-

Program Structure And Algorithm

Crowd Control System

using

Particle Swarm Optimization

Final Project

Report by:

Purva Bundela

ABSTRACT

This report presents a uniform conceptual model based on the particle swarm optimization (PSO) paradigm to simulate crowds in computer graphics. According to the mechanisms of PSO, each person (particle) in the crowd (swarm) can adopt the information to search a path from the initial position to the specified target (optimum) automatically. However, PSO aims to obtain the optimal solution, while the purpose of this study concentrates on the generated paths of particles. Hence, in order to generate appropriate paths of people in a crowd, we propose a method to employ the computational facilities provided in PSO. The proposed model is simple, uniform, and easy to implement. The results of simulations demonstrate that using PSO with the proposed technique can generate appropriate non-deterministic, non-colliding paths in several different scenarios, including static obstacles, moving targets, and multiple crowds.

INTRODUCTION

Crowd control focuses on creating a realistic smooth and flexible motion for virtual human beings by utilizing the computational facilities provided in Particle swarm optimization (PSO). In particular, we present a uniform conceptual model based on particle swarm optimization (PSO) to simulate the motion of all persons in a crowd according to the analogy between a swarm and a crowd.

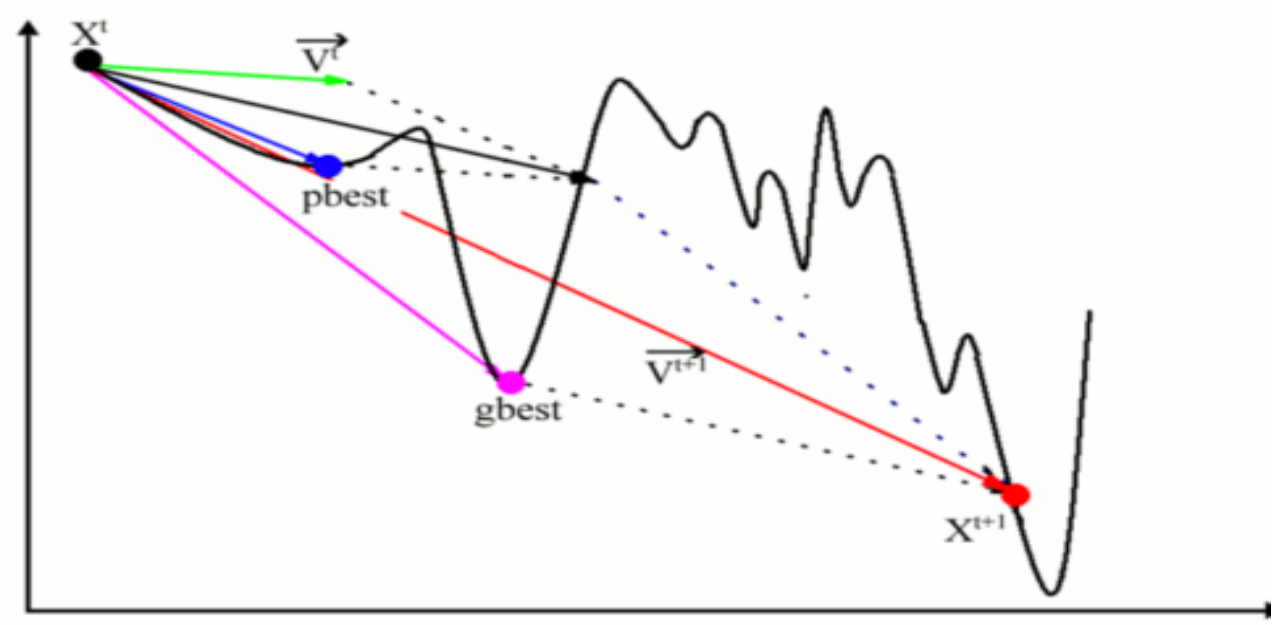
A person can be considered as a particle, which would like to find a way to reach the best solution. Although PSO does possess some characteristics of the crowd behavior, it is still incompatible with the use for crowd control. Firstly, the particle in PSO is absolutely free to fly through everywhere in the given multidimensional space.

However, the environment for a crowd may have obstacles, and the pedestrians in the crowd must avoid collisions, including the collision with the given obstacles and the collision with the fellow pedestrians, where other pedestrians can be considered as dynamic obstacles. These dynamic obstacles are not predictable and may appear and disappear in the environment at any moment.

PARTICLE SWARM OPTIMIZATION

Particle swarm optimization (PSO) is a population based stochastic optimization technique developed by Dr. Eberhart and Dr. Kennedy in 1995, inspired by social behavior of bird flocking or fish schooling. PSO shares many similarities with evolutionary computation techniques such as Genetic Algorithms (GA). The system is initialized with a population of random solutions and searches for optima by updating generations. However, unlike GA, PSO has no evolution operators such as crossover and mutation. In PSO, the potential solutions, called particles, fly through the problem space by following the current optimum particles. Each particle keeps track of its coordinates in the problem space which are associated with the best solution (fitness) it has achieved so far. (The fitness value is also stored.) This value is called ***pbest***. Another "best" value that is tracked by the particle swarm optimizer is the best value, obtained so far by any particle in the neighbors of the particle. This location is called ***lbest***. when a particle takes all the population as its topological neighbors, the best value is a global best and is called ***gbest***. The particle swarm optimization concept consists of, at each time step, changing the velocity of (accelerating) each

particle toward its **p*best*** and ***lbest*** locations (local version of PSO). Acceleration is weighted by a random term, with separate random numbers being generated for acceleration toward ***pbest*** and ***lbest*** locations.



Steps followed by Particle swarm optimization:
Initial
Set the position and the velocity of each particle.
Evaluate
Compute the objective value by the objective function.
Update PBLS
Update the PBLS of each particle by its objective value.
Update PBGS
Update the PBGS in the swarm.
Update Velocity and Position
Update the velocity and the position of each particle.
BLS: Best local solution
BGS: Best global solution

PROBLEM PROPOSED

In recent years, with the rapid increase of population, the number of disasters has increased sharply. It is always seen that in public areas whenever a disaster occurs the more damage is done to humans by human themselves. Therefore, a security system which can ensure all personnel evacuation needs to be established urgently. The security system should be matched with set routes, procedures, effective decision support and management, rescue apparatus, and so forth. An optimized design scheme should be considered in crowd management, which can enhance crowd

evacuation performance. Even in the situation of safety measure failures, available people assembly and evacuation can also be considered as the final safety shelter to avoid the disaster and the economic loss. Thus, the design of efficient evacuation model has become one of the hottest research areas of several industries.

In the personnel evacuation progress, dynamic programming for behavior of personnel is needed. With the wide use of evacuation model, more and more research institutions have been dedicated to exploit it. The majority of evacuation models adopt implicit actions, based on function programming, rule-based behavior, or basic behavior of intelligent agent to express evacuation behavior.

SOLUTION PROPOSED

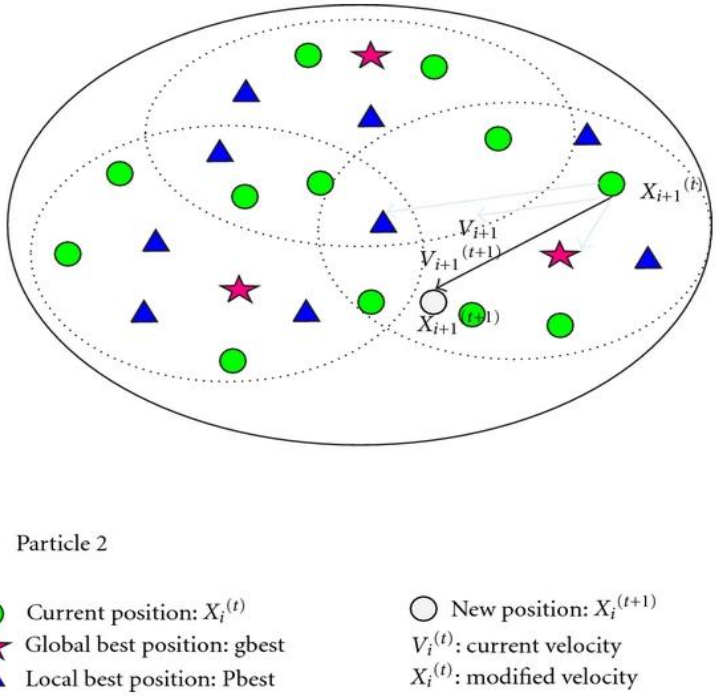
In neighborhood particle swarm optimization algorithm, the particles update their velocity and position according to individual behavior, the behavior of the whole group, and the best individual experience in the neighborhood. In the process of evacuation, the neighborhood of a particle is the channel node area which is connected with the position of current particle. Putting the neighborhood learning mechanism into the PSO algorithm not only can simulate the herd behavior of the crowd in the process of evacuation successfully, but also can effectively avoid the problem of all the particles moving toward the local optimal point which could lead to large-scale staff conflicts and congestion.

Stepwise explanation for Algorithm execution.

1. Create the random particles which will act as human in the region.
2. Initialize the human particles with random positions and velocities.
3. Evaluate the fitness value of system.
4. Calculate system constraints for each particle and total error.
5. Compare the individual fitness value of each particle to its previous value, if it is better than previous one, replace with new value i.e. local best position otherwise remain the same.
6. The position of particle having lowest fitness value is global best value.
7. Update position and velocity of particle according to Step-(1).
8. Go back to step-(3) and repeat all steps until system constraints are met Here

For The optimization of parameters the following constants and equation has been used.

Number of Particles = 50;	Size of the Human_Particles
Iterations = 100;	Maximum iterations for solution.
Dimension = 2;	Dimensional space
C2 = 1.3;	PSO parameter C2, importance of neighborhood best
C1 = 0.14;	PSO parameter C1, importance of personal best
w = 0.9;	PSO momentum or inertia



Calculation for fitness Value:-

Fitness Value = Math.sqrt((xTarget - x) * (xTarget - x) + (yTarget - y) * (yTarget - y));
Where
xTarget = X coordinate of Target position.
yTarget = Y coordinate of Target position.
x = X coordinate of current position of a particle.
y = Y coordinate of current position of a particle.

Fitness value we get in the equation is basically the distance between the target position and the current position of sensor. By looking at the fitness value we get to know about the proximity of the sensors from the target. Fitness value mentioned above is the **Euclidean Distance Formula i.e.**

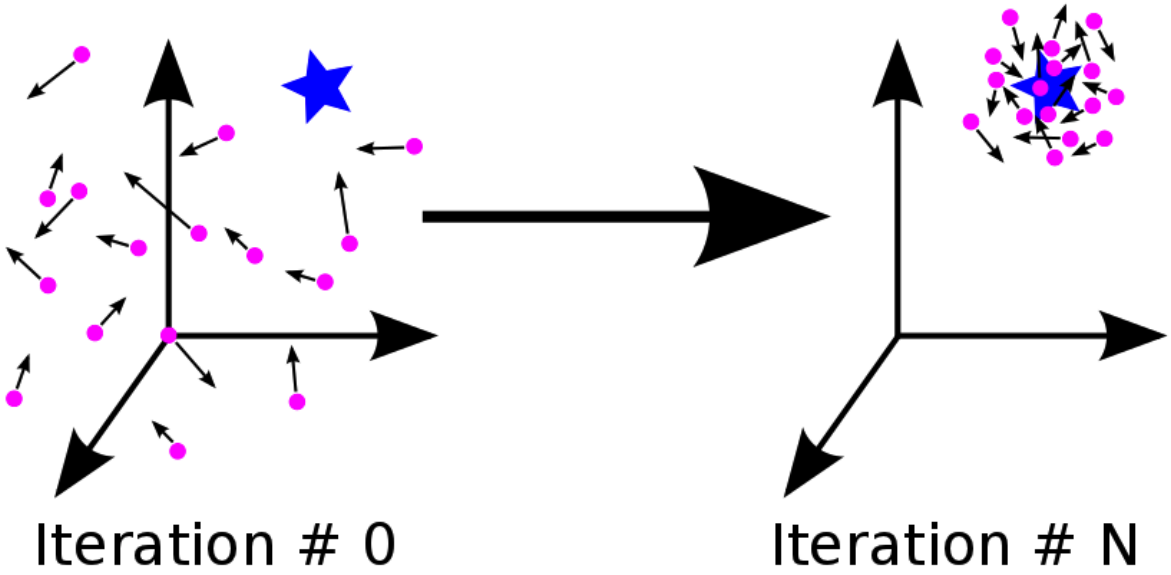
$$Euclidean\ Distance = d = \sqrt{\sum_{i=1}^{N}(Xi - Yi)^2}$$

Calculation for Velocity:-

Velocity = (W * Velocity) + C1 * (R1 * (Local_Best_Coordinate – Current_Coordinate)) + C2 * (R2 * (Global_Best_Coordinate –Current_Coordinate))

Next_Postion = Current_Postion + Velocity;

Thus, after the N Iteration execution of the program all the human_particles converge at a target point i.e our exit door in this scenario.



Fitness Value Calculation:

```
/**
 *
 * @author Panchi
 */
public class Fitness_Human implements Crowd_Project_Constants{



    public  double calculate_fitness(Location location) {
                double result = 0;
                double x = location.getLocation()[0]; // the "x" part of the location
                double y = location.getLocation()[1]; // the "y" part of the location

                result =  Math.sqrt(Math.pow(TARGET_X-x, 2)+Math.pow(TARGET_Y-y, 2));

                return result;
        }



}
```

Achieving Target Value:

```
            double r1 = generator.nextDouble();
            double r2 = generator.nextDouble();

            Human_Particle p = swarm.get(i);

            // step 3 - update velocity
            double[] newVel = new double[PROBLEM_DIMENSION];
            newVel[0] = (w * p.getVelocity().getPos()[0]) +
                                (r1 * COSTANT_1) * (pBestLocation.get(i).getLocation()[0] - p.getLocation().getLocation()[0]) +
                                (r2 * COSTANT_2) * (gBestLocation.getLocation()[0] - p.getLocation().getLocation()[0]);
            newVel[1] = (w * p.getVelocity().getPos()[1]) +
                                (r1 * COSTANT_1) * (pBestLocation.get(i).getLocation()[1] - p.getLocation().getLocation()[1]) +
                                (r2 * COSTANT_2) * (gBestLocation.getLocation()[1] - p.getLocation().getLocation()[1]);
            Velocity vel = new Velocity(newVel);
            p.setVelocity(vel);

            // step 4 - update location
            double[] newLoc = new double[PROBLEM_DIMENSION];
            newLoc[0] = p.getLocation().getLocation()[0] + newVel[0];
            newLoc[1] = p.getLocation().getLocation()[1] + newVel[1];
            x[i]= newLoc[0]*200;
            y[i]= newLoc[1]*200;
            Location loc = new Location(newLoc);
            p.setLocation(loc);
    }
```

Graphical Represtaion Of Human_Particles Evacuating To The Door:

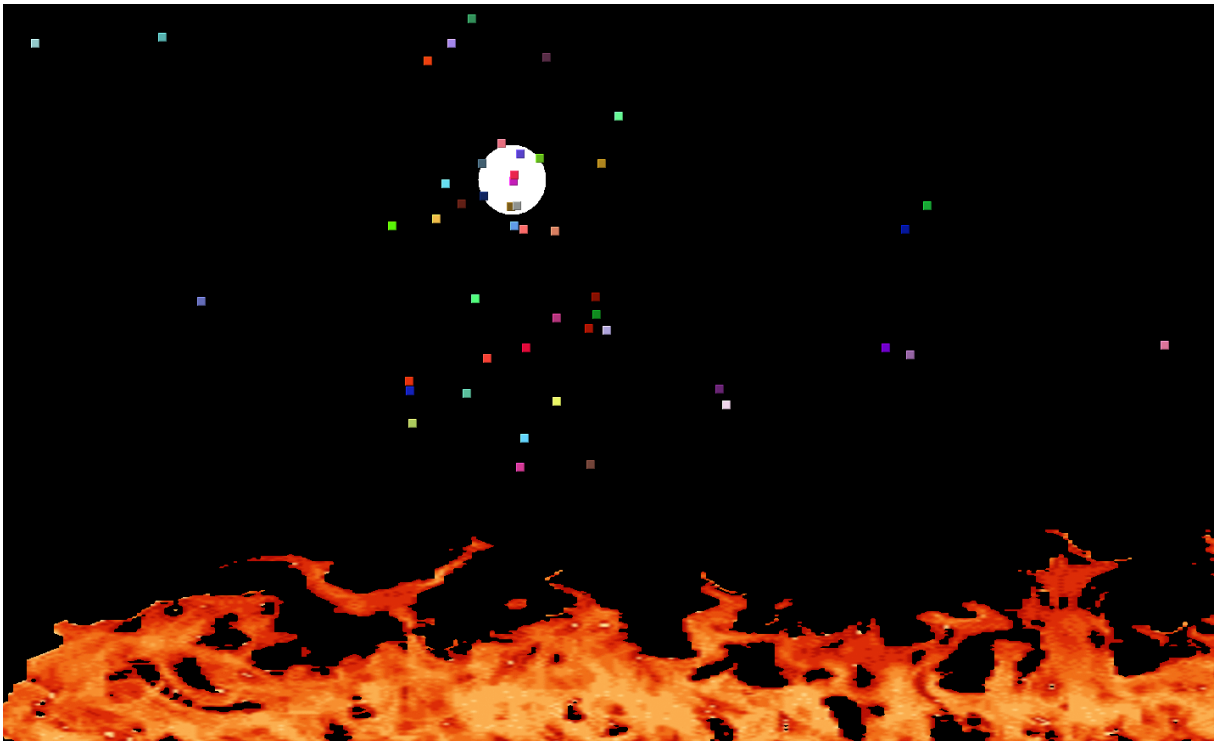


Fig.1 Particles Scattering due to Fire

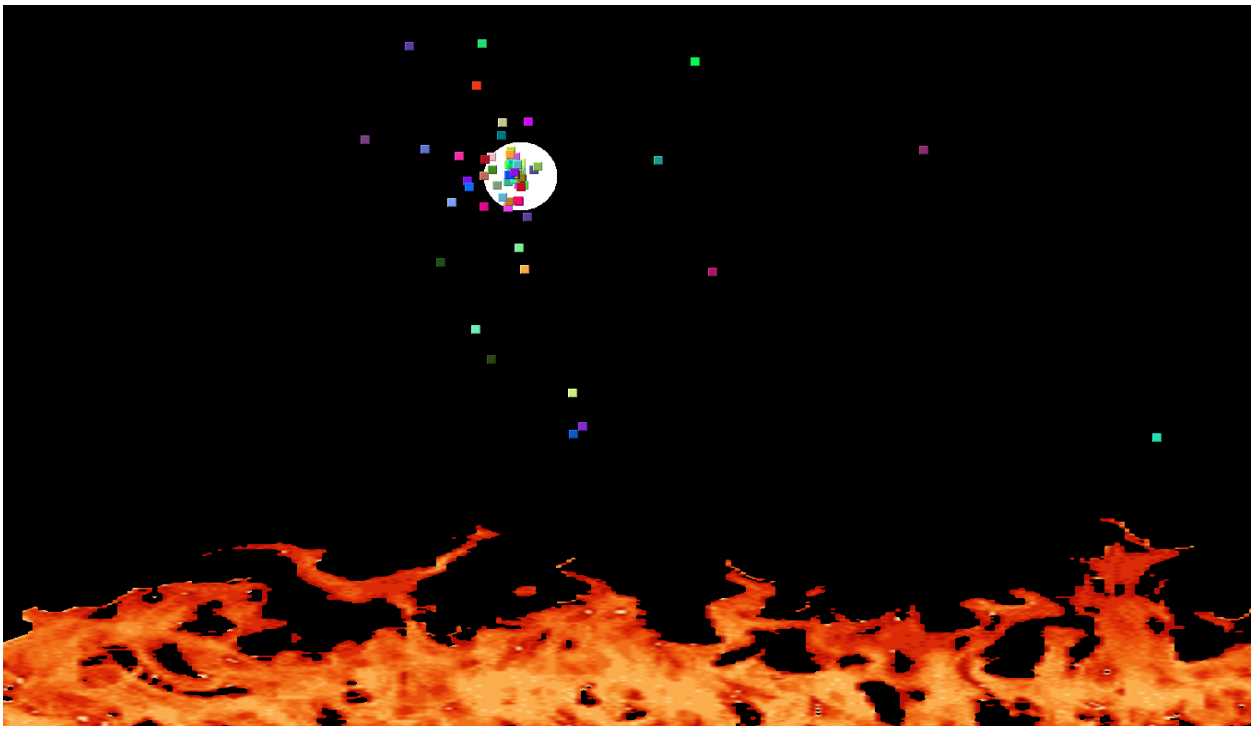
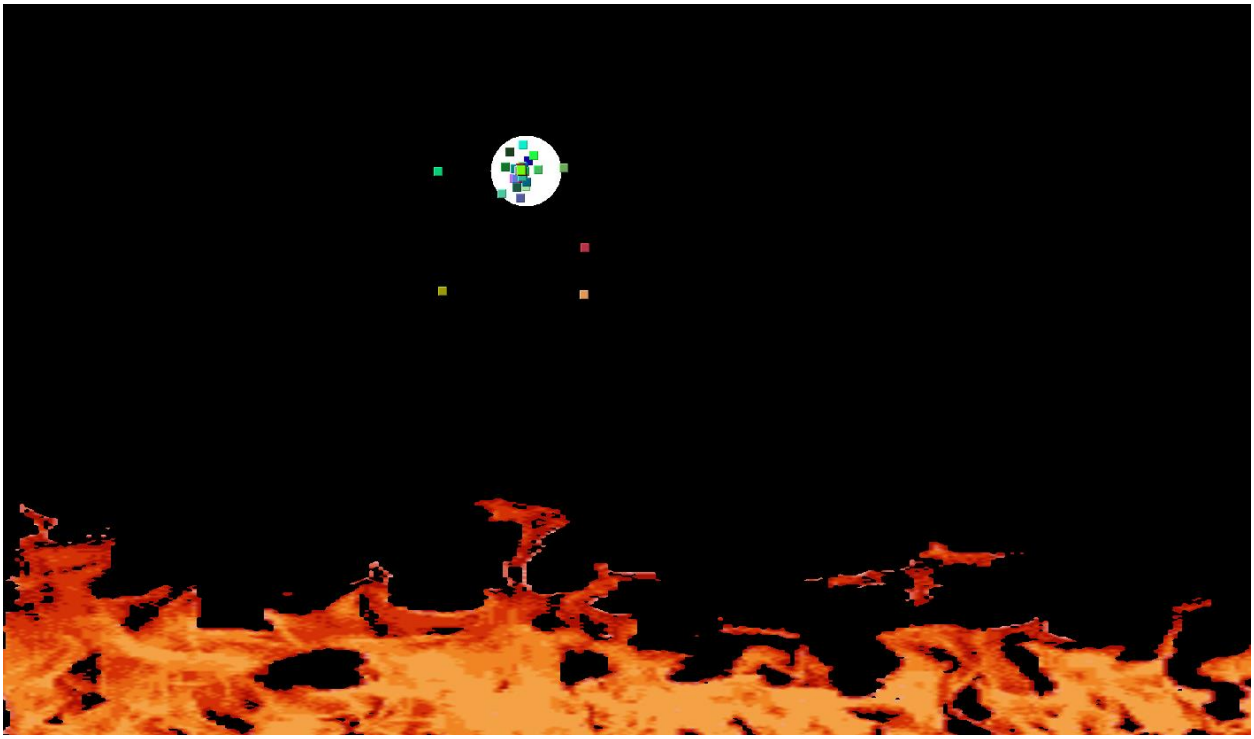


Fig.2 Particles Converging in Order To Find A Exit



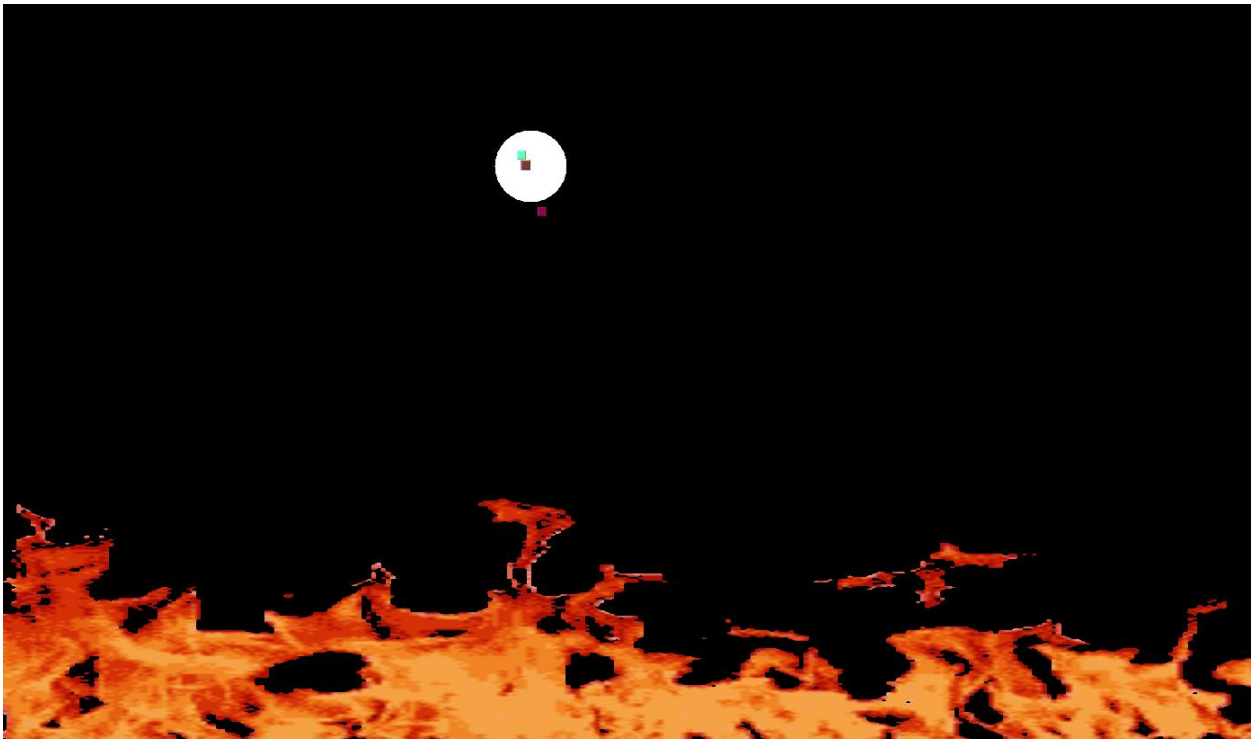


Fig.3 & 4 Particles Found the Exit and trying to Leave.

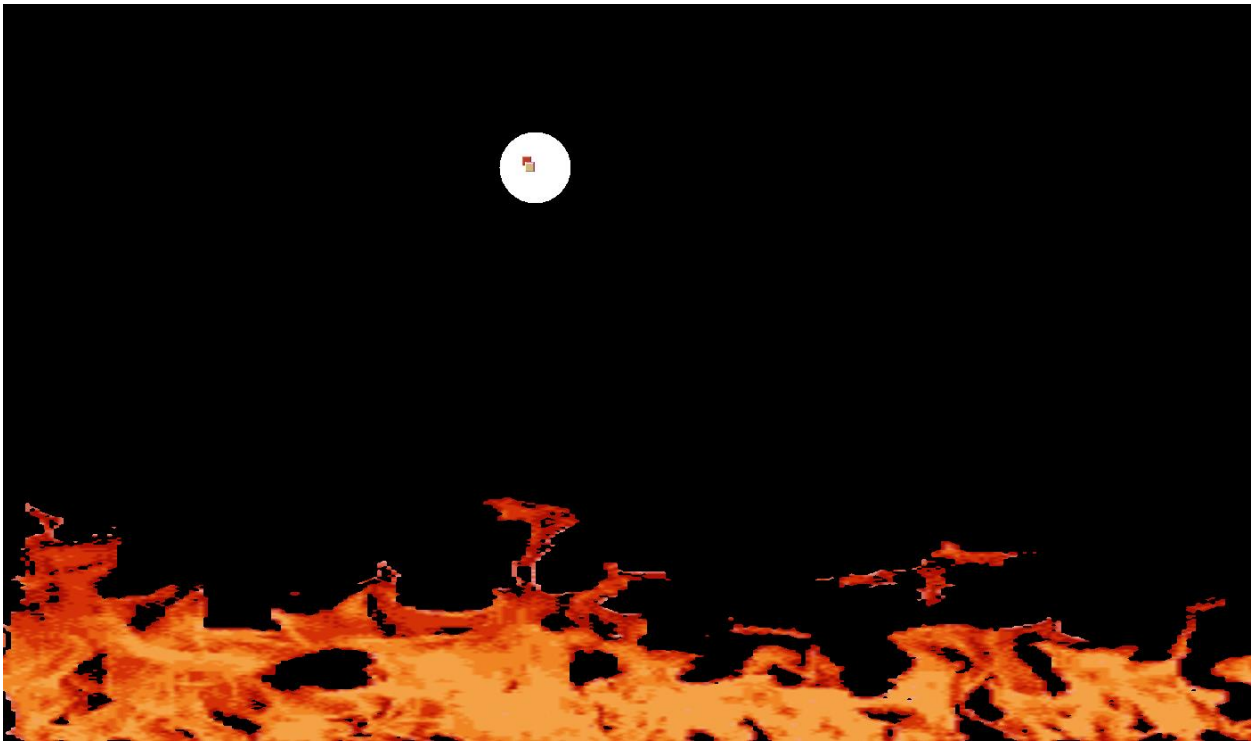


Fig.5 All Particles Are Safely Out Of Danger.

THE INFLUENCE OF DOOR'S DIMENSIONS

Different sizes of the door have quite different evacuation capacities. Obviously, the larger size can get the well effect of evacuation. But in the period of ship design, the size of doors cannot be randomly designed due to the limitation of space and size of ship. Hence, the most appropriate size for doors needs to be designed under limitation of surroundings and conditions to improve the evacuation capacity to the ultimate extreme. The influence of doors' dimension for the process of evacuation is shown in Figures 6 and 7.

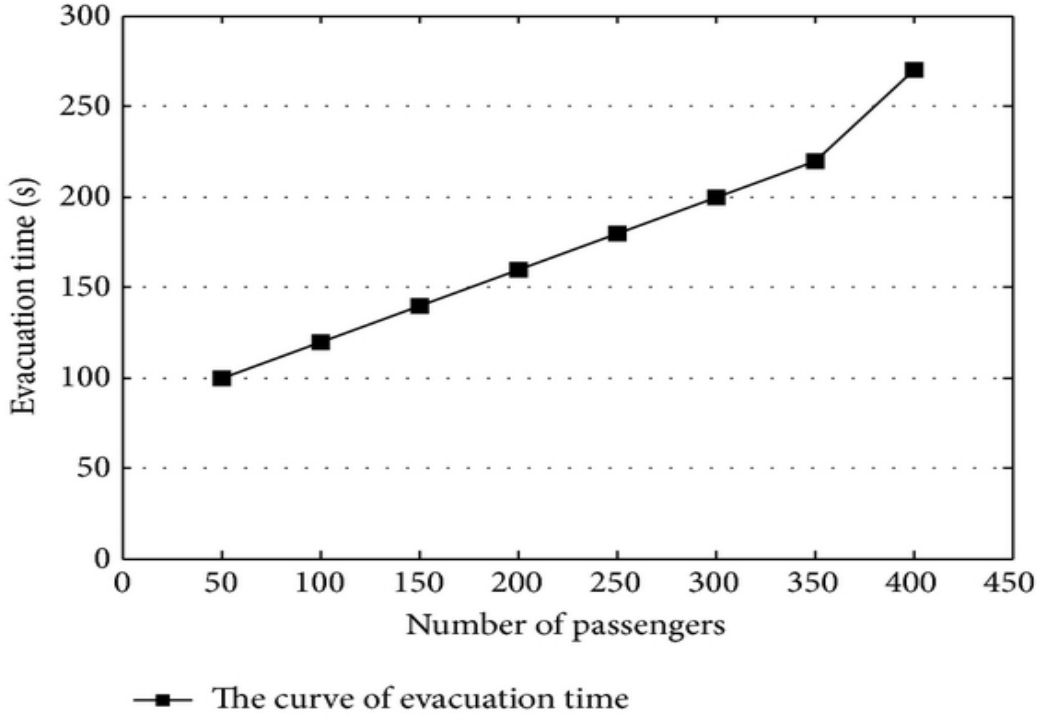


Figure 7: The relationship between evacuation time and the number of passengers.

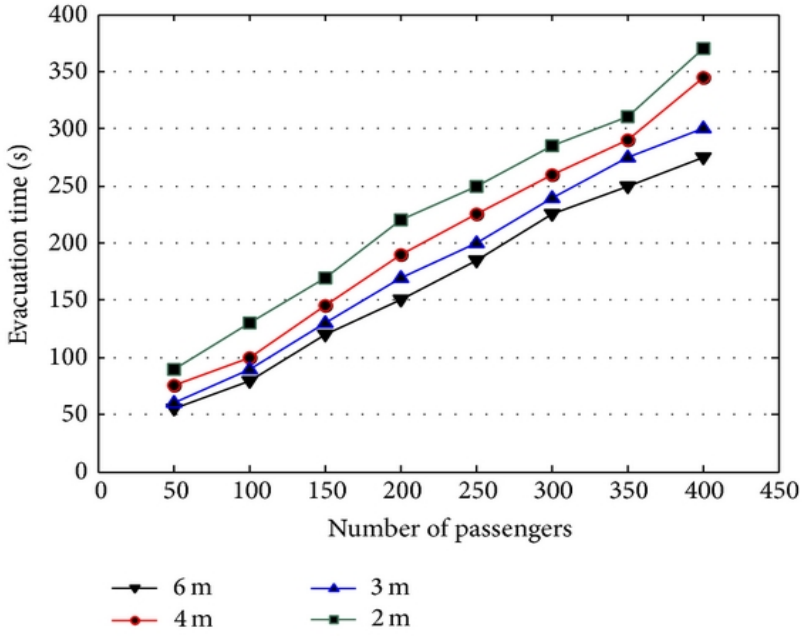


Figure 4: Effect of door size and number of people on evacuation time.

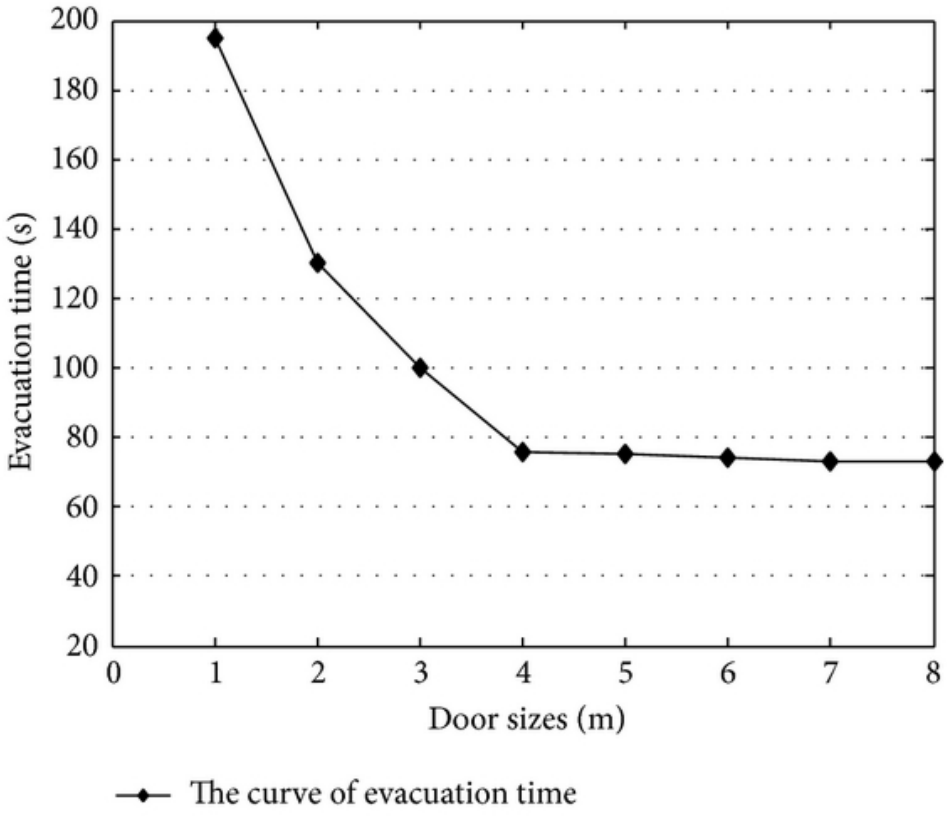


Figure 3: Effect of the door size on evacuation time.

CONCLUSION

Particle swarm optimization (PSO) is an optimization paradigm proposed in the field of evolutionary computation for finding the global optimum in the search space. The concept of PSO is easy to comprehend, and the mechanism is easy to implement. The ability of PSO to reach the position of the optimum creates the
possibility to automatically generate non-deterministic paths of virtual human beings from one specified position to another.
This project mainly studies how to simulate the process of passengers' evacuation under the circumstance of emergency evacuation and establishes the simulation model of the passengers based on the neighborhood particle swarm optimization algorithm. In this paper, we consider the particles in the swarm as the individuals in evacuation, and the behaviors of the particles are directed by their own learning abilities, the perception abilities of environment, and the social attributes, which can simulate the real process of evacuation.