Maze

Programmer Manual

Maze

1. Problem Description

The Maze class consists of a two dimensional character array representing the maze. A Tile pointer is used to represent the current position during the search. A depth first search algorithm is used with a stack in order to find the exit of the maze, or determine if it is not possible to reach the exit from the starting position.

2. Class Maze

Private data members:

Tile* current	position currently used in the search
char board[][]	the array containing the maze data
bool init	flag determining whether the maze has data or not

Private member functions:

print	prints the maze
destroy	deletes neighboring Tile pointers of the current Tile

Public member functions:

Maze	constructor for a Maze object
solveMaze	solves a maze or determines it is impossible
generateMaze	randomly generates a maze to solve
mazeFromFile	gets a maze from a user input file

3. High Level Program Solution

```
Maze
    sets init to false

mazeFromFile
    opens a file from a string input by the user
    if the filename is invalid, set init to false and return to the main menu
    if the filename is valid, initialize the board array according to the data in the file, adding
        '1's around the outside of the maze
    close the input file
    set init to true and return init
```

```
solveMaze
        declare a Tile struct
        declare a stack of Tile pointers
        print out the empty maze
        get a starting position from the user and validate it
        if the starting position is the exit, return
        set the starting position as the current Tile
        push the current Tile onto the stack and mark it as visited
        while the stack is not empty:
            initialize the currents surrounding Tile pointers
            if one of the surrounding Tile pointers is an empty space or the exit, push it onto
                the stack
            set the current Tile to the top of the stack
            while the stack is not empty and the current Tile is visited:
                set the top of the stack to the empty character unless it is the starting
                    position
                pop a tile from the stack
                if the stack is empty, no exit could be found, so print the maze and return
                set the current Tile to the top of the stack
            if the current Tile is the exit, print the maze and return
            set the current Tile to the path character
            mark the current Tile as visited
            delete the neighbors of the current Tile
            declare the neighbors of the next current Tile

generateMaze
        seed a random number generator using the time
        make the outside of the maze all walls
        create a maze with each tile having a 25% chance to be a wall, otherwise the tile is
            empty space
        randomly select a tile to be the exit

print
        set all of the characters to be used by the maze
        if the maze character is a '1', use the wall character
        if the maze character is a '0', use the ground character
        if the maze character is an 'E', use the exit character
        if the maze character is an 'S', use the start character
        print the maze character by character
        print the legend

destroy
        delete the above neighbor of the current tile
        delete the right neighbor of the current tile
        delete the below neighbor of the current tile
        delete the left neighbor of the current tile
```