Cozy Caves

A D&D Dungeon Generator

Hallway Generation Module

Written By
Gavin Lim

Reviewed By
Carlo Cigaral

Hallway Generation Methods

What is the purpose of the Hallway Generation Module?

Our Dungeon Generation Module uses Binary Space Partitioning to generate many subdivisions which potential rooms can be generated in. However, there is no link between the different subdivisions. The Hallway Generation module will be responsible for providing the logic for this.

Points to Consider:

- Will all the rooms be square?
- Will all the rooms connect via the centre of an edge or can it be off centred?
- What will the width of the hallway be?
- How will we deal with L shape hallways?
- How do we determine which rooms connect?
- All rooms must be reachable.
- Will rooms have multiple connections?
- Dungeons should have some creativity (longer hallways/different shapes)
- What if rooms are already touching?
- Could we connect hallways together?

Basic Overview of Code Structure:

Input:

- List of rooms
    - Co-ordinates in a 2D space consisting of the (x,y) of its most top left space and the width and height of the room. Further information on the rooms may be found out from the Room Generation Module.

Output:

- List of objects that represent hallways.
    - Object fields TBD

Methods of Hallway Generation:

Using Pathfinding Algorithms:

Using algorithms such as BFS/DFS or A star could be an efficient way to generate hallways between rooms. It involves traversing between the space between the rooms until finding another room. Then we can create a hallway between the two rooms and proceed traversing until it has been at every room. We would need to pick a starting room and then perform the algorithm.

Pros:

- Efficient.
- Will work 100% of the time - guaranteed that all rooms will connect.
- Results in a tree like structure which will be helpful for later processing

Cons:

- Hallways will likely be to their closest neighbour - no longer hallways are likely to exist.
- Paths may not come out in nice shapes.
- Being efficient may give the dungeon a stale and boring feel - lacks creativity in dungeon layout.
- Will result in only one hallway per room which also lacks creativity.

Finding neighbours using raycasting:

This method involves travelling in a straight line from the edges of a room and finding other rooms on the same x or y axis as the origin room. This process would repeat on all the rooms until they have found all of their neighbours. We can then connect these rooms together.

Pros:

- Allows the rooms to have multiple connections.
- Has some logical sense having rooms connect when they are on the same x/y.

Cons:

- May not connect all the rooms
- Only straight line hallways - lacks creativity

Room Pairing:

This method involves pairing the rooms up randomly and generating a hallway between them. Then, we pair all the pairs and we continue doing this until we have one big connection. We will need to handle the edge case of having an odd number of groups in this method.

Pros:

- Will generate more random hallways - more creativity
- All the rooms will connect with each other
- Rooms will likely have multiple connections - more creativity

Cons:

- Hallways may be very long
- May result in a very messy infrastructure

Delaunay Triangulation + Minimum Spanning Tree:

This method uses the Delaunay Triangulation concept to create links between the various rooms. We do this by picking the midpoint of a room and run the algorithm. This will result in a lot of connections however, so we cut this down by using a minimum spanning tree.

Pros:

- All rooms are connected.
- Generates a lot of various possible paths with Delaunay Triangulation.
- Generates a graph-like data structure which is useful for processing later.

Cons:

- Minimum Spanning Tree may be a boring looking dungeon. Singular Path.
- Complex

Tentative Decision:

We will use the Delaunay Triangulation + Minimum Spanning Tree method. It makes it easy to guarantee connections and generates a lot of different potential paths if we decide to explore various options. The complexity will be fine as there are libraries available to use to perform calculations. In regard to the boring path, we can add back some edges from the triangulation and/or create our own connections to add more flavour to the generated hallways.