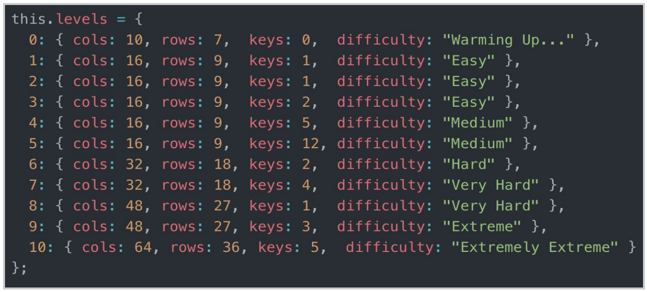
My initial understanding of the project, is to deliver a functional web-based maze game. The overview mentioned that it should be accessible through a website - so my strongest options for language choice are Java, HTML/JS, or a flash game.

I have not made many flash applications, so I most likely won't approach it this way. Limiting my options to either a Java or HTML/JS based game.
Java would be my first choice as it relates to my course more so than HTML/JS does, however this would be my first time creating a Java app for web use. All of the other programs I have written in Java were either simple command line apps, or for use on android systems. So I am not comfortable enough to use it for this particular project. If I was able to create the game as an application for android phones, this would be prefered over the other options.
That leaves me with the decision to create the game using HTML/JS. I have made countless web-based applications/games using HTML/JS, so if I run into any problematic bugs during development, I feel that my knowledge in this area will benefit me exponentially. I will most likely try to draw all elements of the maze on an HTML canvas - the walls can be thin line strokes, and I can use SVG images for the player, keys, exit and possibly a background for the canvas too.

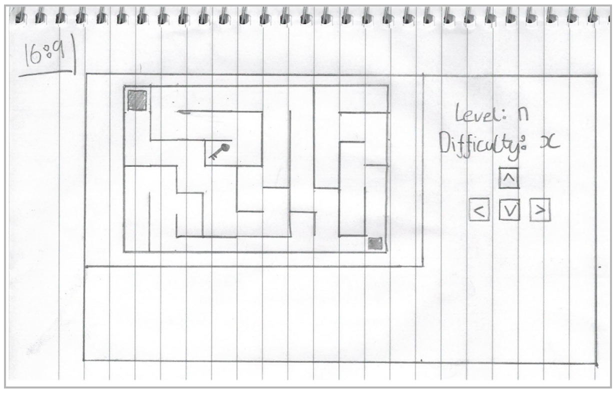
When it comes to creating the maze, I need to use a path finding algorithm. I have created mazes before, and I found that the recursive backtracking algorithm is one of the simplest to implement. I plan on generating an "x*y" 2D array - where x and y represent the number of columns and rows in the maze. I will then be able to run the algorithm on this array, generating a random maze for me to place the player, exit and keys appropriately.

For the actual functionality of the game, I want to create a level system - where the difficulty increases with each level passed. I will probably do this by increasing the size of the maze when needed. I plan to structure the level configuration data in a JSON object, similar to the format below.

I also want there to be some objective other than simply reaching the end. So I want to introduce a 'key' system, where the user must collect all of the keys in the maze before they can unlock the 'exit door'. I think this will aid in the process of increasing the difficulty every level too.

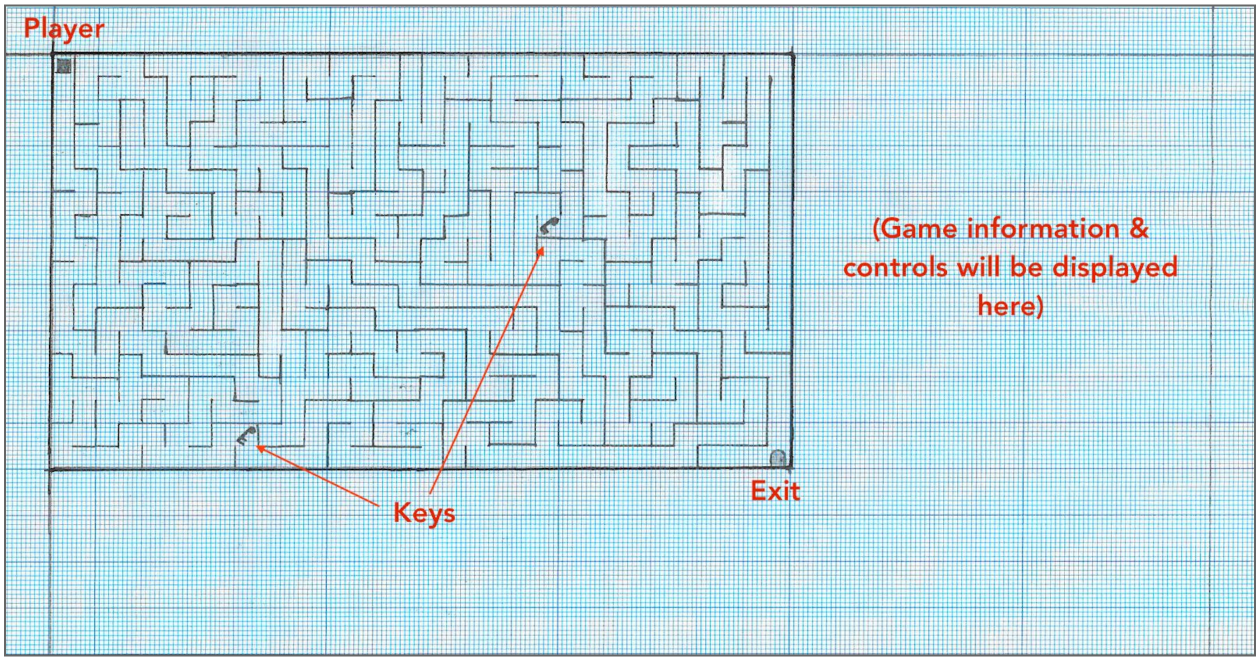


```
this.levels = {
  0: { cols: 10, rows: 7,  keys: 0,  difficulty: "Warming Up..." },
  1: { cols: 16, rows: 9,  keys: 1,  difficulty: "Easy" },
  2: { cols: 16, rows: 9,  keys: 1,  difficulty: "Easy" },
  3: { cols: 16, rows: 9,  keys: 2,  difficulty: "Easy" },
  4: { cols: 16, rows: 9,  keys: 5,  difficulty: "Medium" },
  5: { cols: 16, rows: 9,  keys: 12, difficulty: "Medium" },
  6: { cols: 32, rows: 18, keys: 2,  difficulty: "Hard" },
  7: { cols: 32, rows: 18, keys: 4,  difficulty: "Very Hard" },
  8: { cols: 48, rows: 27, keys: 1,  difficulty: "Very Hard" },
  9: { cols: 48, rows: 27, keys: 3,  difficulty: "Extreme" },
  10: { cols: 64, rows: 36, keys: 5,  difficulty: "Extremely Extreme" }
};
```



The game controls should be designed around receiving user input from a keyboard - so the obvious choice would be to map the arrow keys for directional control of the player. I will probably also map in the same controls to 'WASD' to give the user more than one option.
If necessary I can also add in some buttons to the side of the maze - possibly just 4 directional arrows that the user can click - in the event that they do not have keyboard access.

The example maze that I have drawn (below) is representative of how I want the higher difficulty levels to look. The increased difficulty from more columns, rows and multiple keys to collect, gives the player an extra challenge before completing the level.

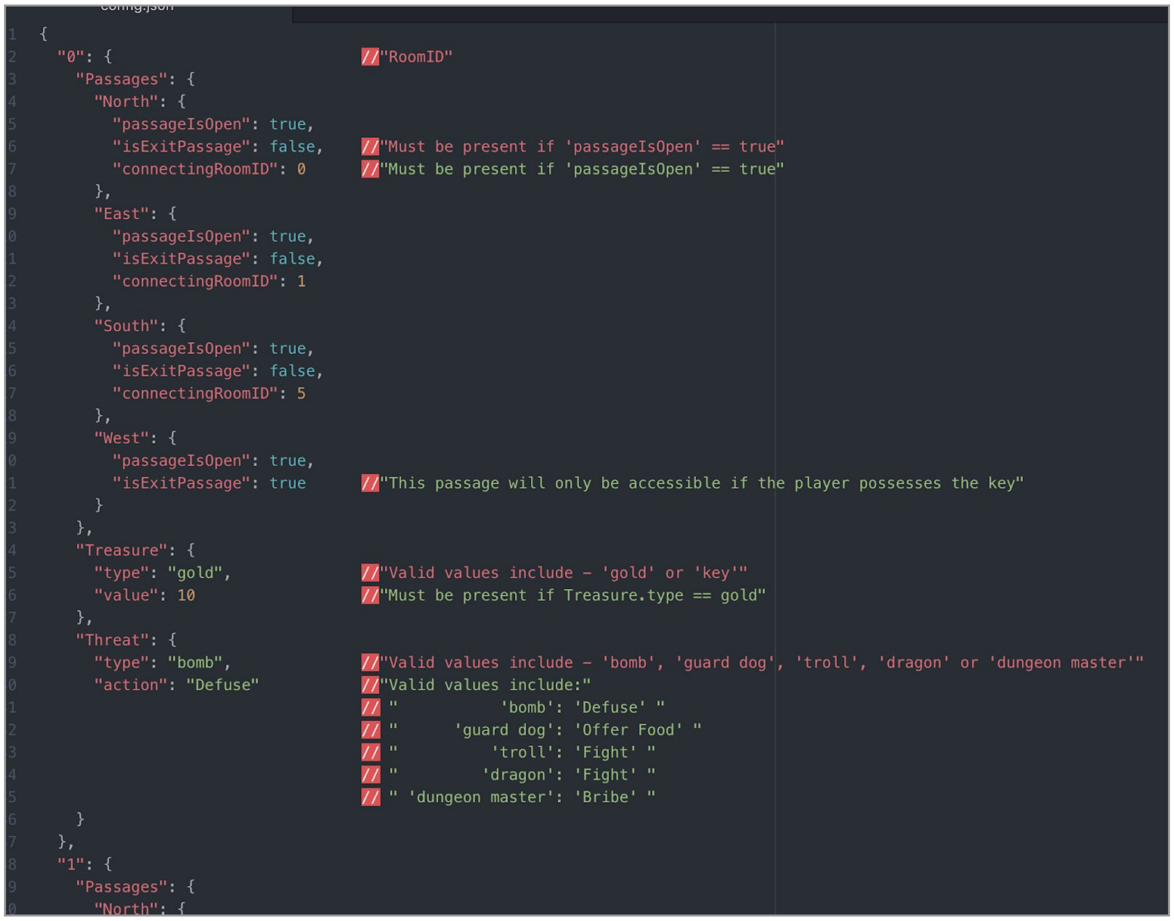


I feel that this gradual increase in difficulty is much needed in a simple maze based game - without it, the game would become very repetitive far too quickly. This particular example would fit the level 6 parameters in the example JSON mentioned previously, consisting of; 32 columns, 18 rows and 2 keys - I would class this as a 'hard' level.

<u>Full Requirements Recieved:</u>

After receiving the full requirements, I now understand how much I have misunderstood the task. I am going to need to completely scrap all plans I previously made. The maze that I designed does not contain rooms - only passages. Also, there are no threats and there is no way to accumulate wealth. Along with the fact that the configuration will also need to be capable of being user defined…

The first thing I have decided to design is the configuration file, as I think that the rest of the game will need to revolve around the format of this data. So based on the game requirements mentioned, I have come up with the following specification. I have added some temporary comments for more information.

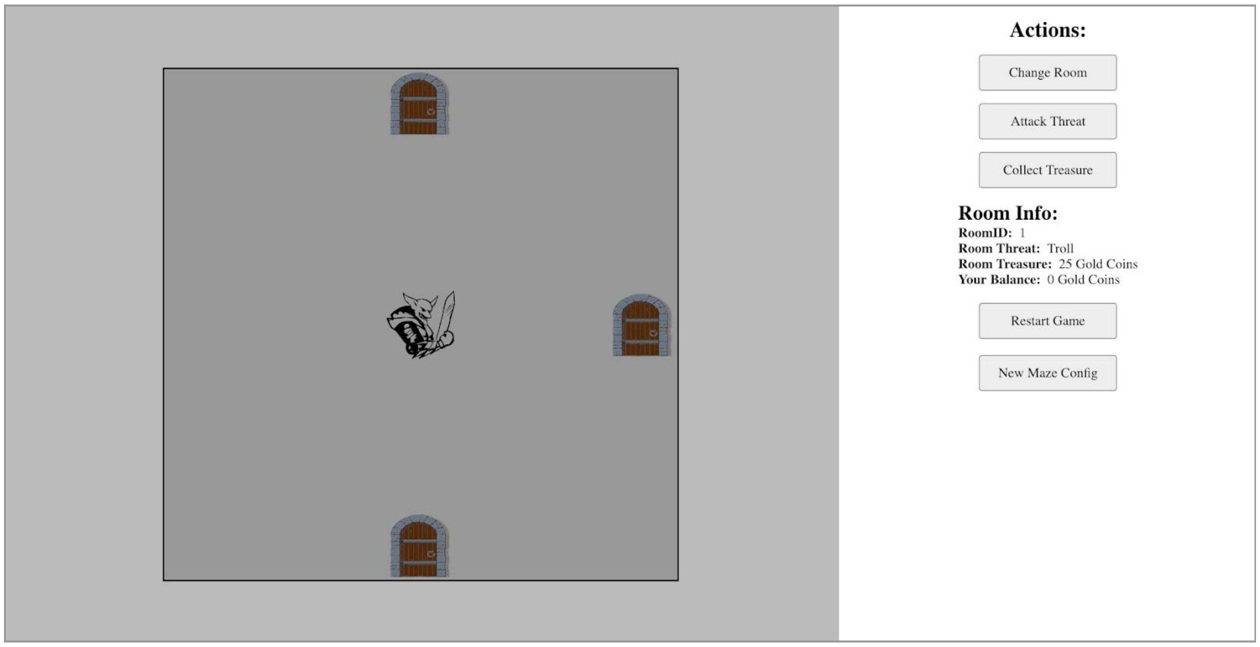


```
{
  "0": {                           //"RoomID"
    "Passages": {
      "North": {
        "passageIsOpen": true,
        "isExitPassage": false,    //"Must be present if 'passageIsOpen' == true"
        "connectingRoomID": 0      //"Must be present if 'passageIsOpen' == true"
      },
      "East": {
        "passageIsOpen": true,
        "isExitPassage": false,
        "connectingRoomID": 1
      },
      "South": {
        "passageIsOpen": true,
        "isExitPassage": false,
        "connectingRoomID": 5
      },
      "West": {
        "passageIsOpen": true,
        "isExitPassage": true      //"This passage will only be accessible if the player possesses the key"
      }
    },
    "Treasure": {
      "type": "gold",              //"Valid values include - 'gold' or 'key'"
      "value": 10                  //"Must be present if Treasure.type == gold"
    },
    "Threat": {
      "type": "bomb",              //"Valid values include - 'bomb', 'guard dog', 'troll', 'dragon' or 'dungeon master'"
      "action": "Defuse"           //"Valid values include:"
                                   // "           'bomb': 'Defuse' "
                                   // "      'guard dog': 'Offer Food' "
                                   // "          'troll': 'Fight' "
                                   // "         'dragon': 'Fight' "
                                   // " 'dungeon master': 'Bribe' "
    }
  },
  "1": {
    "Passages": {
      "North": {
```

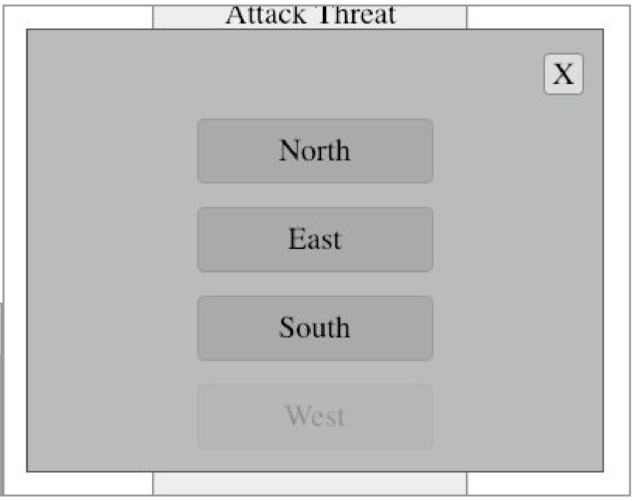
You can see that the main configuration object will consist of other objects containing properties on each room. Each room object will have an object for passages, an object for the treasure, and an object for the threat.
The passages object will contain properties for each direction a player can move in a room (north, east, south & west). Each direction which will have values determining whether or not the passage is open, or if it is the exit passage. If the passage is not the exit, but is still open, it will also require the connecting roomID for that particular passage.
For the treasure and threat objects, the only properties that they will require are the type, and the value / threat-action. For example, for a treasure of type ‘gold’, I would include the ‘value’ property. Or for a threat of type ‘dragon’, I would include the ‘action’ property with a value of ‘Fight’.
For the treasure type of ‘key’ however, there are no additional properties needed.

This format is then followed for as many rooms that the maze may contain.
When it comes to user-defined configuration, I saw an opportunity to create a web tool which will generate the config. This will avoid any human error if the user were to create a raw JSON object themselves.
I created this generator to be run in a procedural fashion - where the user is asked questions one at a time, in order to gather information on their desired maze design. It works with in-time validation - so if the user enters any information that could break the game, it will inform them of their mistake and give them a chance to correct said mistake.
By handling config generation in this manner, I am not only avoiding human error, but it also allows the entire process to be web-based, meaning that the user will never have leave the webpage.

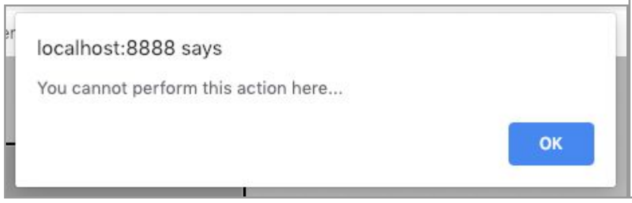
After creating the game to the new requirements, and incorporating the new configuration format, I am very pleased with the outcome.



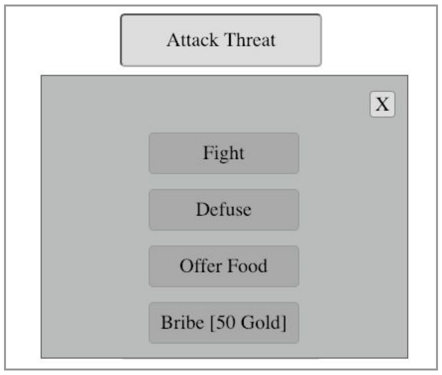
The majority of the left side of the page consists of an HTML canvas element - containing the view for the main game. Whereas the right side will be made up of player actions, and information on the current room.



The "Change Room" button when clicked will bring up the menu shown here. Where only the passages that are open, are available to the player, and any attempt to go down a closed passage will result in the player being alerted that an invalid option was selected.

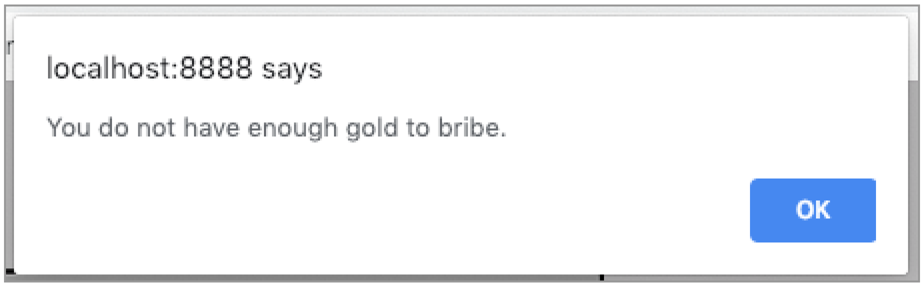

localhost:8888 says

You cannot perform this action here...



For attacking threats, I have left all 4 options available to the player at all times. Whatever action the player chooses, will still occur regardless of its effect on the threat. For example, if I was to try and 'Bribe' a 'Troll', it would not defeat the threat, but I would still lose money from the bribe. If the player does chose an action which has no effect, they will be informed - giving them a chance to try a different action on the threat.

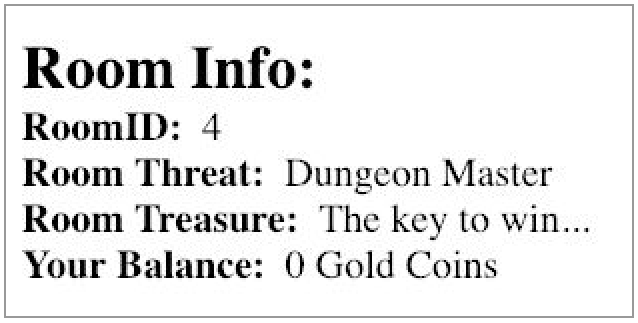
Once the player defeats the threat, the treasure will present itself, ready for collection.
I understand that the requirements of the project mentioned that the player should not be able to change room, until they defeat the present threat. However I was extremely fond of the idea of having ‘Bribe’ as an option for attacking threats, so for that reason I made it possible to change rooms without defeating the threat, as one threat does require wealth to be greater than 50 (the ‘dungeon master’) in order to defeat them. This gives the player a chance to accumulate wealth before defeating said threat.



localhost:8888 says

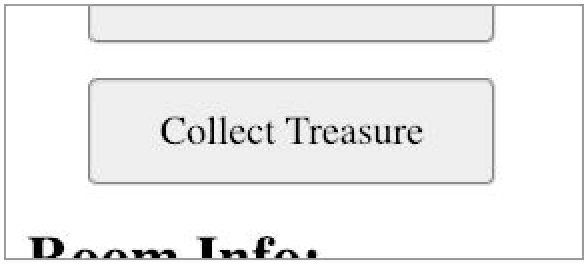
You do not have enough gold to bribe.

It also made sense to me to place the treasure type ‘key’ in the same room as the dungeon master, so that the player is forced to enter other rooms before they can retrieve the key. As the first room the player is placed in is randomly selected, it is possible they could start in the same room as they key, severely shortening the gametime without this addition.

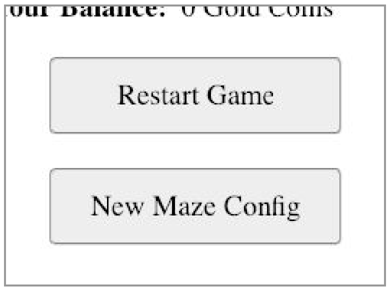


Room Info:
RoomID: 4
Room Threat: Dungeon Master
Room Treasure: The key to win...
Your Balance: 0 Gold Coins

One improvement I would have liked to make would be enforcing this rule of always having the ‘key’ in the same room as the ‘dungeon master’, so that they player always requires wealth to retrieve the key. Due to the config generation tool I have created, this gives to user freedom to choose where the key is placed - regardless of the threat type.



The “Collect Treasure” button is available for the user to click once the room threat has been defeated. There is no sub-menu for this action, it is simply dependant on the threat’s status.



The player can restart the maze at any time by clicking the “Restart Maze” button. Or if they are unhappy with the current configuration of the maze, they can always go back and create a new one by clicking the “New Maze Config” button. From here they will also have the option to use the predefined configuration.