Assignment 2 - Maze (JS/HTML)

Submit Assignment

Due Feb 23 by 11:59pm **Points** 81 **Submitting** a file upload **File Types** zip
Available Feb 3 at 12am - Feb 23 at 11:59pm 21 days

Introduction

Write a program that automatically generates mazes for a player to solve. The purpose of this assignment is to help you learn to build a more complex 2D HTML Canvas game, along with exercising some of those Data Structures & Algorithms neurons from your CS1/2/3 courses. The program will generate random mazes according to a size chosen by the player, then allow the player to solve it, while keeping score and providing the ability to offer hints.

Assignment

Write a 2D HTML Canvas game according to the following specifications...

- Random generation of mazes using one of the Randomized Kruskal's, Randomized Prim's, or the Recursive Division methods, but not the Depth-First algorithm. Refer to the following Wikipedia page for maze generation algorithms: **http://en.wikipedia.org/wiki/Maze_generation_algorithm (http://en.wikipedia.org/wiki/Maze_generation_algorithm)**
- User may choose mazes of size 5x5, 10x10, 15x15 and 20x20
- Hint toggle option : Tells the user the next best square to choose. The hint stays persistent and updated until the user toggles it off.
- In Game Scoring (just do something that makes sense, it doesn't have to be this exactly)
  - 5 points for each correct square along shortest path; first time it is entered.
  - -1 for square adjacent to shortest path; first time it is entered.
  - -2 for all other squares; first time it is entered.
- Simple UI : New Game, High Scores, Credits
  - High Scores must be persisted using the browser's local storage.
  - How you organize the high scores is fine with me. You can organize by maze size or track highest score regardless of maze size, but you'll probably want to record the maze size along with the score.
- While playing, user interface that shows the current player score and elapsed time (accurate to the number of seconds).
- At start, you can choose any initial location for the player and end point for end of the maze. It might be upper left for the start and lower right for the end, but it doesn't have to be those points.
  - Show the ending location with some marker
- As the player moves, they leave a breadcrumb trail behind showing which cells have been visited.
- Other Requirements
  - Ability to toggle a breadcrumbs trail.
  - Ability to toggle shortest path to the finish (a stack helps with this).
  - Display an image that covers the background of the maze.

Many of the above capabilities mean you have to write code that can find the shortest path. This can be done by using a breadth-first search, but feel free to use any reasonable technique. Here is a wiki link that talks about different maze solving algorithms, with a reference to using breadth-first search to find the shortest path (with multiple solutions) **http://en.wikipedia.org/wiki/Maze_solving_algorithm (http://en.wikipedia.org/wiki/Maze_solving_algorithm)**

Control Scheme

Player movement is controlled by using the arrow keys and WASD/IJKL (all three must be active) keys; one key-press moves one square; holding down the key should not repeatedly move the character. For the additional requirements, please use the following controls:

- Hint toggle : Keyboard 'h'
- Breadcrumbs toggle : Keyboard 'b'
- Path to finish toggle : Keyboard 'p'

Technical Notes

Your code must have a game loop that follows the pattern learned from the first assignment. It should look something like...

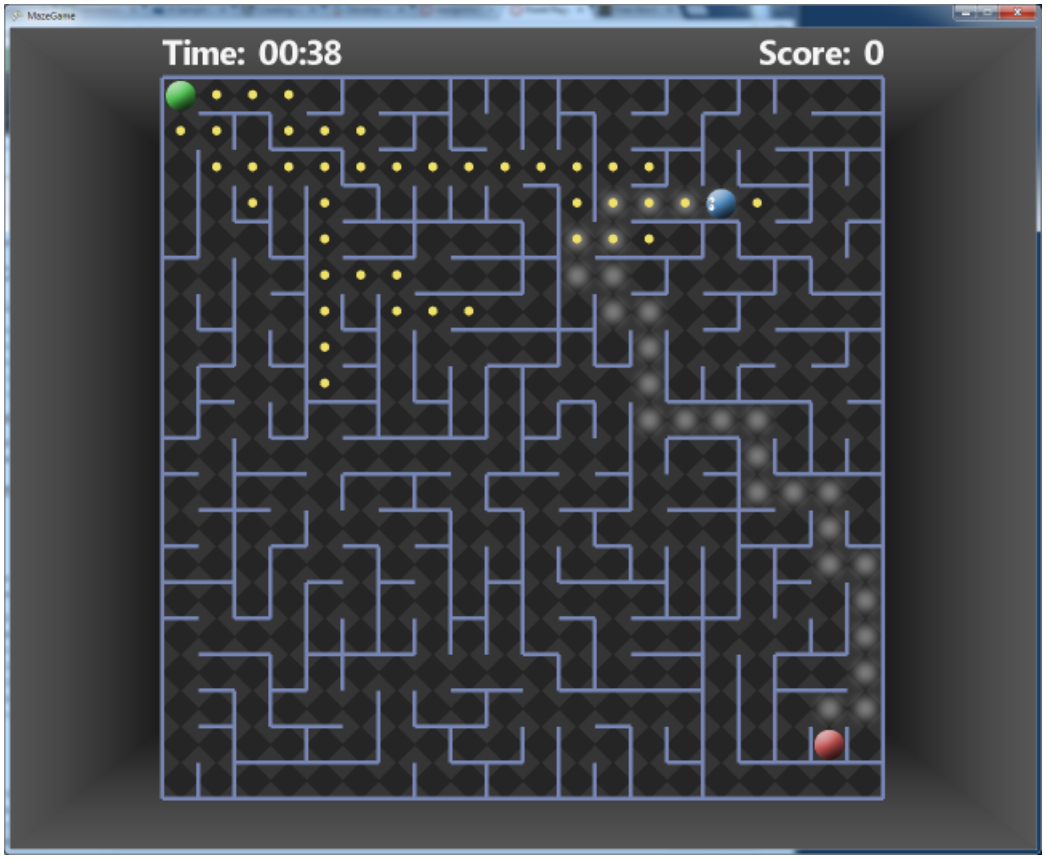
```
function gameLoop(time) {
    ...compute elapsed time...

    processInput(elapsedTime);
    update(elapsedTime);
    render(elapsedTime);

    requestAnimationFrame(gameLoop);
}
```

Development Notes

My recommendation is to begin the assignment by first working on developing the JavaScript code to generate the random mazes. You can create a simple HTML page (or using JS and node from a console) and associated JavaScript file and begin working on the maze generation algorithm. Once the maze generation code is working, then develop a model for the game simulation, along with rendering and so on.

JavaScript is different from C++/C#/Java, and the way you think about and organize your code will be different. Take the time to "think" in JavaScript and code accordingly. Again, I don't expect perfection on this assignment, but looking to see progress made.

Sample Gameplay Screenshot



Assignment 2 - Maze Game (JS/HTML)

Criteria	Ratings		Pts
Works on both Firefox and Chrome. 0 points if it only works on 1 of the browsers. Ideally it should work on all of MS Edge, Safari, Firefox, and Chrome.	**5 pts** **Full Marks**	**0 pts** **No Marks**	5 pts
Random Maze Generation & Rendering	**30 pts** **Full Marks**	**0 pts** **No Marks**	30 pts
Support for all maze sizes	**6 pts** **Full Marks**	**0 pts** **No Marks**	6 pts
Game scoring	**5 pts** **Full Marks**	**0 pts** **No Marks**	5 pts
Basic UI to allow the game to be started/played.	**5 pts** **Full Marks**	**0 pts** **No Marks**	5 pts
High Scores Persisted to the browser's local storage.	**5 pts** **Full Marks**	**0 pts** **No Marks**	5 pts
Breadcrumbs	**5 pts** **Full Marks**	**0 pts** **No Marks**	5 pts
Hint	**5 pts** **Full Marks**	**0 pts** **No Marks**	5 pts
Shortest Path	**10 pts** **Full Marks**	**0 pts** **No Marks**	10 pts
Display of time taken to solve the maze	**2 pts** **Full Marks**	**0 pts** **No Marks**	2 pts
Maze background image	**3 pts** **Full Marks**	**0 pts** **No Marks**	3 pts
			Total Points: 81