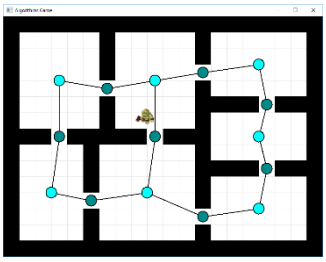
# Assignment 3: Graph searching

In this third and last assignment you have to implement search algorithms both iteratively and recursively, so that Morc-da-Orc can finally explore his lair completely.

***Continue in your code from assignment 2.*** Again, a walkthrough for new parts of the code will be provided during the lectures, demonstrating all the visualization options you have to (/can) use.

---

### *3.1 'Sufficient' requirements:*



- You have implemented a recursive shortest path algorithm.
- You have implemented an iterative shortest path algorithm (BFS).
- When you click on a node, Morc follows the shortest path to that node using whichever pathfinder was passed to it.
  (Shortest path here means: least number of nodes).

---

### *3.2 'Good' requirements:*

- You have implemented all the 'Sufficient' requirements.
- You have adapted your BFS algorithm so that the shortest path is now based on distance, not on node count (Dijkstra).
- It is possible to enable/disable nodes in the graph, to exclude them from the path finding process. (Like locking a door.)
- You have an on-screen indicator indicating whether the dungeon is fully connected, which is updated each time you enable/disable a node.

---

### *3.3 'Excellent' requirements:*

- You have implemented all the 'Good' requirements.
- You have adapted your Dijkstra algorithm to use a heuristic to correctly guide the search (A*) and you can clearly demonstrate the difference in node coverage with respect to BFS and Dijkstra (hint: use the low level graph from assignment 2).