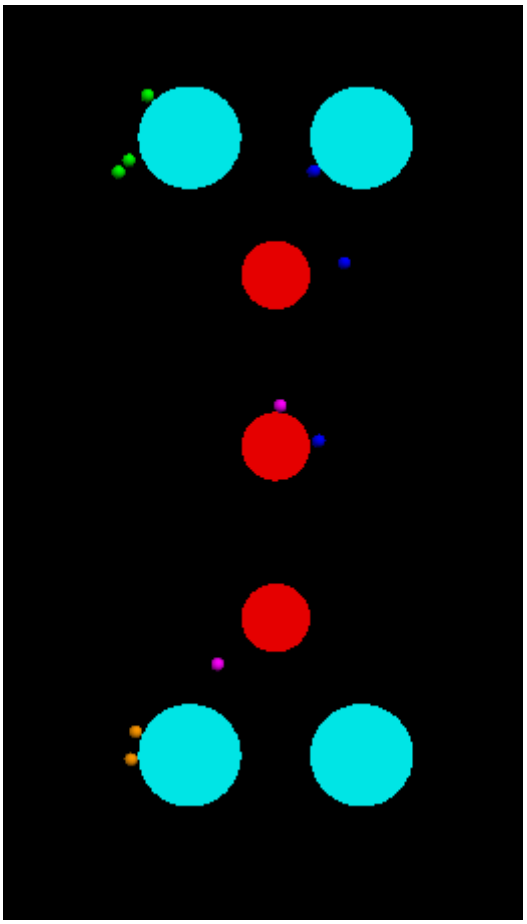
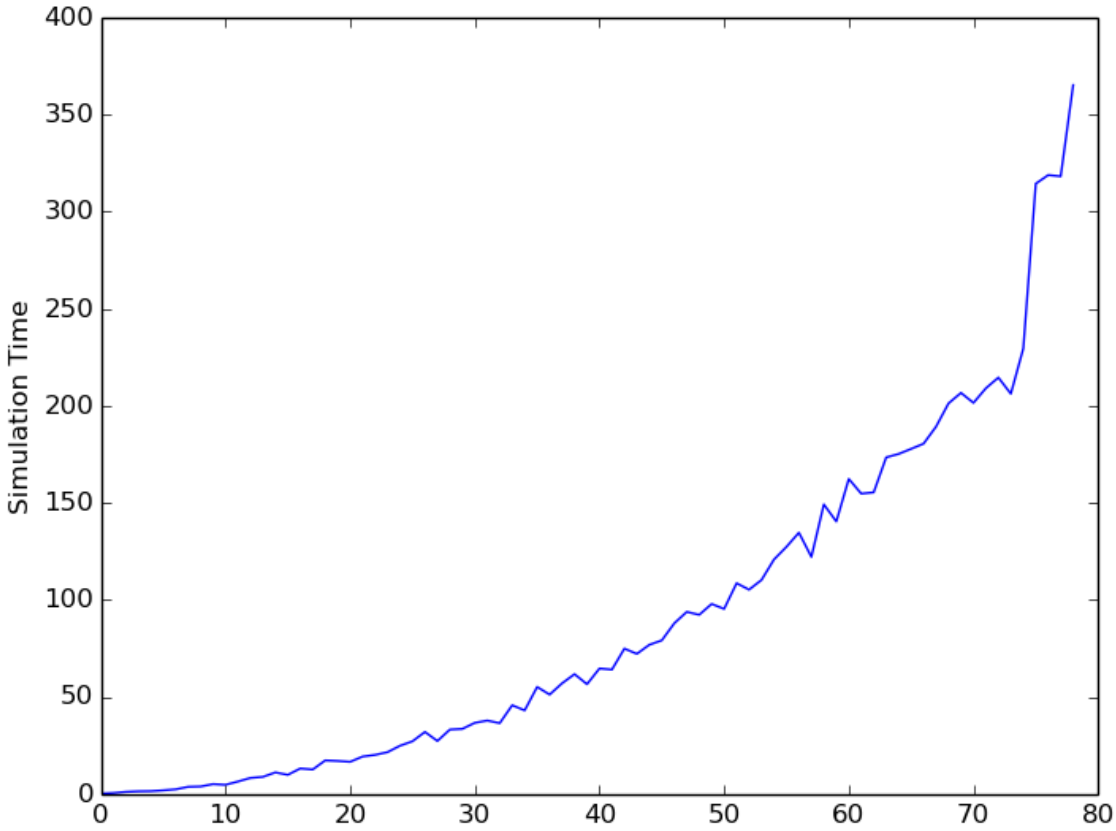


*Figure 7 Scenario 3 - Railway Platform*

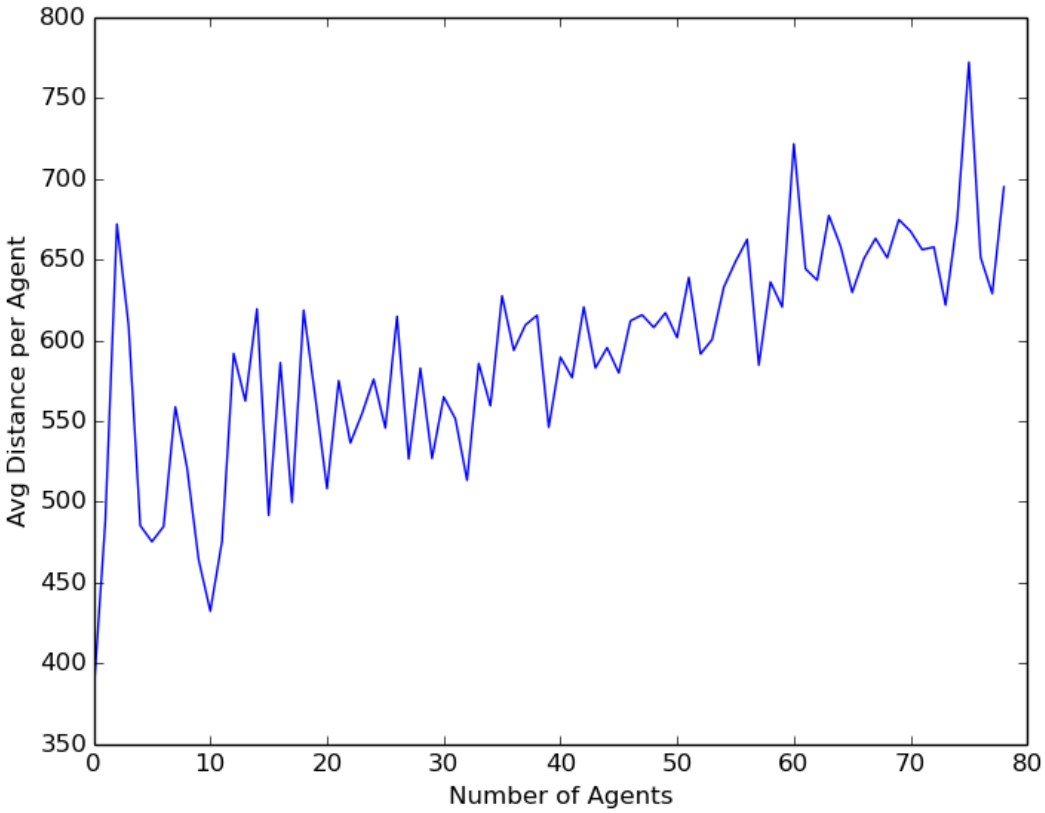
## Results and Analysis




*Graph 1 Simulation Time vs Number of Agents*

Graph 1 is a plot between simulation time represents how long it takes for the number simulation to complete and the number of agents. From the graph, it is observed that the simulation time hold $O(N^2)$ relationship with the number of agents. The simulation time tends to increase with increasing number of agents.

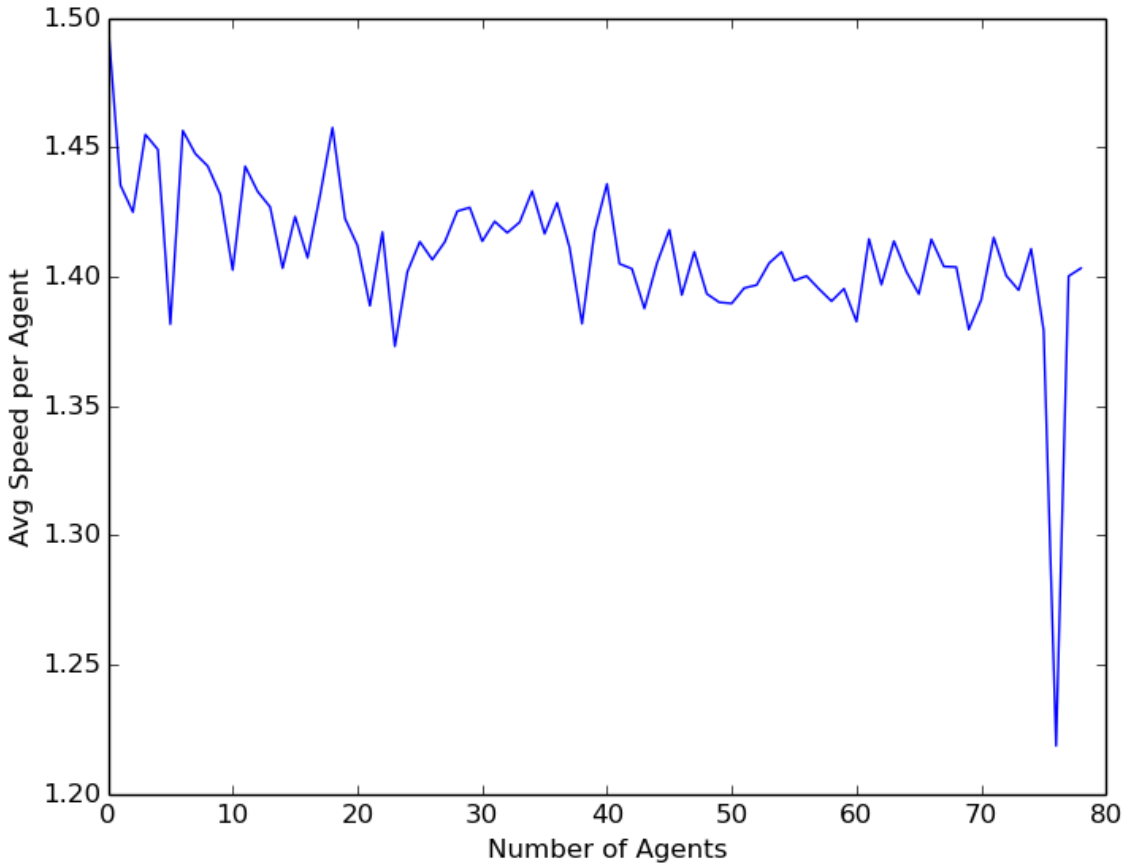
This is a justified result of the project as the social force is sum of force from target, force from **every other agent,** and force due to buildings. Since force from every other agent requires iteration on each of them, we should get an $O(N^2)$ algorithm




*Graph 2 Average Distance travelled per agent vs Number of Agents*

Graph 2 shows a linear relationship between the Number of Agents and Average Distance per Agent. The average distance per agent tends to increase with increasing number of agents.

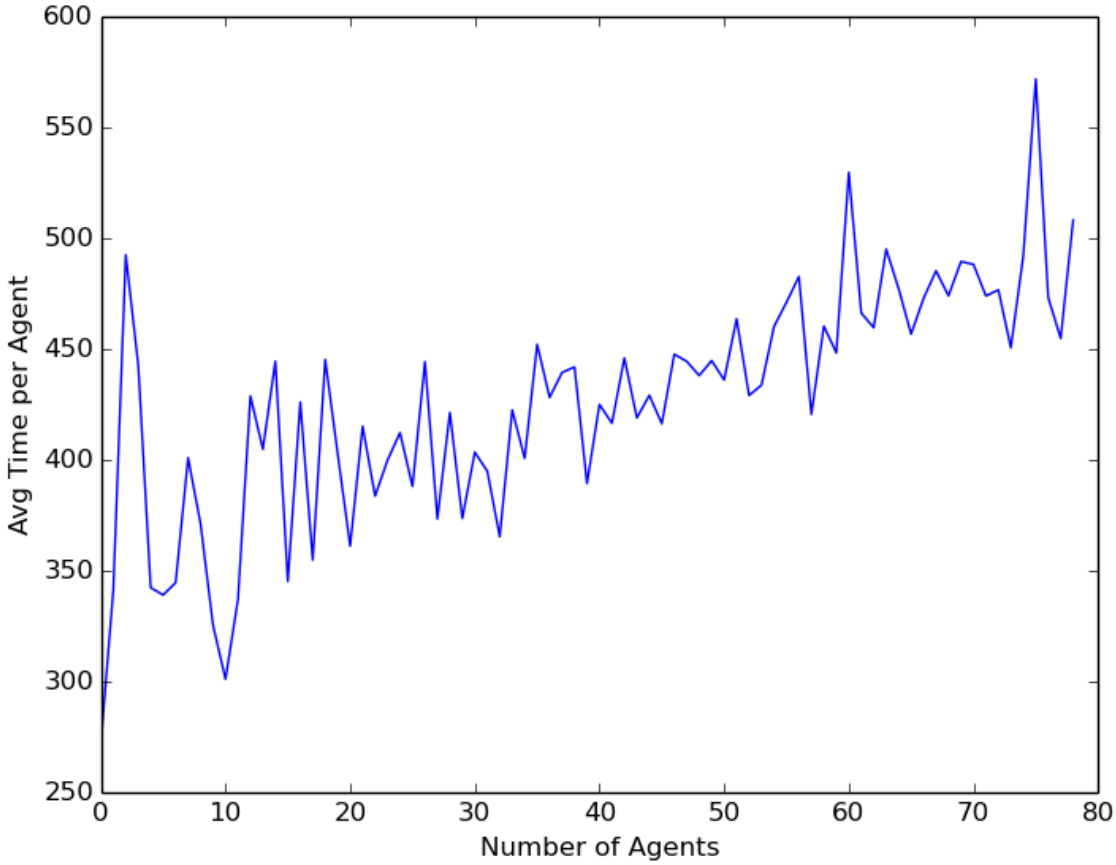
This result can be justified on the basis of crowd in the grid. As the crowding factor of the grid increases, agents need to travel larger distances to avoid each other. That distance will show linear variation as the force per agent will increase with increase in number of agents.




*Graph 3 Average Speed per agent vs Number of Agents*

Graph 3 shows a nearly constant relationship between the Number of Agents and Average Speed per Agent. The average speed per agent tends to remain constant with increasing number of agents.

This shows that the agents will travel more distance to maintain desired speed. Since the average human speed is around 1.5 m/s, our results are within acceptable bounds of scientific experiments. The steep drop can be justified with all agents being randomly allocated the same Finite State Machine and the same points of origin.




*Graph 4 Average Time an Agent travels vs Number of Agents*

Graph 4 shows a linear relationship between the Number of Agents and Average Time per Agent. The average time per agent tends to increase with increasing number of agents.

Since distance is also showing a linear trend, the time an agent stays in the grid should also show linear trend so that velocity can remain constant as $distance \propto time$.

# Conclusion

The Simulation of AGENTS was successfully observed using finite state machine, social force model and probabilistic road map. The Simulation can be used to make crowd animations, war scenes or even general backgrounds in video clips. The complexity of our simulation was $O(n^2)$. This is because we were making calculations for each agent w.r.t every other agent. We were also able to achieve a nearly constant speed for the agent in our simulation; however, it was discovered that the average distance travelled by each agent (and hence the average time an agent remains active) shows a linear approach as the number of agents increase.

# Future Works

The current $O(N^2)$ algorithm can be simplified to O(N log N) as research shows that only the nearest four agents are required for force calculations. This will need an extensive implementation of a new data structure which has so far not been used in this project.

The simulation currently works on spherical agents. This can be rectified with properly designed individuals to give this simulation a realistic feel. That itself would require major design skills and to model all the movements of human was currently beyond the scope of this project.

Lastly the simulation requires a proper development team for any major edits. This can be changed by developing an easy-to-use API which can also include a simple GUI for new scenarios and Finite State Machines.

# References


[1] F. P. Tasse and others, “Crowd simulation of pedestrians in a virtual city,” *Bachelor of Science Honours, Rhodes University,* 2008.

[2] H. Kidokoro, T. Kanda, D. Brščić and M. Shiomi, “Simulation-Based Behavior Planning to Prevent Congestion of Pedestrians Around a Robot,” *IEEE Transactions on Robotics,* vol. 31, pp. 1419-1431, 2015.

[3] B. Ulicny and D. Thalmann, “Towards Interactive Real-Time Crowd Behavior Simulation,” in *Computer Graphics Forum*, 2002.

[4] S. Curtis, A. Best and D. Manocha, “Menge: A modular framework for simulating crowd movement,” *Collective Dynamics,* vol. 1, pp. 1-40, 2016.

[5] L. Kavraki and J.-C. Latombe, “Randomized preprocessing of configuration for fast path planning,” in *Robotics and Automation, 1994. Proceedings., 1994 IEEE International Conference on*, 1994.

[6] M. H. Overmars and others, A random approach to motion planning, Citeseer, 1992.

[7] D. Helbing, I. Farkas and T. Vicsek, “Simulating dynamical features of escape panic,” *Nature,* vol. 407, no. 6803, pp. 487-790, 2000.

[8] M. Sung, M. Gleicher and S. Chenney, “Scalable behaviors for crowd simulation,” in *Computer Graphics Forum*, 2004.

[9] A. Sanfeliu and A. Garrell, “Cooperative social robots to accompany groups of people.,” *The International Journal of Robotics Research,* vol. 31, no. 13, pp. 1675-1701, 2012.