Crowd Simulation

Introduction

The increasing proportion of people living in urban areas brings new challenges to urban planning and architecture. Crowd simulation plays an important role in addressing these challenges. With the help of crowd simulation techniques, urban designers and architects could determine the evacuation time of a massive crowd, predict the behavior of a crowd flow inside of a building or prevent overcrowding during certain events.

A crowd forms when a large amount of people gathers in a limited space. Simulating the whole crowd as a single unit could help understand the behavior of the moving crowd. However, if we divide the crowd into groups that contains 2 to 3 people or individuals, the behavior of the crowd can be more realistic. In a group, people know each might walk together. Previous researcher Reynolds [1] proposed a steering approach known as Leader Following (LF). This approach involves pair agents where the "follower" agent follows the leader and stays on its side. This disadvantage of this approach is that in this basic steering approach, the leader agent does not wait for its follower agent if the distance between these two agents is too large, which is not realistic.

More recent simulations of crowds of people use more complicated calculation. For example, previous approach [2] designs agent as ellipses that have a sense of the environment and plan their own path ahead of time to avoid agent collisions. Unfortunately, the output of this kind of simulation lacks realism and flexibility. Since it does not involve dynamic behaviors such as allowing agents to move in and out of different group or queues based on agent's desire, agents who have planned a path ahead of time might end up in the longest waiting line without being able to switch. In reality, people do not just stay in their waiting line once they choose it, they might need to change waiting lines if there is a better option.

Proposed project objectives

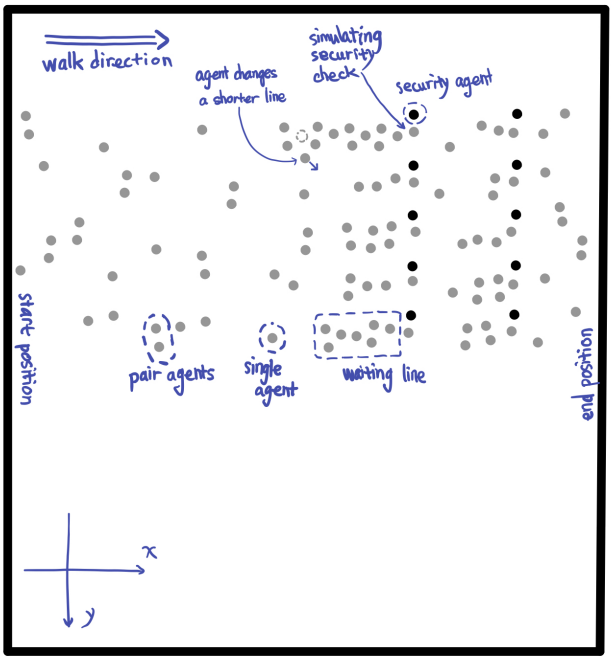
In this project, I will develop a crowd simulation application which aim at creating realistic, unique and accurate crowd.

Scenario: Single agents or pair agent are randomly generated within the initialized range; each agent is initialized with a default start position and end position, and they will walk to their end position.

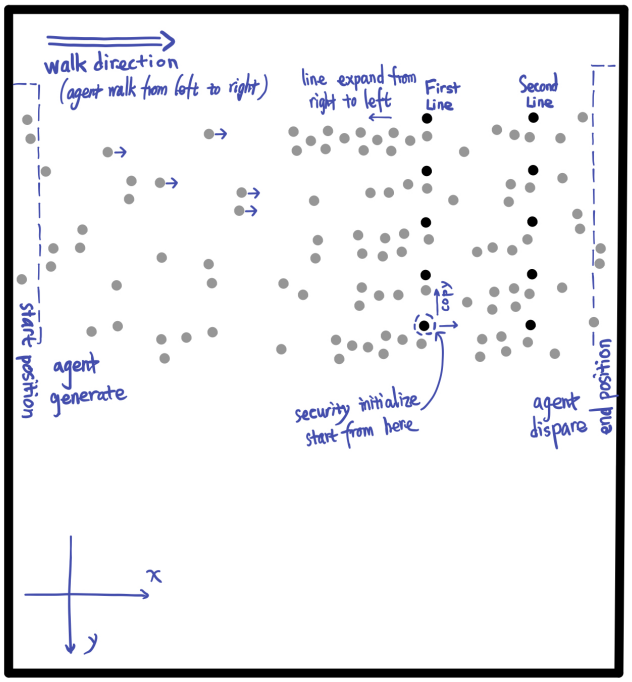
However, before agents reaching their destination, every agent has to finish security check first. (People do ticket checking and security check during the concert event). Thus, every agent needs to stop near the security to simulate the security process. Since the new agents are generating and security process takes time, the number of agents in the scene increase dramatically and a the crowd form. However, instead of generating a massive chaotic crowd, agents in the crowd will queue up orderly and form several waiting lines, and each agent in line will do the security check one by one. After finishing the security check, agents will move to their default end position and depart.

To achieve this goal, I will use an open source state of art navigation mesh construction toolset called Recast to achieve static avoidance and shortest path calculation. What's more, I will also utilize a path-finding and spatial reasoning toolkit Detour to achieve dynamic avoidance among agents in the path and to completed calculation of each frame of the simulation [3]. Using these open source platforms, I will build the lower level of my approach – QueueBehaviorApp. QueueBehaviorApp will simulate crowd behaviors such as pair walking, queue up and form single/pair waiting lines, agent switch from one waiting line to the others and then doing security checking. The image below is the mockup demo that demonstrates the scene I will create. In this scene, agents are generated and walk from left to right. Among each agent, some agents might know each other, so they walk with each other while others walk alone. Once agent reach the security gate, agents stop for a few seconds to simulate the security checking process. If the security gate is occupied, the upcoming agents wait

behind. Thus, the lines are created. If agents in the line are not satisfied with the length of the line, they will look to the left of right side of line to find out if there are any shorter line to go to. If there is a shorter line, the agent will leave its original line and queue up to a better line.



However, before doing that, I will initialize scene to generate input for the crowd simulation. Below is the mockup demo of the application interface with detailed requirement that could help understand the input and default requirements of the crowd simulation:



In this simulation, agents are initialized at the left side of the scene (*start position*) and disappear at the right side of the scene (*end position*). Agents in scene will move from left to right, this moving direction determines the direction of the waiting line, with the waiting line grow from right to left. In the simulation, there are two lines of agents always standing at the same position in the whole

simulation. These two lines of agents play roles as security faculties of the event in the simulation to simulate security check. The upcoming agents will stop in front of the security faculty for a few seconds to simulate a security check (or a ticket check) as we do in our real life. Once agents finish check, they continue moving to their final destination.

Method

The program will use the open source Recast/Detour libraries in Java to achieve the lower level features. I will create a crowd simulation by implementing algorithms and data structures to generate output data file that contains information of every agent coordinate in every frame, and then visualize the results with an HTML canvas to create the results animation.

To make sure the simulated crowd behavior is realistic and dynamic, I will first collect different crowd behavior features from several rea-life crowd video recorded by Dr. Ricks research lab. Those videos recorded the walking crowd during several events such as concert, Disney On Ice, etc.

Expected results

Agents single or in pair walk across the scene and pass through two lines of gates. When a there has large amount of people have appeared, agents line up and create a certain number of waiting lines and each agent consecutively pass through the gates one by one. By comparing the length of the distance between agent's current position to the gate and length of nearby waiting line, agent in the waiting line might increase or decrease its anxiety level. Once agent's anxiety degree reaches an upper bound, agent will leave its own waiting line and line up at a new line.

Format of report

1. Application source code
2. Application animation
3. Application screenshot

Project mentor

Dr. Brian Ricks

Qualifications

Comparing the simulation result with the real-life video record to see how the simulation result compare to the video.

References

[1] Reynolds,C.:Steeringbehaviorsforautonomouscharacters.In:GDC,pp.763–782(1999)

[2] Baig, Mirza Waqar, et al. "Realistic modeling of agents in crowd simulations." 2014 5th International Conference on Intelligent Systems, Modelling and Simulation. IEEE, 2014.

[3] Open source React and Detour. https://github.com/ppiastucki/recast4j