Cherie Hua

Project Proposal

**Sources Used**

* an edited version of cmu\_112\_graphics.py from <https://www.cs.cmu.edu/~112/notes/notes-animations-part1.html>

**Feedback From Lecture 12/3/19**

* To find the most optimal move, make a list of probabilities and sort that list. Take the first value of that list (the lowest probability) and make the bot click that tile.
* Make sure the bot doesn’t lose.

**Project Description**

Minesweeper Teacher — a game that analyzes the best possible next move from the player’s position, the probabilities of a mine in each square, and the best optimal way to reach the player’s position in order to teach the player how to play minesweeper in the most efficient way (least number of moves).

**Competitive Analysis**

[Minesweeper Solver](https://www.logigames.com/minesweeper/solver) allows you to input a custom board, then solves the board based on the given information. My project will also use a given board and look for the best possible solution. However, this solver does not guess and doesn’t show probabilities of a mine being in each square.

[Automatic Minesweeper Solver](https://www.nayuki.io/page/automatic-minesweeper-solver) is a bit closer to what I want to implement. This program makes guesses (and sometimes wrong guesses) to find the best way to solve a board. The source code is available in Java, so I can look at how to program calculates which guess is the best.

[Minesweepr](https://mrgris.com/projects/minesweepr/) is similar to Automatic Minesweeper Solver, except it has detailed documentation. I can use this is as a guide for my own project.

**Structural Plan**

* Files: 1 file for base minesweeper game (minesweeper.py), 1 file for the optimal next move predictor/probability display (probabilityPredictor.py), 1 file for program that finds the optimal way to get to the player’s location (optimalPathFinder.py)
* Detailed structural plan for minesweeper.py

class Minesweeper(App)

class Board

function create board

2D list

function getCellBounds:

finds which cell is pressed

class Game

function Flag

function Remove Square

function mousePressed:

if this is the first time the mouse is pressed, call make board

class Tile

object Square

removable, when number is removed then a number or blank square appears

object Mine inherits from Square

looks like square, but when it's removed the game ends

(the player hit a mine)

object Number

shows up when a square is removed, showing the number of mines in the 8

surrounding squares

object Blank Square (no surrounding mines)

shows blank square

#outside of classes

function win

displays location of mines and ends game

text = you win

function lose

displays location of mines and ends game

text = you lose

Minesweeper(width = 800, height = 800)

* probabilityPredictor.py
* optimalPathFinder.py

**Algorithmic Plan** [5 pts]: A detailed algorithmic plan for how you will approach the trickiest part of the project.

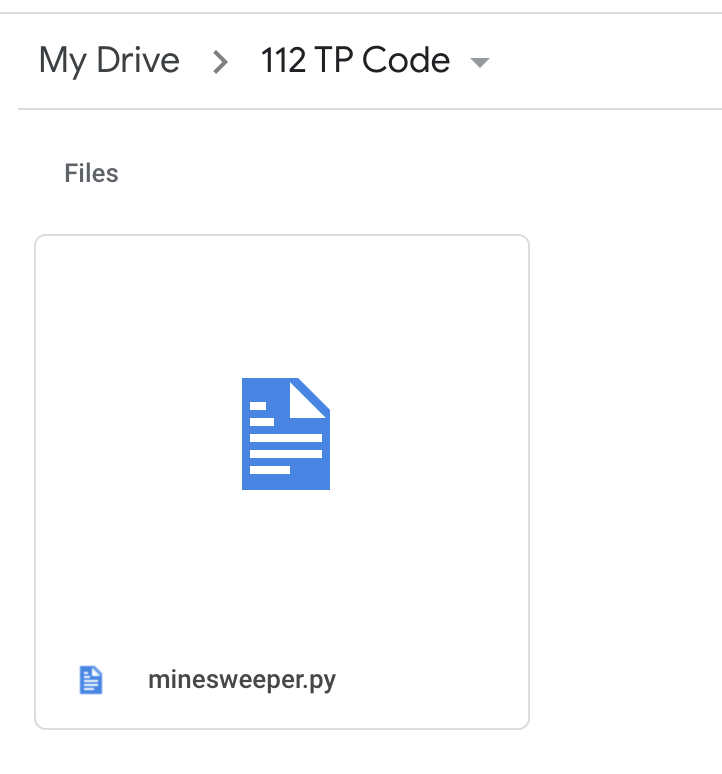
* Tricky Part #1: Finding the probability of mines (the rest of the teacher program is built off of this, so it’s extremely important)
  + look at the surrounding numbers
  + start w/ 100% probability, look at adjacent numbers
  + based on number patters such as 121,
* Tricky Part #2: Efficiency — what counts as a move?
  + what counts as 1 move: clicking a square, clicking a number (to clear all unflagged squares around it), flagging??
  + from the Minesweeper wiki: “Some players never flag because time spent flagging can be better used to open more squares. This style is called No Flags, or NF. Flaggers argue that flags allow you to chord and clear multiple squares at the same time. It is generally agreed that NF is more efficient near high numbers (5,6,7,8) while Flagging is more efficient near low numbers (1,2,3,4).”
  + This can be an advanced feature that calculates whether it would be more efficient to flag or clear (click on the number).

**Timeline Plan**

|  |  |
| --- | --- |
| Date Complete | Feature |
| Wed Nov 20 | Base Minesweeper Game |
| Sat Nov 23 | Find probability of each square to predict next optimal move, add explanations for each optimal move |
| Tue Nov 26 | Keep track of moves, find way to get to player’s move in the most optimal way, allow them to scroll through moves |
| 5 Dec | UI, different modes, refine probability calculator, extra features |

**Version Control Plan**

I will upload code to Google Drive every 50 lines or so.



**Module List**

I am not planning on using any modules.

**TP2 Update**

* UI change — the teacher is now on the side of the screen instead
* instead of scrolling through moves, just predict the next move and you’re given an explanation
* can turn on a setting to show probabilities of each mine in a square based on the user’s board

**TP3 Update**

* explanations