SYMUA — A simple survival guide

version #

The way is lit. The path is clear.
We require only the strength to follow it.

Assistants C/Unix 2023 <assistants@tickets.assistants.epita.fr>

Contents

*https://intra.forge.epita.fr

1 The group

Our Group is formed of 6 SCIA students:

- Alexandre Lemonnier
- Adrien Houpert
- Victor Simonin
- Baptiste Bourdet
- Alexandre Rulleau
- Sarah Gutierez

2 The project

2.1 Introduction

A multi-agent system is a type of software that enables multiple agents, or autonomous entities, to work together to achieve a common goal. These agents are typically designed to be intelligent and able to make decisions on their own, using a variety of techniques such as machine learning and artificial intelligence. Multi-agent systems can be used in a wide range of applications, including robotics, traffic control, and environmental monitoring.

There are many challenges associated with designing and implementing multi-agent systems. For example, coordinating the actions of multiple agents can be difficult, and there may be conflicts between agents that need to be solved. Additionally, it can be difficult to ensure that the agents are working towards the common goal and not pursuing their interests.

Despite these challenges, multi-agent systems have the potential to revolutionize a wide range of industries, from transportation and logistics to healthcare and finance. By enabling multiple agents to work together and make decisions independently, these systems can help to solve complex problems and improve our daily lives.

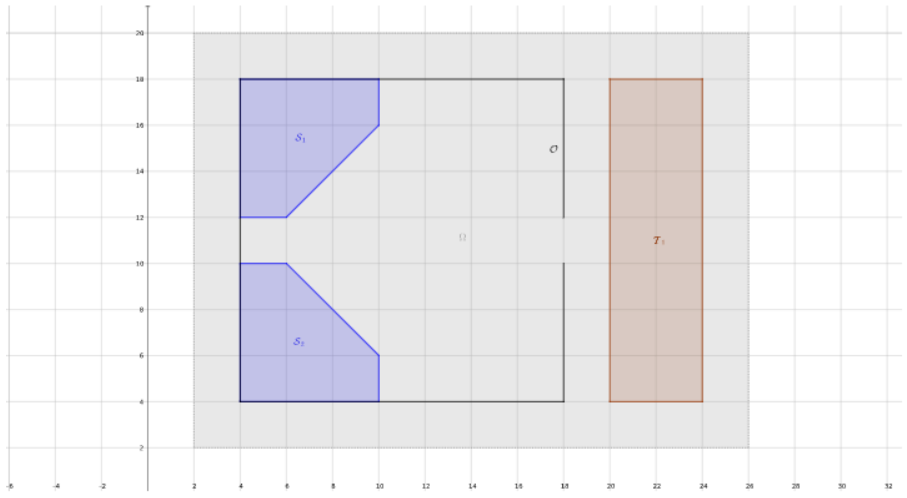
For our project, we have chosen to represent a pedestrian flow simulation. We choose this project due to the recent events that happened in Seoul where people have been crushed to death during Halloween at Itaewon. Also, we decided to test our agent model on the heavy pedestrian traffic of Shibuya crossing to analyze at which point this place is too crowded to circulate or even survive.

To ease our task we started from the crowddynamics project which already implements the engine we need. Therefore, we started this project and implemented our idea by adding new agent behaviors and maps, we also fixed pre-existing bugs.

2.2 Crowddynamics

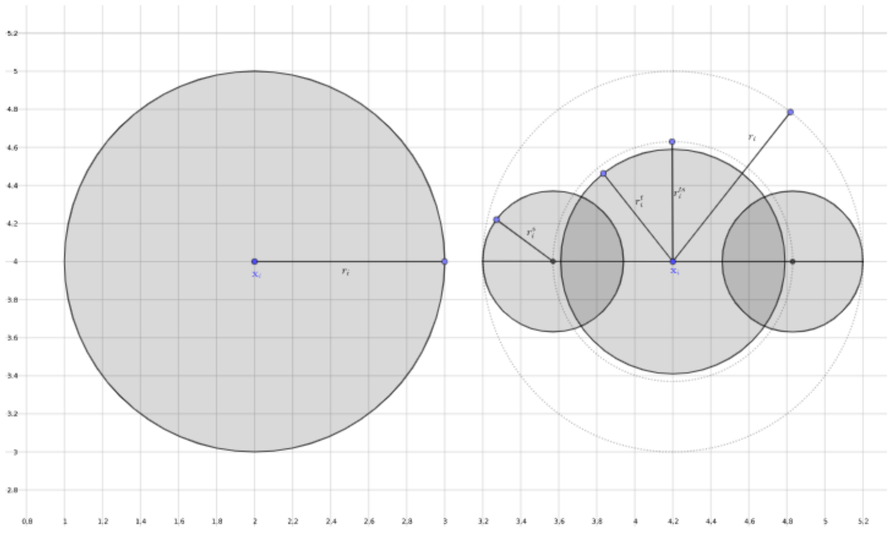
2.2.1 Presentation

The *crowddynamics* project provides us with a whole engine for our simulation and a basic example. To be precise it provides us with a proper graphical interface to observe our simulation.



Example of a field with two sources, one target, obstacles and no agents. Link to Geogebra figure.

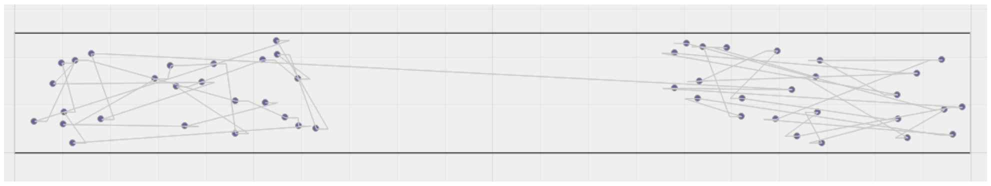
It also provides us with a code interface to properly spawn agents and move them. Two types of agents are available: circular or three-circle agents, we decided to use the circular ones to simplify the simulation.



Circular and three-circle agent models. Link to Geogebra figure.

2.2.2 Cleaning the project

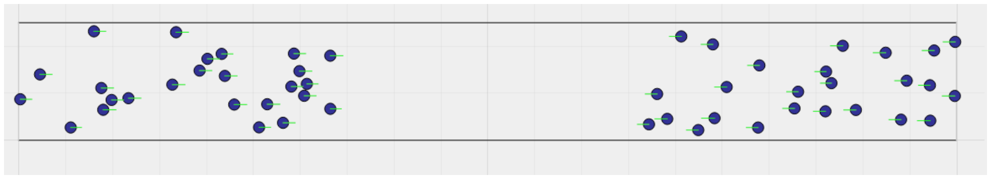
We started by trying to make the initial emph{crowddynamics} project works properly, unlike what is displayed on Github. Here is the initial state we had:



All points were linked with everyone with lines instead of having one for every agent that points to their respective direction.

Also, the points only represented the center of agents and not their actual hit boxes, so we had no graphical validation that our program worked well.

So after cleaning the code and the aforementioned bugs we had the following:



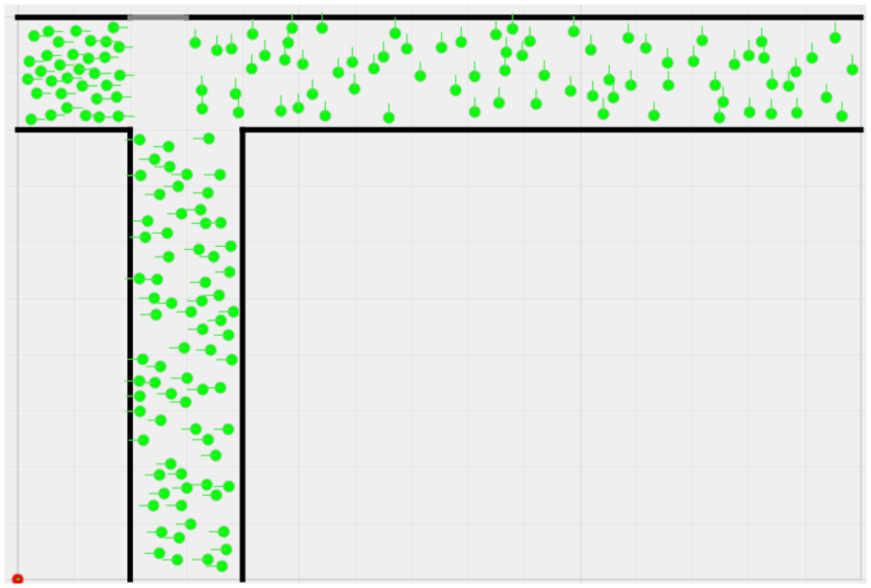
We had also problems with the agents. Once reached their targets they were not disappearing. So we had to make sure they diseapeared once they reached their target for them not to be stacked at the end.

2.3 Agent behavior

To properly simulate this case, we decided to simulate continuous pedestrian traffic. Therefore we decided to select specific start and destination points, then we regularly apply the following behavior:

- Select a random start and destination points
- Spawn an agent at the start position and assign it a random direction
- Once the agent reaches his destination despawn him

Furthermore, to properly imitate the event in Seoul we created another behavior. If an agent is surrounded by too many other agents, he slowly loses his life until he dies.



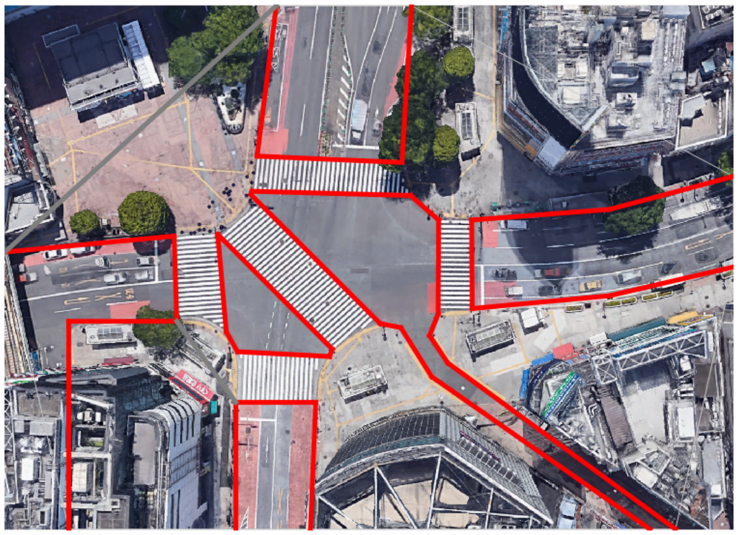
Losing life impacts the behavior of the pedestrian, by adding a temporary target in the opposite way of their initial target, making them try to go back and avoid dying.

2.4 Creating the Shibuya representation

In this part our objectives were to recreate an environment in which the crowd flow could be problematic. Shibuya is a famous square where everyday you can witness massive crowds take the crossroads. By recreating this famous place we believed we could better understand the behavior of people in stressful situations like this.

To properly visualize our idea we had to recreate the Shibuya square into our engine. To do this we imported an image of the square and put it in background.

Once that was done, all that was left to do was create the map borders to prevent our agents to walk on undesired areas. To do this we used polygons from the shapely library. Here is the final results:



After a few simulations and after further reflection we saw that we could not have the same problem as the Itaewon street. The square is a open space and people could still overflow on the roads to avoid being crushed. So we decided to directly simulate a street that looks like the Itaewon street to simulate what happened during Halloween.

2.5 Simulation of a street like Itaewon street

Then we created a simulation to represent the case that happened in Itaewon. The goal of this example was to force our agents to be in a situation where they are forced to be stuck at a moment in a little street and observe if our model was working or not.

To be more accurate, we implemented a new behavior: panic. When an agent is near death, he starts panicking and tries to go in the opposite direction. That way our agents have more realistic behavior.

We observed that people mainly got stuck at the intersection where agents come from multiple directions and collide. We observed that this position is the central point where the agents die, despite the panic system.

2.6 Conclusion

With our multiple experiments, we can conclude that the incident of Itaewon cannot be replicated at the Shibuya crossing as people will just walk on the road if space is needed. Therefore we can deduce that this could only happen in a highly crowded and closed place.

We then replicated the Itaewon mall street where the accident happened and we observed the same behavior that happened in real life. We then observed that when agents got stuck alongside too many agents going in the same direction they start to die. Even trying to escape from this situation by going in the absolute direction is not enough as other agents are blocking the way. Therefore we can say that this simulation replicates accurately what happened in real life.

2.6.1 Instructions

Here are the instructions to launch our project. You first need to install conda on your computer. Then you can run the following commands:

```
cd src/qtgui
conda create --name symua python=3.8
conda activate symua
pip install -r require.txt
conda install -c conda-forge scikit-fmm
conda install shapely
python cli.py run
```

The last command will launch the graphical interface, you can then:

- Click on *open*
- Open the file *init.py*
- Click on *initialize simulation* then *start*

The way is lit. The path is clear. We require only the strength to follow it.