Lee Taber

My proc-gen dungeon is fairly simplistic as of now. It generates in the following manner. First, it generates a starting tile in the center of the room. This is always three by three squares and always directly in the center of the room. Next, it starts to add in some rooms. A room has random sizes from 2x2 to 3x3 and starts to place these from the center. The rooms are places by starting at the first room in an arraylist of rooms and checking if there is a room north of it. If there is, then it checks east, and so on around that room. If all those spaces are full, it then checks around the next room in the list, and around the room after that, until all available positions are filled. Once the rooms are in place we add the start and end tiles, which are represented with stairs going up and down. The stairs are always placed “randomly” where randomly means: somewhat far from one another, not on the edges of the map, and in different positions every time. Once those are placed, I add some random noise to help break up the blocky rooms. For every tile a d10 is rolled, if it is a ten, walls are switched into floors, and floors are switched into walls. Next items, in this case keys, are added to the dungeon. The items are added truly randomly, which means that some dungeon levels are a bit unsatisfying. It just finds random floor tiles in the dungeon and switches them into keys. Finally, we have a dungeon but the way it was built doesn’t guarantee there is a path to each item. The last step is to start from the starting staircase and move towards each item in the list (the keys and the exit). If there is a wall in the way, it is switched to a floor. In this way, each floor is solvable, but unfortunately not always in a very interesting way. Once the dungeon is generated, several more are generated as well, so the player must navigate 5 levels to reach the end.

The original design I was looking at was much more complicated than what I ended up actually doing. After finding an example of proc-gen dungeon code online, I thought it would be cool to make rooms and corridors that went from each room to one another. This would make much more complex shapes, but I wasn’t able to wrap my head around it in time for this project. I’ve had a chance to talk to some other people in the CM department who do more work with proc-gen stuff and I was able to get some advice on where to look at improving my work. For example, I talked to Batu who had some advice for simple and slightly more complex pathfinding strategies. I ended up going with one of the simpler ones, but I could probably figure out a more complex one with some more time. This was sort of the path I ended up taking while making this project. I saw and became aware of more complex constructions that would lead to more interesting results, but I didn’t have the knowledge or time to fully explore those other ways of generation.

I think one big thing that I learned from this project was to not get to focused on one style of doing things too early. I focused on one way of developing the dungeon (which it turns out is kind of bottom-up), when other ways might have been used to better effect in the time that I had. For example, I was recently shown an example of how Splunky generates its high-level path first using simple numbers. I could have probably developed something similar with my tile/room-based array version which would have been an interesting route to explore instead of making a space and then carving a path through it. Overall this was a good experience and I enjoyed the processes of designing a proc-gen dungeon. It makes you really appreciate the work professional developers go into making their algorithms generate fun and interesting content for people to play with. If I were to develop this further I would really try to “find the fun”, but I think this is acceptable currently as a student project.