Yoav Nathaniel

Artificial Intelligence

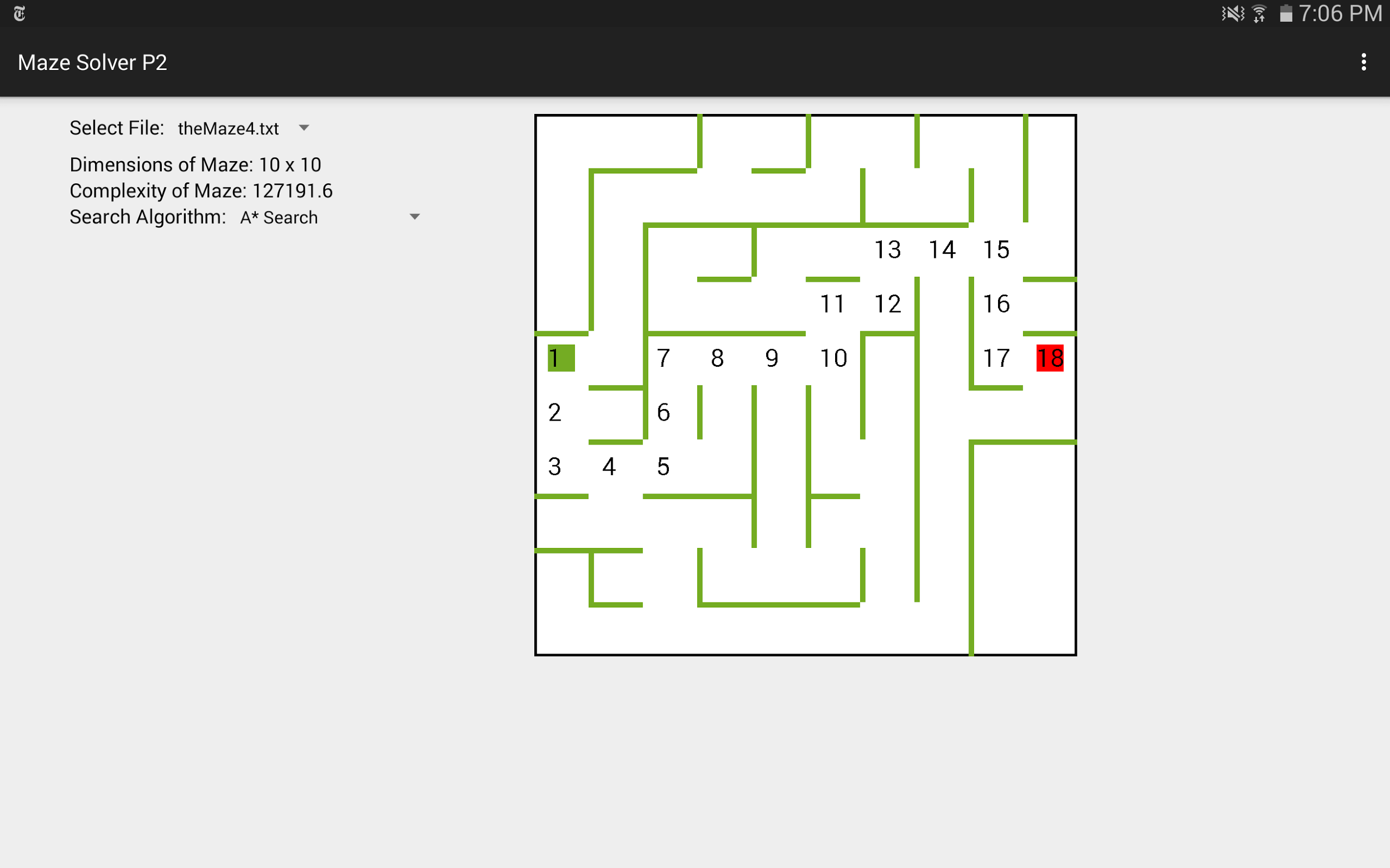
Professor Kumar

March 10, 2015

Project 2 - Maze Solver - Technical Manual

**1 - Activities:**

MainActivity - after extensive research and trial and error phase (of which I overcame), I managed to mimic the CompleteActivity, but using Imageview, Canvas, and Bitmap.



**2 - AI Algorithms:**

The project uses Branch and Bound, Depth First Search, Breadth First Search, Best First Search and A\* Search.

All of the above algorithms are found in the MazePath class and called either by BuildGrid methods or FileManager methods. Their results then are drawn by DrawGrid’s drawSearch method.

Note: Branch and Bound is the only algorithm that is really separated into two parts, one finds all of the paths while the other finds the number of turns in the path passed in.

**3 - Bug Report:**

I do not suspect of any critical bugs in my code.

I think the Branch and Bound algorithm may be producing slightly wrong results for the complexity of the maze.

For a maze to work, there must be proper maze format - spaces between all walls. a 6x6 grid with all of the walls should have a first row filled with the following:

ASCII characters:

032 045 032 045 032 045 032 045 032 045 032 045 032 013 010

Hex Letters:

20 2d 20 2d 20 2d 20 2d 20 2d 20 2d 20 0d 0a

**4 - Feature Report:**

Lacking Features:

None.

Extra Features:

Might not be considered extra - print the steps of each search algorithm on the maze.

**5 - Log:**

**2/14/15**

--Create a new project Maze Solver P1. Import copy Grid, Cell, Wall, and Room classes from Project 1.

--Create the xml for activity\_main.xml. Let the maze dimensions work with radio buttons. In MainActivity.java, I created a system radioClick function to detect clicks on radio buttons and change variable dimens accordingly. The dimensions currently only go from 6x6 to 10x10.

**(0.75 hour)**

--Experiment with working with a file. I’m trying to play around with File class to see how to create or overwrite a file in a specific directory.

**(1 hour)**

**(Total 1.75 hours)**

**2/16/15**

--Created generateMaze and writeToFile functions in MainActivity of Maze Solver P1. generateMaze Generates a random maze and creates a char array filled with ASCII codes of the maze. Then it calls writeToFile which converts the char array into a byte array and puts that in theMaze.txt file in Documents folder of the Android device.

--Finished with Maze Solver P1.

**(1.25 hours)**

--Created Maze Solver P2 and, inside of MainActivity, set up readFromTextFile function. It reads from the text file set up by Maze Solver P1 and reads the data to see the dimensions of the Grid.

--Imported Cell, Wall, Room, and DrawCell classes from Maze Generator (Project 1).

-- Wrote BuildGrid class - it builds a grid from the data and dimensions read by the text file.

**(1 hour)**

**(Total: 2.25 hours)**

**2/24/15**

--Created MazePath class, responsible for generating all searches.

--Began Branch and Bound algorithm in MazePath - worked on findPaths.

**(2 hours)**

--TroubleShoot - findPaths would not find Options correctly and so would not advance to new paths.

--Rewrote findPaths with an organized mind. All paths done move from pathsFound to pathsDone.

**(2 hours)**

--Revised all of the code to fit a grid corresponding to the teacher’s qualifications. Instead of having a grid of size 6x6 = ((6 \* 2) - 1)^2, I changed it to ((6 \* 2) +1)^2. It accounts for the walls on the borders of the grid.

--Code is all scrambled and not really working.

**(2 hours)**

**(Total: 6 hours)**

**2/28/15**

--Changed code back to original form

**(1 hour)**

--Changed the way text files are being read - they do the math and build a ((6 \* 2) - 1)^2 grid from ((6 \* 2) +1)^2 file.

**(1.5 hours)**

--fixed issues with translating the grid. Inside the BuildGrid class.

**(2 hour)**

--drawSearch method in DrawGrid class and depthFirstSearch method in MazePath class. The first search method of the project, besides the branch and bound which is completed.

**(1.5 hours)**

--Created checkCellDistance, Breadth First Search, and Best First Search in MazePath. checkCellDistance is a method calculating the distance between a given cell and the goal cell.

--Finished drawSearch method to display the steps each algorithm takes. It is a generic method that simply takes a path

--detected error for implementing different map sizes. Will fix sometime in the near future.

**(2 hours)**

**(Total: 8 hours)**

**3/2/15**

--Fix error for implementing different map sizes. I accidentally hard coded the reader to take only 6x6 maps. Now it’s set to a generic formula that takes all of the maps sizes.

--Fix error for best first search. For some reason it was not working, I could not identify the reason. I rethought of the algorithm and wrote it in a more efficient and less buggy way..

--Fixed the font of the numbers in maze. Now they match the varying sizes of the cells better.

**(1 hour)**

--Begin implementing a\* search algorithm. Wanted to use a priority queue but the comparator would not work as it would involve logic existing outside of the properties of the cells. Decided to stick to a vector<vector<cell>>. Might be inefficient.

**(0.5 hour)**

--Keep working on a\* search algorithm in MazePath class. I try to mix the best first search and the branch and bound. Currently not successful at 6x6 and other maze sizes crash.

**(0.5 hour)**

**(Total: 2 hours)**

**3/9/15**

--Fix major bug in case there are no files in the Documents directory. Program would generally crash, but not anymore.

--Wrote descriptions for the different classes and methods, and filled out the technical manual.

**(Total: 2 hours)**

**Total Project Time: 22 Hours**