Project Proposal
Anahita Hassan (arhassan)

Project Proposal

1. **Project Description**
   Name: 112 TA Simulator
   Description: You are a TA at Office hours and you have to find the OHQ student (who is first on the OH Queue) that is at the other end of the maze -- however, the maze is filled with ghost students who want to track you down and bug you with small questions. Your goal, as a TA, is to avoid the ghost students and reach the OHQ student as fast as possible.

2. **Competitive Analysis**
   This project is a mixture of pacman and a general maze solving game. The main pacman elements are primarily the student ghosts, who wander throughout the maze trying to track the TA down, and the TA must avoid them or else there will be some sort of penalty on their score. The difference between 112 TA simulator and pacman is that in pacman the goal is to eat all of the pellets throughout the maze, while in 112 TA simulator the goal is to just get to the other end of the maze. Additionally, this game is similar to any generic maze solving game in that the user has to get their avatar from one end of the maze to another.

   One general note is that pacman is not a perfect maze -- you should be able to loop through it easily in order to avoid ghosts. However, Prim's and Kruskal's algorithms are generally for perfect mazes (where there is exactly one path from one end to the other). I'm planning on implementing Prim's and Kruskal's such that my maze(s) are not perfect mazes and the player should be able to more freely run throughout the maze. The ultimate goal of the game, however, is still to reach the other end of the maze where the OHQ student is at.

3. **Structural Plan**

main.py	Modes: Home Screen Mode: keyPressed, redrawAll Help Screen Mode: keyPressed, redrawAll Game Mode: timerFired, keyPressed, redrawAll movePlayer, keyReleased Main App: initImages, etc for sprite sheets alterListOfPaths, removeMaze, appStarted makeBoardOfCoords, prims methods from prims.py, etc. Other/Misc: - If the user clicks the 'Prim's' button, the maze will be generated with Prim's algorithm, and same for Kruskal's. - Leaderboard tracking / Login
classes.py	Classes: Board Class: Maze generation TA Class: Keep track of movement OHQ Student: Sprite sheets Ghost Student Class: Keep track of ghost student's movement AND the TA's movement (using BFS to track TA down) Power up Class: Different unique attributes, perhaps when TA reachers a certain powerup, the ghosts tracking the TA will go back to their original position
prims.py	Implementation of prim's algorithm for maze generation
kruskals.py	Implementation of kruskal's algorithm for maze generation
bfs.py	Implementation of bfs algorithm for maze solving *note: maze solving will show a highlighted path that the user can take, and it takes into account the moving ghost students *also, ghost students will use bfs to track down TA movement

Modes (main.py):

Home Screen Mode	keyPressed, redrawAll
Help Screen Mode	keyPressed, redrawAll
Game Mode	timerFired, keyPressed, redrawAll movePlayer, keyReleased
Main App	initImages, etc for sprite sheets alterListOfPaths, removeMaze, appStarted makeBoardOfCoords, prims methods from prims.py, etc.

Classes (classes.py):

Board Class	Maze generation
TA Class	Keep track of movement
OHQ Student	Sprite sheets
Ghost Student Class	Keep track of ghost student's movement AND the TA's movement (using BFS to track TA down)
Power up Class	Different unique attributes, perhaps when TA reachers a certain powerup, the ghosts tracking the TA will go back to their original position

4. **Algorithmic Plan**

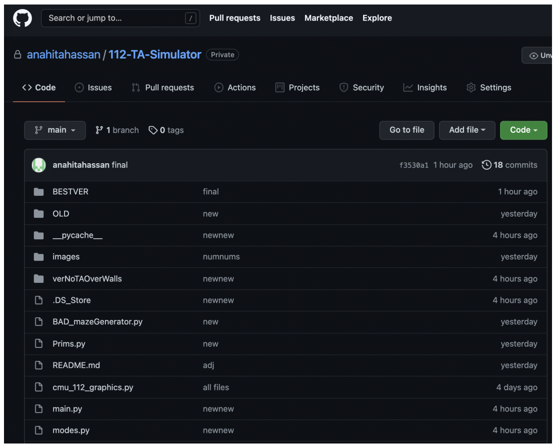
The most complex part of the project is most likely the maze generation and solving. I'm planning on using Prim's, Kruskal's, and the BFS algorithm for this. I'm planning on having a feature where the user can decide if they want the maze to be generated with Prim's or Kruskal's, etc. As of TP1, I've implemented Prim's and Kruskal's algorithm (though it is buggy, the edges aren't filled out, and the maze isn't quite as 'loopy' as I want it to be). To implement them, I've used Graph Theory principles (nodes and edges). Additionally, I am using BFS for maze solving. I'm planning on having a feature where if the user clicks the 'Help' button, a path showing the solution will appear, and it will also take into account the moving ghost student, meaning, this highlighted path can change. This will be a tricky part of the project, and I plan to approach it using the core concepts in BFS and graph theory principles.

5. Timeline Plan

Nov 19 (TP1)	Basic sprite based movement, maze generation, modalApp Complete implementation of Prim's and Kruskal's algorithm.
Nov 20	Fix the issue with the maze walls/corridors (the path drawn from Prim's/Kruskals's should be the corridor, not the wall!) Write up a BFS algorithm (highlights solution path). Ghost student class, get them to move around, then start implementing powerups. Make the maze have more than 1 solution.
Nov 21	Basic ghost student pathfinding (BFS). Finish implementing Kruskals.
Nov 22	Powerups implemented. Ghost student pathfinding / highlighted path with respect to ghost student movement.
Nov 23 (TP2--MVP)	Reach MVP (algorithmic complexity component completed and working), updated design docs
Nov 24-28	Debugging (if needed), leaderboard/login, make UI visually appealing (draw Gates Commons background, perhaps draw better sprite sheets if time)
Nov 29-30	Minor last minute changes, make project more visually appealing, make demo videos, readMe, codebase, etc.
Dec 1(TP3)	Submit all

6. Version Control Plan

I'm using Github to back up my code and store different versions.



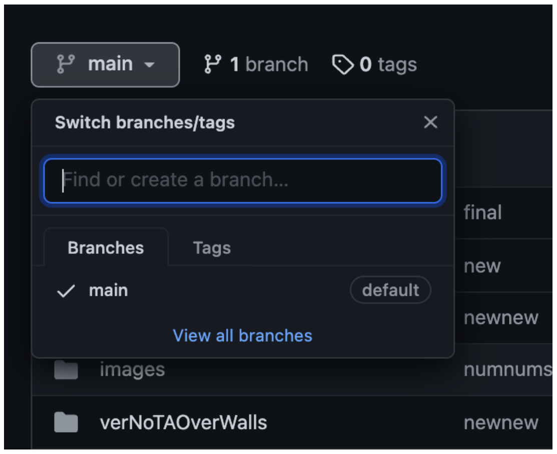
anahitahassan / 112-TA-Simulator Private

main · 1 branch · 0 tags

anahitahassan final · f3530a1 1 hour ago · 18 commits

BESTVER	final	1 hour ago
OLD	new	yesterday
__pycache__	newnew	4 hours ago
images	numnums	yesterday
verNoTAOverWalls	newnew	4 hours ago
.DS_Store	newnew	4 hours ago
BAD_mazeGenerator.py	new	yesterday
Prims.py	new	yesterday
README.md	adj	yesterday
cmu_112_graphics.py	all files	4 days ago
main.py	newnew	4 hours ago
modes.py	newnew	4 hours ago

Github Repo



main · 1 branch · 0 tags

Switch branches/tags

Find or create a branch...

Branches · Tags

✓ main default

View all branches

Version control: branches

7. Module List

I'm not planning on using any external modules/hardware/technologies.

8. <u>TP2 Updates</u>

I have not made any major design changes.
Few small updates:

1. I've decided to use a perfect maze instead of a more 'loopy' maze like in pacman that would allow the avatar to go in circles or loops. This would make it harder for the user to get from point A to B given the enemies, so I'm placing power ups at various parts of the maze that can do various things like (1) make all the enemies return to their original locations or (2) the TA can temporarily 'take off the blue 112 TA sweater' and disguise as a fellow student so if they bump into enemies it's harmless.
2. In my TP1 meeting, I realized that when I made edges connecting nodes in my Prim's/Kruskal's algorithm, I accidentally drew the edges as the walls, when they should've been the corridors. Fixing this only required changing a little code in gameMode_keyPressed.
3. Instead of having a 'help' button, I changed it so the help action occurs when the user presses 's'.
4. My initial plan was to have the enemies track the TA with the BFS algorithm, instead I am using a backtracking/recursive algorithm.

9. <u>TP3 Updates</u>

Since TP2, I have added several new features.

- Bonus round (Random walk maze generation):
    - In main.py, I have a bonus mode where the user can earn additional points for every student in the maze they hit. This is kind of similar to the pellets in pacman, except here they are students in the maze and the TA gets additional health points for reaching them.
    - For maze generation, I used the Random walk algorithm which makes the maze a lot more 'loopy'.
- Leaderboard and Text editor:
    - There is now a leaderboard on the home screen page. You have to enter your name in the text editor (which works for special keys like backspace, left/right arrows, etc) and your score will be saved at the end.
    - The leaderboard saves everything in a text file so past high scores can still be displayed each time the program is run.
- Power up
    - Randomly generated in the maze. If the player reaches a power up, the enemy students that are tracking it will momentarily stop, giving the player the chance to catch up to the end of the maze.
- Scoring System
    - Player starts off with 20 health, each time they bump into a student in the maze it's -10 health (for however long the student and player are in the same location). In the bonus round, the player gets +1 health for every student it hits in the maze.
    - Names and scores are saved globally in a txt file
- Misc:
    - Many shortcut commands changed, they are all summarized in the ReadMe file and in the help mode in the game.
    - I ended up not using Ghost Student class and Power up class
    - In addition to prims.py and kruskals.py, I had randomWalkAlgo.py (for maze generation with random walk algorithm) and backtracker.py (for maze solving). All the complex algorithms have their own file, basically.