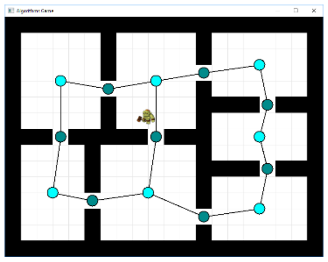
Assignment 2: Graphs, tiles & agents

In this second assignment you have to convert your Dungeon into graphs and tiles, learning how intermediate forms of data might help you solve a more difficult problem. In addition, you will practice general list manipulation a bit more by moving Morc-da-Orc through its new lair.

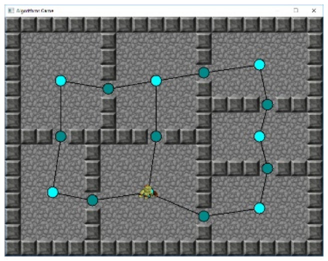
Continue in your code from assignment 1. Again, a walkthrough for new parts of the code will be provided during the lectures, demonstrating all the visualization options you have to (/can) use. This code only has to work with the 'sufficient' requirements from assignment 1.

2.1 'Sufficient' requirements:



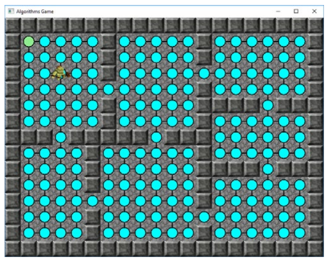
- Your graph replicates the dungeon structure with a node for every room and door with connections between them.
- Make sure that Morc queues clicked nodes and visits them in order. Morc is not allowed to go 'off graph' anymore, so make sure you ignore the clicked nodes that would cause this.

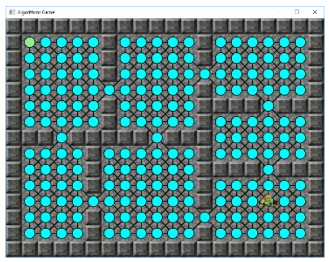
2.2 'Good' requirements:



- You have implemented all the 'Sufficient' requirements.
- Your tiled view replaces all black and white lines with the correct tiles.
- Morc will now only 'listen' to a node click when he is not walking. As soon as Morc detects a node click, he will start moving around on the node graph randomly until the queued node is reached, with two restrictions:
  - Morc should not go back to the node it just came from, unless that was the only option.
  - In case Morc can reach the queued node from its current node directly, it should do so.

2.3 'Excellent' requirements:





- You have implemented all the 'Good' requirements.
- Your graph replicates the dungeon structure with a node for every walkable 'tile'. See sample images on the right, both implementations are ok.

Optional challenge: everything works with the 'Excellent' version of assignment 1