|  |
| --- |
|  |
| Path Planning |
| CP468 Term Project |
|  |
| **Jake Buller, Michael Dougall, and Travis Miehm** |
|  |

|  |
| --- |
|  |

# Project Description

Consider a Museum room that is patrolled by N robots at night. At a pre-determined time, the robots are supposed to rendezvous at a given point R in the room. The robots move inside the room, and the room contains obstacles, such as chairs and benches for the visitors, paintings, sculptures etc. The robots are supposed to know the location of the obstacles in the room.

Implement an A\*-based algorithm to compute the path of each robot, from its initial position to the given rendezvous point R.

# Design Decisions

When beginning to tackle the problem presented our first decision was which programming language to use. For this project we decided to use C#. It was chosen because it is a modern language and it is used in the industry for producing release-quality software. Early on it was also decided that we would use GitHub during the project. It helped in the case that a problem arose as we could go back to previous versions of code and determine the changes that led to the problem. It also ensured that everyone was using the same version of the code when they went to work on it.

Next we needed to decide how the various elements of the problem would be represented programmatically. The map was stored as two dictionaries, one representing the vertices and one representing the edges between these vertices. The vertex dictionary takes the vertex label as a key and returns a Point2D datastructure, containing the x and y coordinate of the vertex in the map. The edge dictionary takes the starting vertex label, fromLocation, and will then use a second dictionary and the intended vertex, toLocation, and will return the distance between these two vertices. This datastructure can also be used to determine if connections exist and allows for the creation and modification of the map.

The robots and rendezvous point were represented as Point2Ds, a previously mentioned datastructure. Since there are multiple robots they were stored as a List of Point2Ds. The robot Point2Ds are able to change their location as the robots would, enabling the search to be executed.

Since we were told to make the program A\*-based we required a heuristic function to implement the tree-search. We chose to use a direct path heuristic function, meaning the function will return the straight line distance between two points on the map. This made sense as our robots would only be given four directions that they could move in, so moving as close to the goal as possible would always result in the most efficient path.

Once the main components of the program were created and simple test were executed successfully a full scale implementation of the solution was made. This then needed to be optimized in order to produce reasonably timed results. In order to reduce some of the computation time it was decided that the path finding for each robot would be executed in parallel on separate threads.

Next the map was created in such a way that it did not allow the robots to move “see” beyond of the barriers of the map. This was done by creating the mapping of the room like a graph of paths, similar to a road map. The only “roads” the robot can travel down are those that are given to them. This meant that the robots would simply move to where they were told they can go and not have to figure out where they cannot.

# Execution Instructions

# Test Results