Summary

Finding an unobstructed route from one game world location to another is a fundamental part of the Artificial Intelligence simulation for non-player characters in computer games. This challenge is to design and implement an algorithm for finding a route for a game character to row around an island to reach their home ship. The algorithm must deal with various island shapes, and the challenge will be scored based on how short a route is generated for each of a set of test islands.

Details

Fred the pirate has been exploring a new island looking for treasure and now finds himself floating in his little rowboat in the baking Caribbean sun, thirsty and tired. He needs to get back to his ship as quickly as possible so you need to find the fastest route for him to take before the heat addles his brain.

You have the following information to plan his route:

His current location, his ship's location and a set of points marking the limit of the navigable space around a single island, these are all indicated as red crosses in the image above. All coordinates will be specified as two floating point numbers X and Y, where X indicates the distance from the left edge of the map and Y indicates the distance from the top edge of the map.

The points marking the navigable space around the island will be supplied in clockwise order, indicating the vertices of an n-gon.

The map data will be supplied in an XML file with a documented structure, and the results must be supplied in an XML file in a specified structure. Examples will be provided.

For simplicity, you should ignore the size of the rowboat and ship, treat them as just point locations.

Fred cannot carry his rowboat across land, so you should not plan a route that crosses the island.

Challenge Day

You will be supplied with:

- An XML file structure definition and some example files.
- A set of test case files to use with your implementation.

You should return:

- A set of generated routes, one for each test case, in XML files.

Required Skills & Knowledge (or advanced research)

Ability to rapidly implement your algorithm in a suitable programming language.

Knowledge of suitable data structures to store the input and output data, and intermediate data as required by your algorithm.

Fundamentals of 2D mathematics, including vectors, length calculation, dot products and their uses, and methods of detecting intersections of 2D lines and other geometry.

Graph theory and path optimisation methods.

XML file reading and writing.

Resources

Computers as necessary to implement the algorithm you design.

Suitable programming tools to edit and execute your implementation.

Libraries as required for easing implementation, for example XML parser/generators and 2D plotting libraries if you want to visualise the input and output data.

Approach

There could be several possible algorithms that will produce a route, some will produce better routes than others but be more complex to implement. Perhaps implementing a simpler algorithm first and then iterating on the algorithm will ensure that some results will always be available to submit for the challenge.

Visualisation of the data is not essential but may be useful for tuning your algorithm.