

COM S 327, Spring 2019

Roguelike Roadmap

Students who intend to develop on their own code base through the entire project can benefit from knowing where the project will go. Here is a roadmap plotting the general direction of our development. See the individual project specifications (as they are posted) for detailed descriptions.

- 1.01** Generate a dungeon. The dungeon consists of a set of rooms, and corridors that connect them. It also contains rock, which is all areas which are neither room nor corridor, and stairs, which don't function yet and can be walked over like floor. The dungeon must be fully connected. 1.02 will require a dynamically allocated room array, so you may want to include one here. 1.03 will require that the rock cells have hardnesses.
- 1.02** Write dungeons to disc and read them back. This will be the most rigidly specified assignment of the semester. You must write your dungeons to a file that can be loaded by any student's (correct) dungeon reader. Your dungeon reader must be able to read any student's dungeon.
- 1.03** Calculate shortest paths from all points in the dungeon to the PC (player character). Some NPCs (non-player characters) can tunnel through rock with efficiency dependent upon its hardness, so two different distance maps are required: One for non-tunneling monsters, and one for tunneling monsters. The latter requires Dijkstra's Algorithm. The former could use either Dijkstra or DFS.
- 1.04** Add the PC and randomly generated NPCs to the dungeon. Make the game "play" by itself as the NPCs seek out the PC. There is no user input. You may optionally add AI code to control the PC. Game ends when either the PC or all NPCs are dead. A character dies when another character moves into the dungeon cell that it occupies. NPCs will have multiple possible characteristics, so that they don't all behave the same way.
- 1.05** Add a curses interface to the game. PC is now controlled by the player.
- 1.06** First assignment using C++. The PC and NPC structures must be changed to classes. If you have a character structure that these are based on (as in the instructors code), then you must use inheritance here. Add fog of war, requiring a remembered terrain map for the PC. All code is to be changed to C++ (rename the files and compile with g++; the heap may optionally remain as a C file). All new code for the rest of the project is written in C++.
- 1.07** Given a specification for monster and object descriptions, you will implement a parser to load these descriptions into the game. Descriptions will be text. You will only be required to write the monster parser. Instructor will supply an object parser. Students writing their own code may use the instructor's standalone object parser or write their own.
- 1.08** Use the parsed monster and object descriptions to load monsters and objects into the game. No more random monsters. New, specified monsters replace random monsters. Monsters have hit points, but we won't use them yet; still have the combat semantics from 1.04. Objects are placed on the floor, but no functionality to pick up or use them yet.
- 1.09** Implement object pick up and equipment. Update combat semantics to use damage dice and hitpoints.