Map Generation Documentation

I broke the map generation process down into a few levels of abstraction.

Top - the map you see on-screen

Middle - an array of integers

Lowest - we have rooms and doors

Not actual rooms and doors, these rooms and doors are structs that denote all the stuff you see onscreen. A room has a seed, a position, a size, the stuff it contains, and their positions. Doors just have a position and an orientation (north-south, east-west).

First we generate a list of rooms. For each room, we perform the following process:

- **Seed** -> generate a number between 0 and 1. We use this to generate other stuff but it's mostly in the background.

- **Size** -> generate a width and height for the room

- **Place** -> I wanted all the rooms to be closely connected; there are no hallways because that leads to some game issues I wanted to avoid - namely, having chasers follow you down a straight line, being shot at by things that you can't avoid, and having lulls in the action. To ensure that this new room will be connected to another, we pick one of our rooms so far, pick a random side on that room, and attempt to place this room there. If it collides with or gets too close to any other room, we pick a new room to connect it to and try again. If this fails too many times - I set it to around 1000 max failures per room - we throw out that room entirely. If it succeeds, we add this to our list of rooms

We do this process until we've hit our max number of rooms or we've failed to place too many rooms in a row. Once that happens, we start the next step, which is populating the rooms. This is really just about the enemies and health merchants, not the upgrades or boss rooms.

- **Fill** -> get the area of the room and generate a number based on it. We have upper and lower exponential bounds on this so we could generate something with a lot more or a lot less enemies for different difficulty settings. Then we get a random number between 0 and 1 and that number tells us what type of entity it'll be - a chaser, a shooter, a charger, a mine layer, or a heal merchant.

- **PlaceEntities** -> Pick a spot in the room, make sure it doesn't collide with anything else, and put this entity there.

Now we've got all our rooms, they're all connected spatially, but they need actual connectors between them. For that, we have doors. Basically we just compare all the rooms to see what's lined up properly for a door, and if they're lined up properly and they overlap enough, we add a door there. That's pretty much it.

At this point, we take our rooms and our doors and we convert them into ints on a two-dimensional array of integers. Rooms are broken down into floor, walls, and corners. Doors are just dependent on whether they're horizontal or vertical.

Once that's done we need spots for the curse, the upgrade, and the ladder down to the boss room. For the curse and upgrade I pick 2 rooms at random, excluding the starting room, and place them there. I don't do any checks to ensure they end up in different rooms - the idea is that normally we'd be generating bigger maps so the odds of them ending up in the same place would be low but act as a fun surprise. If we had a bigger number of upgrades I'd probably be making sure they were all in unique rooms. Finally for the ladder to the boss room, I just picked whatever room was furthest from the starting room to ensure you'd have a decent chance of finding the upgrades before getting there.

At the end of this process, like I said, we've got a 2 dimensional array of integers. Now all we do is convert all those integers into actual tiles on the map or instances of entities and the map is done.