Social Distance Simulation

Artificial Intelligence final term project

Group members: Kalpana Shankhwar (D06522022), Calvin Liu (B06902100), Zavien Huang (B07902082)

Prof. Jane Yung-jen Hsu
Department of Computer Science & Information Engineering
National Taiwan University

Calvin Liu (B06902100)

Zavien Huang (B07902082)

Kalpana Shankhwar (D06522022)

Communication Mechanism: Group chat (Messenger) + Occasional face to face meetings

Github link:

https://github.com/CalvinL15/AI-Final-Project

Motivation/Background

- Social distancing is the enforced idea that governments around the globe have been pushing and encouraging or even forcing their citizens to abide by during this pandemic.
- Social distancing means physical contacts should be avoided, which would help in reducing the chances of COVID-19 spreading, thus lowering the rate of infection.
- The main drawback of social distancing such as quarantining/lockdowns is the decreased productivity of our society and economy. For example, many regions cannot even perform normal tasks to thrive regularly which put many jobs at risk; either work is still allowed but with extreme distancing measures or even worse, many employees simply get laid off.

Motivation/Background

- If we can find the route from our origin location to the intended destination where social distancing can be maintained, many restrictions that are currently in place all around the world to curb the spread of COVID-19 can be eased since social distancing can be maintained more easily all the time, which means that resuming/restarting the economy can be achieved faster and safer too.
- All in all, it is important to prevent the spread of virus which can be reduced by minimizing the distance between people. The minimum social distance conventionally agreed by most countries is 1-1.5m. This project presents an artificial intelligence model to maintain a 1m distance among the pedestrian passing through a predefined region.

Related Work

- Study of research paper “Enabling and Emerging Technologies for Social Distancing: A Comprehensive Survey.” regarding the possible artificial intelligence, machine learning and computer vision techniques for social distancing among people.
- Study of the application of artificial intelligence in monitoring the social distancing between the people in real world.
- Study of AI applications to detect the social distancing violations.
- Application of AI techniques for social distancing on the shop floors, construction sites, manufacturing units, traffic junctions, airports and business parks.

Target problem

- Our target problem is how the player can optimally reach the goal from its initial position to the goal while maintaining social distance with all of the pedestrians that are wandering around in the environment.
- This is a multi-agent problem that has a main goal-based agent, our player that is trying to reach the upper right corner goal and pedestrians that are the simple reflex agents.
- The states is the location of the agents in the environment for that particular instance that the player needs to optimally determine its next path. Our actions for the agents are limited to move up, down, left, right and stop with no diagonal movement.

Problem Specification

- In this project, we design a simulation in which we have a maze (gridworld), a player, and pedestrians.
- Player starts from bottom-left of the maze and has to reach to the destination which for this implementation is fixed at the top-right of the maze with the condition of always maintaining a defined distance with the pedestrians who are passing through the maze.
- Player has no information about how the pedestrians walk and map's layout.
- In this project, our 1.5m of social distancing translates to 3 units in our gridworld implementation, where we define unit as: abs(row_P- row_Pi) + abs(col_P - col_Pi), where row_P: y coordinate of player, col_P: x coordinate of player, row_Pi: y coordinate of a pedestrian i, and col_Pi: x coordinate of a pedestrian i.
- Therefore, the player and pedestrians should never be within 3 or less units (>= 4 units is safe).
- However, some pedestrians can wear a mask. If a pedestrian x wear a mask, then the safe distance between player and pedestrian x become >= 1.

Problem & Program Specs

- It takes 4 arguments to run the program: “name of the program” “level” “pedestrians_move_case” “algorithm_case”, where:
  - Level: determine the map of the simulation. We designed 6 mazes. (0-5)
    - 0: small with 1 pedestrian, 1: small with 2 pedestrians (11x17)
    - 2: medium with 2 pedestrians , 3: medium with 4 pedestrians (19x45)
    - 4: big with 3 pedestrians, 5: big with 5 pedestrians (29x99)
  - Pedestrians_move_case: determine how the pedestrians move. (0-2)
    - 0: pedestrians do not move
    - 1: refer to slides 11, 2: pedestrians move into the original position of the player
  - Algorithm_case: determine the algorithm that the player use to reach the destination (0-1)
    - 0: no algorithm (player move with keyboard keys: for fun only), 1: player moves with the designed algorithm.
- Example: python finalproject.py 3 0 1
- Some pedestrians can wear a mask, some do not. If a pedestrian x wear a mask, then the safe distance between player and pedestrian x become >= 1.

Proposed Solution (Methodology)

- In our gridworld, there would be multiple pedestrians moving in a specified way.
- Since our main agent or the player would always start in the most bottom left corner and its goal is at the most upper right hand corner, it would be consistently be subject to cross the group of pedestrians.
- We should constantly check if the distance between player and each pedestrian are larger than 3 units. We can achieve this by checking the current state and consider its potential move that lead its successive states into a resulting distance of >3 units between the player and each pedestrian.
- Because it is the player's main worry to maintain social distancing (pedestrians don't care at all), there are times where player cannot reach the destination, and the simulation would terminate as soon as the distance between the player and a pedestrian is within 3 units. (FAILURE)
- If the player successfully reaches the destination, the program will output "Success! {steps}", where steps = number of steps taken to reach the destination

Proposed Solution (Detailed Algorithm)

Pedestrian Movement Algorithm:

- Our pedestrians only move randomly if there are no walls around it within 2 units.
- When pedestrians do bump into one wall (not a corner with 2 walls blocking two available movement options), their instinct is to move the foremost away from the wall or along the wall for a while. Typically the better option would be towards the place that potentially has less walls.
- If they do end up in a corner. They should only have two movement options and will choose the direction to move towards if the potential state does not have 2 walls, or better, no walls.

Proposed Solution (Detailed Algorithm)

- Our gridworld is powered by the pycolab engine. The agents are created with 2 types: one is for the main goal-based agent, our player, and the others are our simple reflex agents, the pedestrians.
- Our most optimal movements for the player is by either moving up or right (diagonal move is not possible in this simulation) as the destination is in the upper right corner. However, it does not necessarily mean that the player will always move up or right.
- The player might move in another direction or just stay in its current position in the case where moving up or right might potentially result in a violation of social distancing. If it continued to only move right or up, it would either simply know that the next state would lead to a violation of social distancing or predict that the next action of pedestrians would most likely lead the player to be within 3 units of them.
- Move priorities: 1. Up or Right (depending on position of pedestrians and borders) 2. Not Moving, 3. Down or Left

Proposed Solution (Detailed Algorithm)

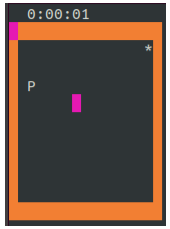
- Since the destination is in the upper right corner, it would always be possible for the player to move up or right (in a no obstacle map). So, when the distance unit between the player and each pedestrian is larger than 5, the player will always either move up or right. Why?
- In this simulation, the player and all pedestrians move simultaneously. In the worst case, the distance unit of the player and a pedestrian can be reduced by 2 (they both are moving closer to each other). So, since >3 is the safe distance, 5 is the safe distance for us to move optimally
- When the player are close to the pedestrians, we would have to determine the next best moves. We should first opt for the player to not move since moving down or left means we are going away from the destination. However, if the player not moving would lead it to most likely violate social distancing in its successive states, then we don't have a choice; we have to either move left or down.
- Another thing to consider is the wall. The player has to adjust its move accordingly when it encounters wall in its way.

Experiments (Design)

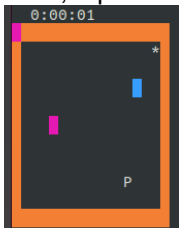
- We had different sized maps with varying numbers of pedestrians. It ranges from small to large maps and very minimal amount of pedestrians to a handful of up to 5.
- There are straight walls detached from one another and away from the wall that act like a barrier, such as the road or another building blocking the path of a person.
- The player would always attempt to start from the bottom left hand corner and optimally move towards the upper right hand corner where the goal is located.
- It would only fail and have the program terminate in cases where it simply cannot move anywhere without violating social distancing.

Experiments (Design)

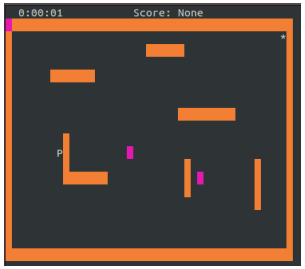
Maze 0, 1 pedestrian



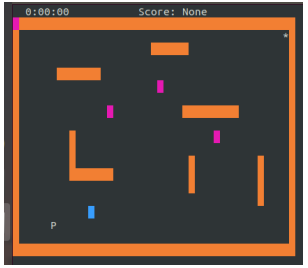
Maze 1, 2 pedestrians



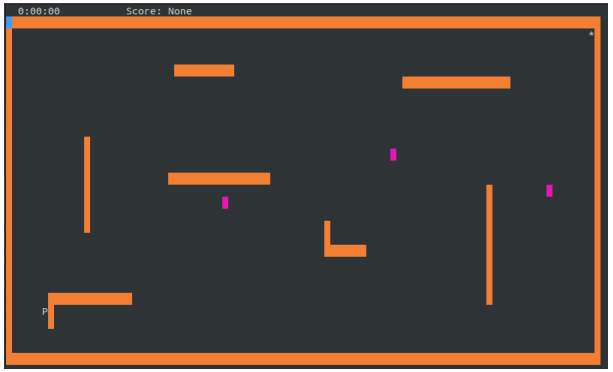
Maze 2, 2 pedestrians



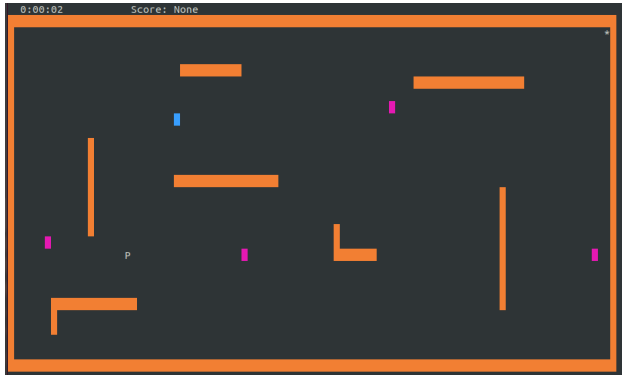
Maze 3, 4 pedestrians



Maze 4, 3 pedestrians



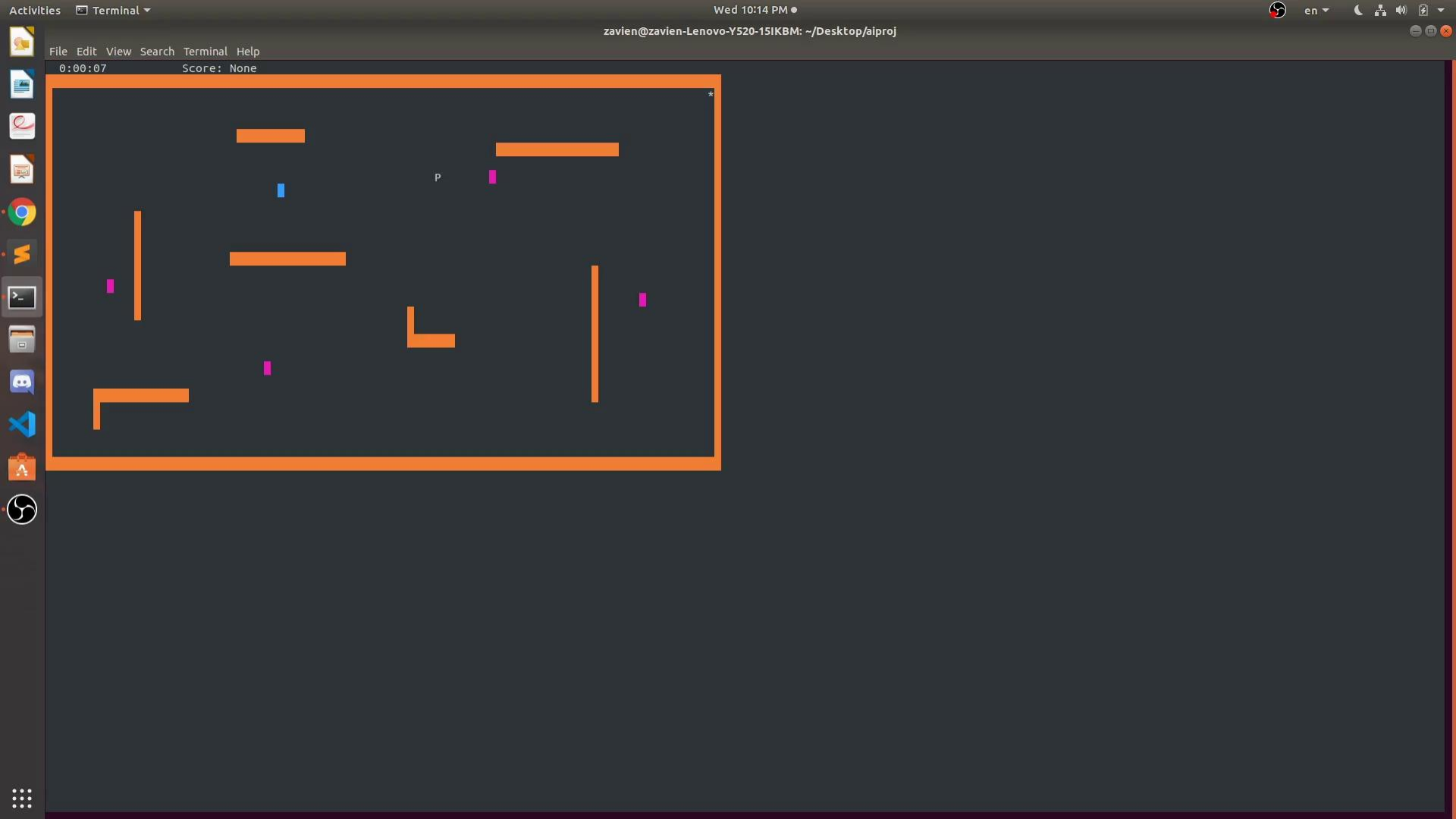
Maze 5, 5 pedestrians



P: player
*: destination

Pink square: a pedestrian without mask

Blue square: a pedestrian with a mask


Activities
Terminal
Wed 10:14 PM
en
zavien@zavien-Lenovo-Y520-15IKBM: ~/Desktop/aiproj
File Edit View Search Terminal Help
0:00:07 Score: None
P

Experiments (Resources + Tools)

- Modules used:
  - Pycolab
  - Sys
  - Curses
  - Random

Experiments (Results)

- The results can vary as each time the program runs it is unique. The chances of a reoccuring scenario is slim. However, smaller maze leads to the lower chance of unique scenarios.
- Most of the times we run the program, the player would reach the goal in a reasonable amount of time, about 10-30 seconds, depending on the size of the maze.
- When the maze is more compact with more pedestrians, the chances for the player being trapped and being forced to violate social distance would be higher. Thus, this would prompt the simulation to end by terminating the program.
- The next slide contains the results from a shell script running the program 20 times each for mazes 0 to 5. It shows the success and fails and if it succeeds to reach the goal, the number next to it is the total amount of steps the player has took.

Experiments (Results)

Maze 0:
FAILURE!
Success! 44
Success! 46
Success! 36
Success! 41
FAILURE!
Success! 34
Success! 30
Success! 57
Success! 48
Success! 50
Success! 50
Success! 51
Success! 46
Success! 45
FAILURE!
Success! 22
FAILURE!
Success! 37
FAILURE!
Success Rate: 15/20

Maze 1:
FAILURE!
FAILURE!
FAILURE!
FAILURE!
Success! 25
Success! 44
FAILURE!
Success! 37
FAILURE!
FAILURE!
Success! 26
Success! 37
Success! 40
Success! 56
Success! 28
Success! 32
FAILURE!
Success! 31
Success! 32
Success! 37
Success Rate: 13/20

Maze 2:
Success! 96
Success! 101
FAILURE!
Success! 105
Success! 110
Success! 83
Success! 73
FAILURE!
FAILURE!
FAILURE!
Success! 74
Success! 64
Success! 61
FAILURE!
FAILURE!
Success! 65
Success! 67
Success! 82
Success! 65
Success! 67
Success Rate: 13/20

Maze 3:
Success! 88
FAILURE!
Success! 84
FAILURE!
FAILURE!
FAILURE!
FAILURE!
FAILURE!
Success! 90
FAILURE!
Success! 61
FAILURE!
Success! 77
FAILURE!
Success! 81
FAILURE!
FAILURE!
Success! 88
FAILURE!
Success! 68
Success Rate: 8/20

Maze 4:
Success! 163
Success! 122
Success! 124
Success! 122
Success! 130
Success! 135
Success! 134
Success! 124
Success! 151
Success! 137
FAILURE!
Success! 152
Success! 124
Success! 135
Success! 126
Success! 120
Success! 134
Success! 120
Success! 126
FAILURE!
Success Rate: 18/20

Maze 5:
Success! 131
Success! 126
Success! 148
Success! 119
FAILURE!
Success! 131
Success! 129
Success! 121
FAILURE!
Success! 145
Success! 142
Success! 125
Success! 149
Success! 126
Success! 142
Success! 138
Success! 131
Success! 147
Success! 136
Success! 128
Success Rate: 18/20

Experiments (Analysis)

- We implemented shell scripting to run the program 20 times and collected the results.
- We have considered 6 cases for the maze to check the success rate for the player to reach the goal, indexed from maze 0 to maze 5 (can be changed via first argument in stdin).
- Each simulation is done 20 times for all 6 cases and their data has been recorded.
- The number of steps is displayed along with the SUCCESS!. SUCCESS 44 means that the player reaches the goal in 44 steps.
- As per the chart mentioned above, maze 0 has 12 successful runs out of 20 and so on.
- Maze 5 gives the best results in which the player achieve 18 successful runs out of 20.
- According to the experimental analysis, we can conclude the rate of success mostly depends on the size of the maze. It is clear that if the size of the maze is bigger, the player gets larger space to move by maintaining the social distance from the pedestrians.
- The number of steps required to reach the goal is also comparatively larger for the bigger maze.
- There are not many outliers in terms of the steps.

References & Resources

- Nguyen, Cong T., et al. "Enabling and Emerging Technologies for Social Distancing: A Comprehensive Survey." arXiv preprint arXiv:2005.02816 (2020).
- “Pycolab.” GitHub, GitHub, Inc., 2017, github.com/deepmind/pycolab.

Q&A

These questions are pasted directly from the livestream chatbox.

- Q: 1. The last picture in your presentation, what is each rectangle means? What is the orange one means?2. As you mention in the beginning, you want to do the simulation; however, the demo you show is not a simulation. Could you show the simulation?
  A: 1. Each rectangle represents a pedestrian or a player depending on the colors that separate them. The orange one is a special pedestrian where a player can violate social distance with it because they are wearing a mask. 2. Shown in the project ~
- Q: Is this practical or just a game?
  A: This is more of a “education” type of simulation that just simply shows the high risk and easy contamination that spreads COVID-19 if simple social distancing suggestions are violated. If a person is to come within 1.5m of another, then a mask could help tremendously.

Q&A

- Q: What is the relation between this topic and social distance?
  A: It is a simulation to show how movement of a cluster of people can still be at least respected by a person with social distancing.
- Q: Is it possible to use this at real world?
  A: The goal of this simulation is not to “use” it, but more of a visualizer and an education opportunity.
- Q: so all the agents are constantly moving right? Is there a GUI to show it?
  A: There is indeed a GUI showing how each agent moves in its own unique way, with the player, our main agent, prioritizing moving right and up to get to the goal ASAP.
- Q: How do you sure it is optimal?
  A: Our main agent, the player will only move either up or right towards the goal since those are most optimal moves. It will only move in other directions or stay in place if there is a wall or the next state would result in a violation of social distancing.

Q&A

- Q: Can you explain how do you divide the map?
  A: We have different maps available and in terms of “divide”, we “divide” them by separating portions of the map with walls that the agent will need to work around.