Design Documentation

Project Description

My tp is going to be called 112maze and it is going to be a maze game that uses raycasting to create 3D visuals. The objective would be to collect a key and get to the end of the maze which is in a fixed location. As you progress through, the difficulty increases by a longer/more difficult maze or different enemy placements or different key placements. Key and enemy placements are random.

Competitive Analysis

A lot of other maze games are 2D/2.5D or often have gameplay similar to Pac-Man where you collect items of all locations in the maze to beat the level. My game will be another common archetype where it has enemies and items to collect to open a door down to lower levels where mazes get harder.

Structural Plan

I'm going to structure things by putting my maze algorithms in one file and splitting them up into classes/methods. Then for things such as players and enemies and items, I'll place them into their own file and split them up into different classes. Finally, the visuals will be handled in a separate file along with the main code to run everything.

Algorithmic Plan

For the maze generation (which I have finished one algorithm), I will look at the Wikipedia page for the algorithms and try to reconstruct the algorithms based on the pseudocode given. For algorithms that I would use, I would use one of DFS, Prim's, and Kruskal (extra after MVP).

Raycasting, I have a few tutorials and videos I could watch (and not look at the code in) to get an idea of how to implement it into my code. I might use DDA for this to make it more accurate.

Finally, for pathfinding of enemies, I'll also use Wikipedia pages and tutorials to get a general idea of the pseudocode and algorithm to implement it into my game. The pathfinding algorithms would be one of DFS and BFS

Timeline Plan

By TP0, I had my maze generation finished and a maze solving algorithm.

By the TP1, I want to try to have raycasting figured out and have it working somewhat (not necessarily working with my maze but hopefully that would happen as well).

Then, by TP2, I would likely hit MVP with pathfinding enemies, objectives for the maze, and maze completion (moving to the next level) done and have win-loss conditions to be fulfilled. Have a small splash screen. Only one level for now.

By TP3, I'd try to implement as many maze algorithms as possible and add some of these features: I will try to make a button that lets the user find a path to fulfill the objectives and complete the maze. There will be a minimap to let users see where they have traveled so far. There will be a timer so that people can see their times (or a countdown to make it harder). Fix movement to be mouse-based for turning. Fix key and enemy placements to be more logical (further away from the player and maze ending for the key, and enemies are placed between the keys and the players to have more interaction.

Version Control
The form of version control I am using is a GitHub repository which will be in a screenshot in the zip folder. I commit and update this repository every session of coding I finish (usually 1-2 times a day).

Module List
None (other than random and math)

Updates To Documentation Since TP1
I changed the descriptions to be SLIGHTLY more specific of “items” to keys and added a clearer definition of what MVP is. I also added a few items to what I want to do post-MVP.

Updates To TP since TP1 (TP2 Updates)
I tethered all of the components together so that the maze and raycasting work together. I fixed/added shading to the render and also sort of fixed the fish-eye effect. I added an end-goal and key collection to the game and fixed the solving algorithm to first lead you to the key and then the end. However, these two have not been added to be rendered in 3D yet.

Finally, there is an enemy but it moves very robotically between cells by teleporting to them. Enemies are rendered in 3D and you can dodge enemies by standing in spaces between their jumps.

Updates to Documentation Since TP2
I changed the algorithmic plans to match what I actually did and removed the algorithms I didn’t do such as Dijkstra, A*, and Eller’s.

Updates to TP since TP2 (TP3 Updates)
I moved the 2D to become a minimap and have the 3D as the permanent and main way to play. I tried to fix the rendering of the 3D enemies but it is still a bit janky, but should be more uniform. I added Prim’s and Kruskal’s maze generation and made it randomize between the 3 algorithms each time it generates a maze. I made enemies randomize between dfs and bfs (which I don’t think really matters in terms of gameplay for now). I added a rendering of the key and maze to the game. Last, I added a small feature where mazes would have small holes to make it an imperfect maze towards the lower levels, and from level 5+, they would transform to become a perfect maze.