Original Request:

We would like a general update on how your plan has changed (now that you are trying to execute it), as well as what your status is (how much progress has been made). Additionally, we would like you to answer these specific questions:

1. Your group had an ambitious design. While you had a great concept, it was less clear what was going to happen to the player: what kinds of things would appear, what options there would be, how the "puzzles" and weapons would work, ... Please describe how it has become more concrete (in terms of what the player is going to see and do, what are the enemies going to do, why its going to be fun, ...).
(note: our goal isn't necessarily to understand your design, but more to have confidence that you've worked it out)

2. Your group has an ambitious goal in terms of look and feel (although you seem to have hit the ground running there!). Your initial success inspired confidence that you will be able to pull something off. Our concern is how you will be able to put the pieces together. Can you give us some sense of how you are going to do the "software engineering" or architecture? (in a way that will fit in with the dynamic programming environment, and distributed team.

Additionally: If you are having (or foresee) any issues or problems where we might be able to help, please let us know. (this can come direct from any group member - especially if there are personnel issues).

---

1. The goal of the game is to complete the current dungeon level by finding and solving a puzzle room. The player will move around the dungeon trying to find this puzzle room. Once the puzzle room is solved the player will be able to go to the exit room completing the current level. The dungeon will be randomly generated with key rooms place at random in the dungeon after the fact. There are 3 rooms that will be spawned regardless of how big the dungeon is (spawn room, puzzle room, exit room). How they will be connected will be random. All rooms will be base on a 2d grid structure.

Players will have the following
- Verbs:
  - Move
  - Change the current light to blue (pausing enemies and changing things, such as textures, in the current room)
  - Draw on the ground with chalk/salt (creating obstacles that enemies cannot pass through)
  - Push certain objects.
- Resources:
  - Sanity: Sanity will server as both life, and ammunition for the ability to turn light blue.
  - Chalk/salt: Players will be able to collect salt/chalk that is randomly scattered throughout the dungeon.

In order to simplify movement, we are planning represent each room as a 2d grid. Drawing with chalk is done by moving to a point in the room and pressing a button. The current area that the player is standing over will then be turned into a chalk square.

Enemies
- Attack:

Enemies will reduce sanity once they occupy the same grid cell as the player.

Proposed movement and AI:

Move towards the player.

Move in a preset pattern.

Move along the grid randomly changing direction after a set number of squares

Sample puzzles include:

Sun Dial - clock puzzle

"the door opens at midnight is written on the floor" There is an obelisk and you have to cast blue light to make the shadow point to 12.

Code cracking - shadows are casted (with blue light) and gives you hints to a riddle you have to solve (Scribble the answer onto the floor with the chalk). Example - Silent hill.
Play with idea of casting light onto the back wall and have a certain thing only show in the shadow.

Dumb monster - the monster always walks in a straight line, but you have to lead it to a certain tile on the floor. Say the floor is covered in squares that kill a shadow on contact. you must use your salt to diverge them to safe routes.

Glass floor. Player comes into a room with what seems to be no floor. eon entrance player is told "this is an opening that seems to go on forever". Player cast blue light to see floor tiles - one path to get through.

Phasing blocks - some only appear when blue light is lit and some only when the normal light is lit. Must change the lights in order to pass the different type of blocks.

The player experience will be enjoyable because we will be challenging the player to explore an unknown space while fending off shadow enemies and solving puzzles.

---

2. Our general software engineering approach is outlined in the attached image. This should give you a sense of how we are dividing the work, and how the engine works on a general level. A very simple model for game objects allows people to create new classes easily and then create these objects for the game in one js file. Then, objects refer to each other using global containers for different objects types, if necessary. This is not the cleanest solution, but done is better than perfect. This engine has been used for Alex's project 1 and project 2 and has been refined from repeated use.

A brief explanation of dungeon generation is as follows:

Create a room and determine how many directions the player will be able to move. The number of directions available will have an upper bound of 4, or the number of rooms left (whichever is smaller), and a lower bound of 1(this may be the direction that the room in entered from. The number of directions is determined by Math.random()*#OfRooms.

For each new direction, generate a new room with the inverse of that direction as an available passage. Add these rooms to a stack and decrease the number of rooms left by the number of rooms created.

Continue until you have all the rooms you need.

For development we are using git and github. This has been successful and allows multiple people to develop things very easily. We have divided labor in a way to ensure that conflicts will be minimized. The project can be seen here https://github.com/alangenfeld/cs679.