Pathing Simulator

CSCE-312 (Computer Organization) Capstone project for the Online Nand2Tetris Course:
www.nand2tetris.org

Purpose

- “Pathing-Simulator” is a small-scale sandbox game that allows the user to create their own traversable grid that the program will attempt to navigate through from a starting point to a finishing point and draw a corresponding shortest path. This program allows the user to visualize graph traversal algorithms in real-time and make their own graphs to see how the traversal algorithm navigates through them.

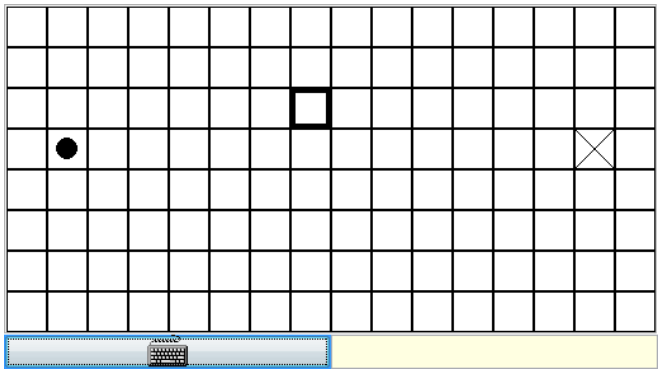
- The original purpose behind making this game/visual-tool was to create a user-controlled environment that could visualize graph traversal algorithms that many students have already learned about and implemented in their corresponding Data Structures & Algorithms course. Such students have run graph traversal algorithms such as BFS and DFS on their graphs to find a shortest path but may not of had the chance to actually see them execute in real-time, this tool attempts to solve that problem by visualizing the BFS (breadth-first-search) algorithm in action.

Concept

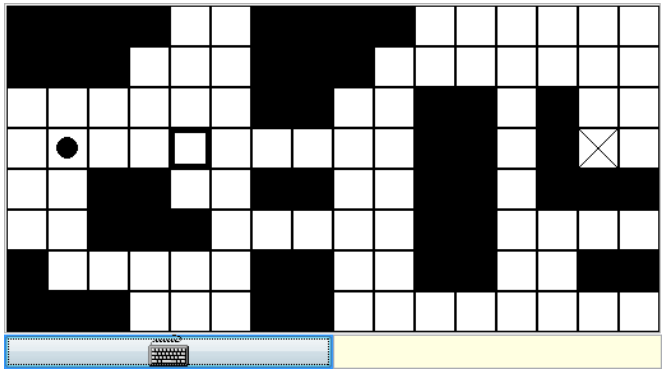
- You are given a 8x16 cell grid, a starting point (the black circle), and a finishing point (the X)
- From here you can change the starting and finishing points, and you have the ability to “paint” the cells either white (traversable) or black (non-traversable) with the selector cell.
- Once you are satisfied with your grid layout, you can enter the execution stage and the program will attempt to navigate from the starting point to the finishing point by drawing a line, if a path cannot be drawn, the program will let you know and prompt you to try another grid layout.
- If a path exists, the program will draw it, if there is no path, the program will let you know. The path-finding algorithm will be visualized when you enter the execute stage to show the user what is going on “under-the-hood” in the algorithm.
- There are many different possible layouts to try, the choice is up to you!

Game Instructions:
Set Start Cell: S
Set Finish Cell: F
Move Selector Cell: directional keys
Draw Path: E

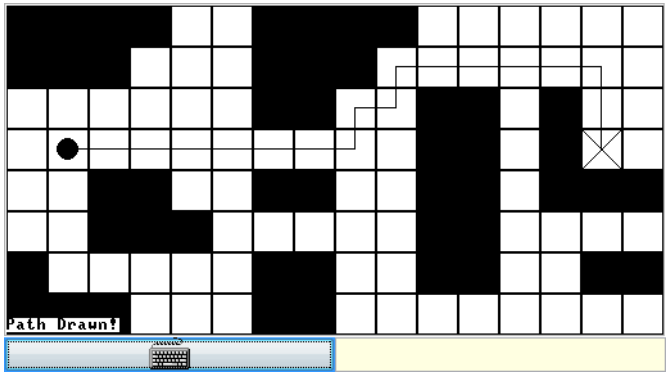
Step 1: Load the game



Step 2: Paint your grid



Step 3: Draw the Path!



Rules

- Since this is a traditional sandbox game, there aren't any rules.
- The player can set the starting/finishing cells where ever they desire and can paint whatever cells they wish.
- However, if the player implements a grid layout such that there is no path from the starting point to the finishing point, the traversal algorithm will attempt to run, and then prompt the user if a path cannot be determined. This is shown in the video demonstration.

Challenges

- In order to implement the game, two separate auxiliary data structures and two algorithms were implemented:
- **ObjectArray.jack** – implements an array data structure that holds a specific instance of a class. The program uses this objectArray to create a 2D array of objects that holds each of our individual cells (8x16 grid). Jack's native array data structure only handles integers.
- **ObjectList.jack –** implements a list data structure that holds specific class objects which has all of the conventional list methods such as push, pop, enqueue, dequeue, etc. This list data structure can be used as a stack/queue data structure, which is what was used in order to implement the BFS algorithm.
- **BFS-Algorithm –** enqueues the starting cell (set by the user) as the eldest parent, which then visits each unvisited and traversable neighboring cell in the cell's adjacency list containing its north, east, south, & west neighbors, marking each as visited, setting their parent cell, and queuing them for further iteration. The algorithm iterates, checking every node in the same BFS-tree level until moving onto the next level, stopping immediately when the finish node is found.
- **Shortest-Path-Algorithm –** Since the destination is always stored in memory, we start at the finishing node and draw a path back to the starting node by back-tracking through the BFS-tree starting with the finishing node (the finishing node being a leaf in the tree, and the starting node being the eldest parent or root node). The algorithm draws a line from the center of the current cell to the center of the current cell's parent cell, continuing on until a path from start to finish is drawn.
- Other graph traversal algorithms could be implemented into the game as well for visualization purposed, such as the DFS algorithm or the more widely used A* algorithm. With the ObjectArray and ObjectList data structures implemented, all the tools one would need to implement these algorithms are there. For the sake of time at the end of the semester, I was only able to implement the BFS-Algorithm.