Evacuation Simulation: EvacX

Members: Diana Lysova, Andrew Chen, Noah Kochavi, Conrad French

Context

In our modern world, we unfortunately have to deal with many unprecedented events that may happen in our lives. From flooding in coastal cities, tornadoes in the midwest, to even the horrible intruder threats in our schools. In many cases, local governments have been caught off-guard, with many not even mounting a proper response until hours later. Denizens of affected or at risk communities may wonder, "is there anything that can be done?"

Our group says that there is. EvacX is a simulator that will be able to mount a much better response than past measures. In this project, we will use active intruder threats as an example to display the suite of functionality EvacX is capable of. Although, EvacX should ideally be able to mount a response against a variety of threats.

Goal

The goal of the simulation is to guide as many victims to the exit nodes as possible. Intruders will be present and will try to eliminate as many victims as possible, so it is important for the system to make the best decisions in both the short and long term.

To achieve this we will be using several graph algorithms as well as levels of optimization to allow the system to run in an efficient and timely manner

Floor-Plan

To represent the floor-plan of the establishment we were running the simulation in, we decided on a **Weighted Di-Graph**.

This graph would consist of rooms and hallways, which would be represented as nodes. The edges within this graph represent the connections between various hallways and rooms. In addition, there are special nodes labeled "exit". These exit nodes are the destination of all victims as they would be safe upon reaching the nodes.

Algorithms

1. **A* Search** finds the shortest path from two vertices. It differs from Dijkstra's in that it does not bother calculating vertices farther from the destination. Every iteration tends to bring in vertices that are closer to the destination.

2. **Edmonds Karp** find the maximum amount of flow from a source to a sink (destination). In this context, the algorithm is finding the maximum amount of people that can be evacuated from a source to the exits.

3. **K-Means** is an unsupervised machine learning algorithms that classifies inputs based on some metrics. In this context, the metrics are the proximity from the intruder and the location of each victim. From these metrics, K-Means will generate *clusters* of the danger levels of victims. The higher a danger level of a victim, the more priority the system will give to them.

4. **Quad-Tree** is a tree data structure whose nodes are positions within a display. A node is only present if there is at least one item within a quadrant. In this context, only quadrants that have victims or the intruder and stored, saving memory and allowing for fast collision detection.

Phase I

The main goals of this phase consist of:

1. Create a GitHub Repository & Invite all group members
2. Read & Learn Godot & GDScript and create a project
3. Load in assets needed for the simulator
4. Investigate the Algorithms required for the simulation