CS 351: Design of Large Programs
Project 4: Mazes
Due: 4-15-21

Brooke Chenoweth and Joseph Haugh

Spring 2021

1 Assignment Description

This is a group project. Each group consists of 2 students that I chose based on your survey responses and other factors.

In this assignment you will be tasked with implementing several algorithms for maze generation as well as maze solving. For maze solving you will have to utilize multithreading in your algorithms.

2 Maze Generation Algorithms

1. Randomized Depth First Search
2. Randomized Kruskal's
3. Randomized Prim's
4. Aldous-Broder
5. Recursive Division
6. More Info: `https://en.wikipedia.org/wiki/Maze_generation_algorithm`

3 Maze Solving Algorithms

1. Random Mouse
2. Wall Follower
3. Pledge
4. Trémaux

5. Maze Routing
6. A*
7. More Info: `https://en.wikipedia.org/wiki/Maze-solving_algorithm`

4 Requirements

1. Must use JavaFX for all visualizations
2. Must strive for code reuse
    - For example you should only have a single notion of a board
    - Should only have a single notion of a cell on the board
3. Must choose a random starting point that is reachable from the maze
4. Must choose a random end point on the opposite wall to the starting point that is reachable from the maze
5. Must implement the randomized depth first search generation algorithm
6. Must implement the randomized Kruskal's algorithm
7. All solving algorithm need to visually show where each solving agent is in the maze
8. Must implement the random mouse solving algorithm
9. Must implement the wall follower algorithm
10. Must use no more than 1000 threads at any one time
11. Must augment the random mouse solving algorithm to use multithreading
    - Start at the beginning of the maze
    - The mouse should not backtrack
    - When the "mouse" encounters a junction in the maze it must create a copy of itself for each available path
    - Each copy must proceed on its own thread while the original mouse dies
    - When one mouse reaches the end of the maze all other mice (threads) must be killed
12. Must augment the wall follower algorithm to use multithreading
    - Start a wall follower at the beginning of the maze
    - Also start a wall follower at the end of the maze on a different wall
    - Both wall followers must be on their own threads

13. Must implement an additional generating algorithm from the list above
14. Must implement an additional solving algorithm from the list above
15. If you implement an additional generating/solving algorithm you can receive extra credit but only once
16. If you make the board look visually appealing you can receive extra credit

5 Mandatory Generation Algorithm Descriptions

Randomized Depth First Search:

1. Choose the initial cell, mark it as visited and push it to the stack
2. While the stack is not empty
   (a) Pop a cell from the stack and make it a current cell
   (b) If the current cell has any neighbours which have not been visited
      i. Push the current cell to the stack
      ii. Choose one of the unvisited neighbours
      iii. Remove the wall between the current cell and the chosen cell
      iv. Mark the chosen cell as visited and push it to the stack

Randomized Kruskal

1. Create a list of all walls, and create a set for each cell, each containing just that one cell.
2. For each wall, in some random order:
   (a) If the cells divided by this wall belong to distinct sets:
      i. Remove the current wall.
      ii. Join the sets of the formerly divided cells.

6 Program Inputs

Your program must accept a file name as a command line argument. The file will have the following format:

```
{window size}
{cell size}
{generator}
{solver}
```

Where:

- {window size} is an integer that determines the length and width of the window
- {cell size} is an integer that determines the lenght and widht of each cell
- {generator} is one of the following:
  - dfs
  - kruskal
  - prim
  - aldous
  - rec
- {solver} is one of the following:
  - mouse
  - mouse_thread
  - wall
  - pledge
  - tremaux
  - routing
  - astar

For example a file might look like this:

```
900
10
kruskal
mouse_thread
```

This means your window needs to be 900 x 900, each cell is 10 x 10, uses kruskal's algorithm to generate the maze and the random mouse algorithm with multithreading to solve it.