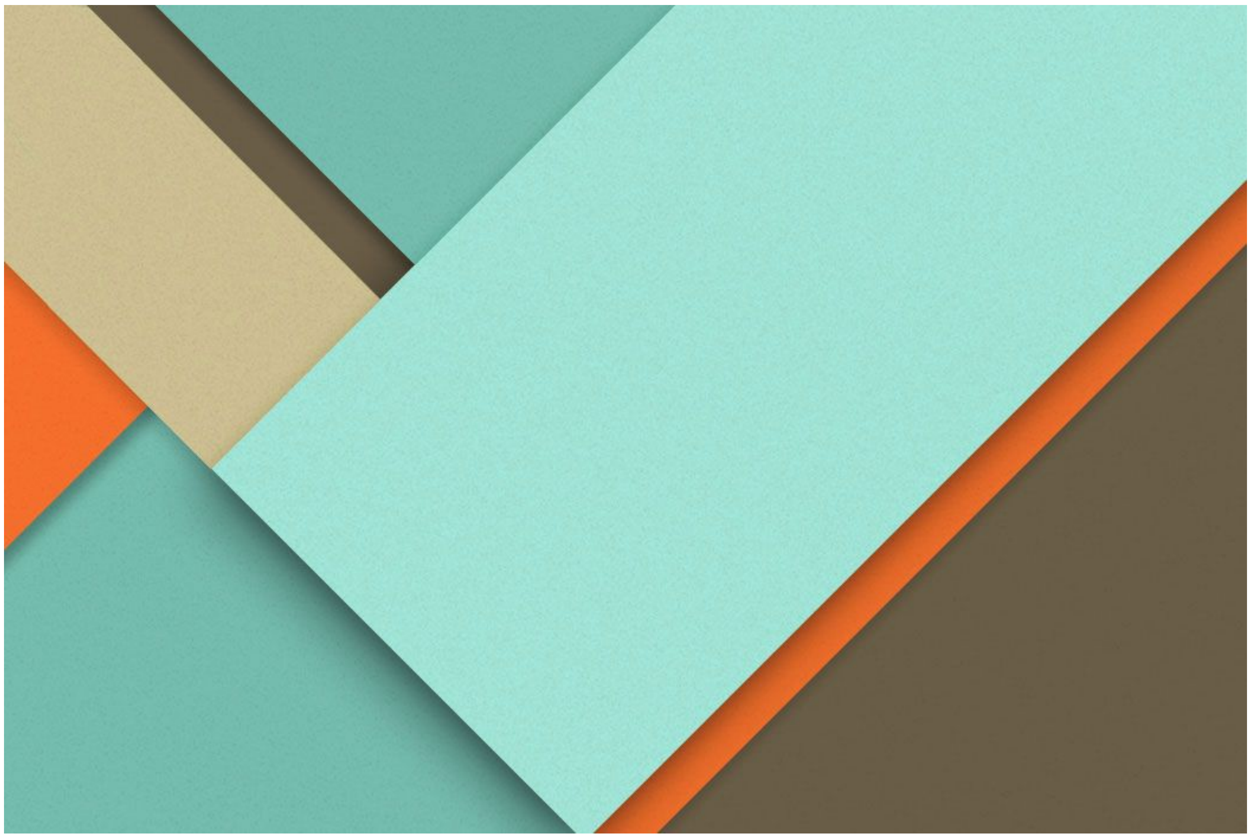


Artificial Intelligence (COMP6065)

18.01.2018

Mike Daniel Wibowo - 1901521500

Muhammad Arvin - 1901521526

BINUS International

Computer Science

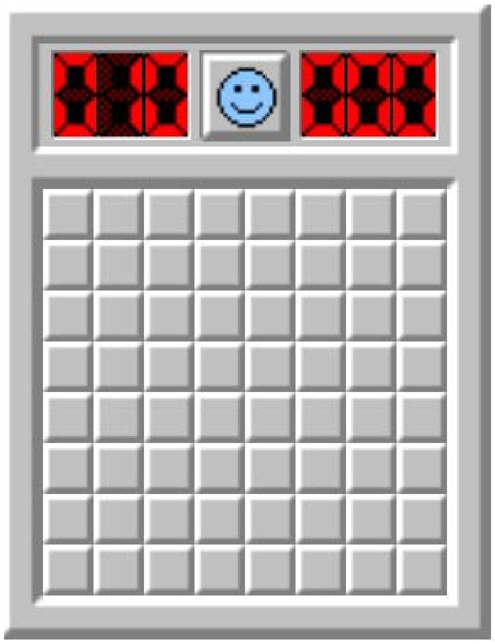


Table of Content

Introduction

1. Background

Minesweeper is a single-player puzzle video game. The objective of the game is to clear a rectangular board that contains a hidden “mines” or bombs without detonating any of them, with help from clues about the number of neighboring mines in each field. Our minesweeper AI can play the board with a probability of the mines in the area, and obvious algorithm. The project is inspired by Minesweeper from Microsoft Corporation.

2. Programming Language

The puzzle is entirely using a C++ and effolkronium random library. We choose C++ as the programming language because C++ is faster than Python, because the puzzle didn’t have many calculation. Python is fast if only used in a program that calculate big data because python use more RAM and GPU. And We used effolkronium library because there are no guarantees as to the quality of the random sequence produced. And we can do almost everything with random by simple 'get' method, like getting simple numbers, bools, random object from given set or using custom distribution.

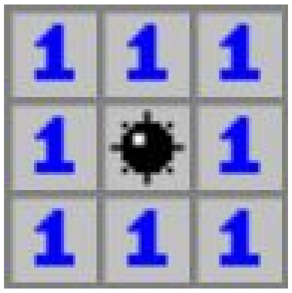
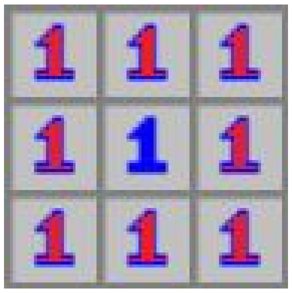
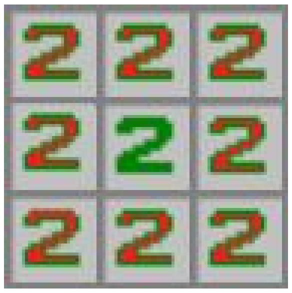
3. Compiling the Source Code

The result of this program is done on a computer with Windows 10 64 Bit Version 1709 Build 16299.192. We use Code::Blocks 17.12 program to write the code. Since Code::Blocks 17.12 didn’t have any pre-installed compiler, we using the common compiler for Code::Blocks 17.12 named MinGW-w64.

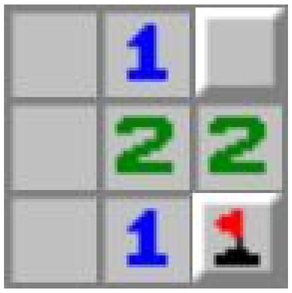
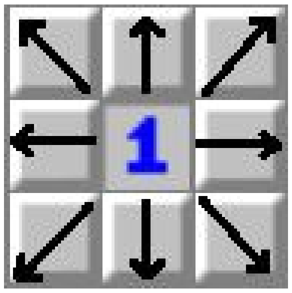
Technical Specification

1. Algorithm

Our minesweeper structure is based on 2 algorithms. The first one is probability algorithm made by Bai Li (Luckytoilet) (https://luckytoilet.wordpress.com/). The second one is obvious algorithm is a referrals from Our teaching assistant Albert Darmawan. Our probability algorithm is the modified version of Luckytoilet's probability algorithm. Probability algorithm will check all 8 direction for any unopened box, if it is empty or not flagged it will put a probability that it might be a mines. If the number is '2', '3', etc it will be increased by the original number shown.

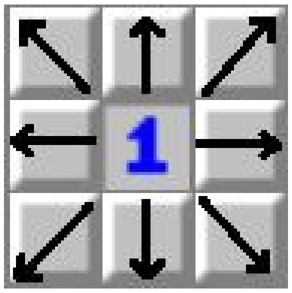




Obvious algorithm will scan 8 direction for unopened box based on the number generated from the mines

2. Algorithm Implementation

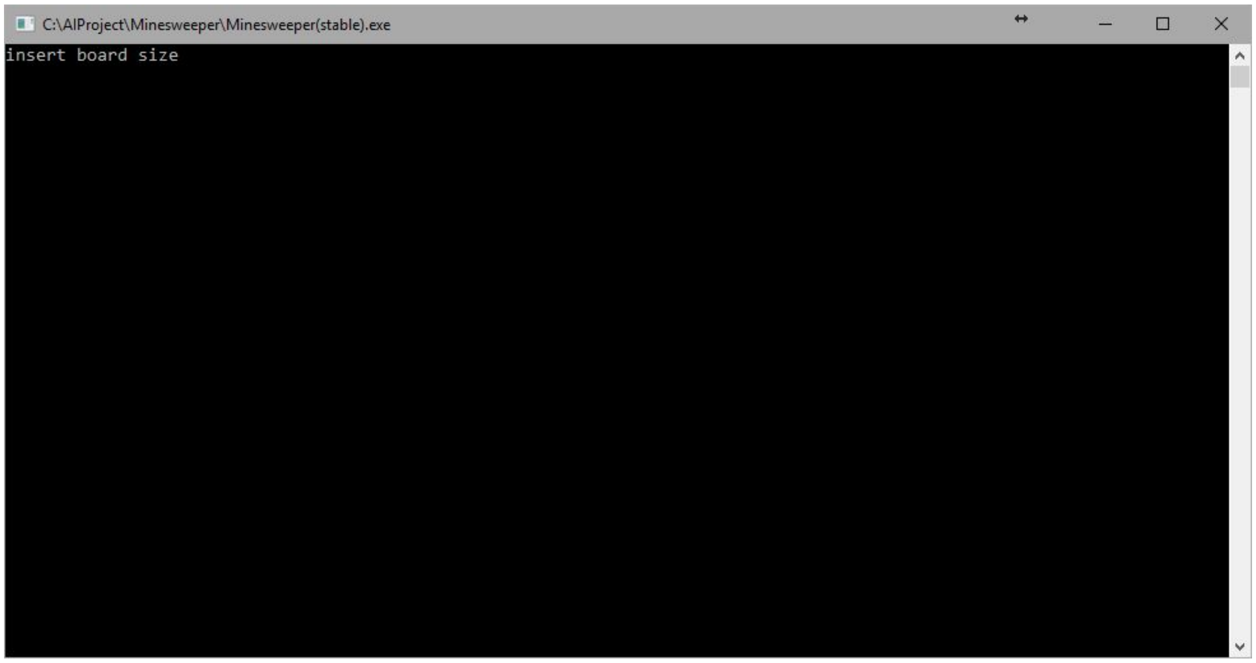
The AI will try to find the mines and flag them according to the algorithm above. The AI will stop if the mines remaining is 0 or if all mines have been flagged. The AI will not always win nor lose. This AI is only have 20%-30% win rate in every situation. This is because there is always 50%-50% win rate in every minesweeper puzzle. This AI only have 20%-30% win rate because every minesweeper AI cannot be perfect. If the AI didn't found any obvious choice after scanning the board, the AI will began to choose any unopened board from the lowest probability. If lowest and highest probability spawn in the same coordinate it 100% obviously a mines, and the AI will always flag that box in that condition.



Testing

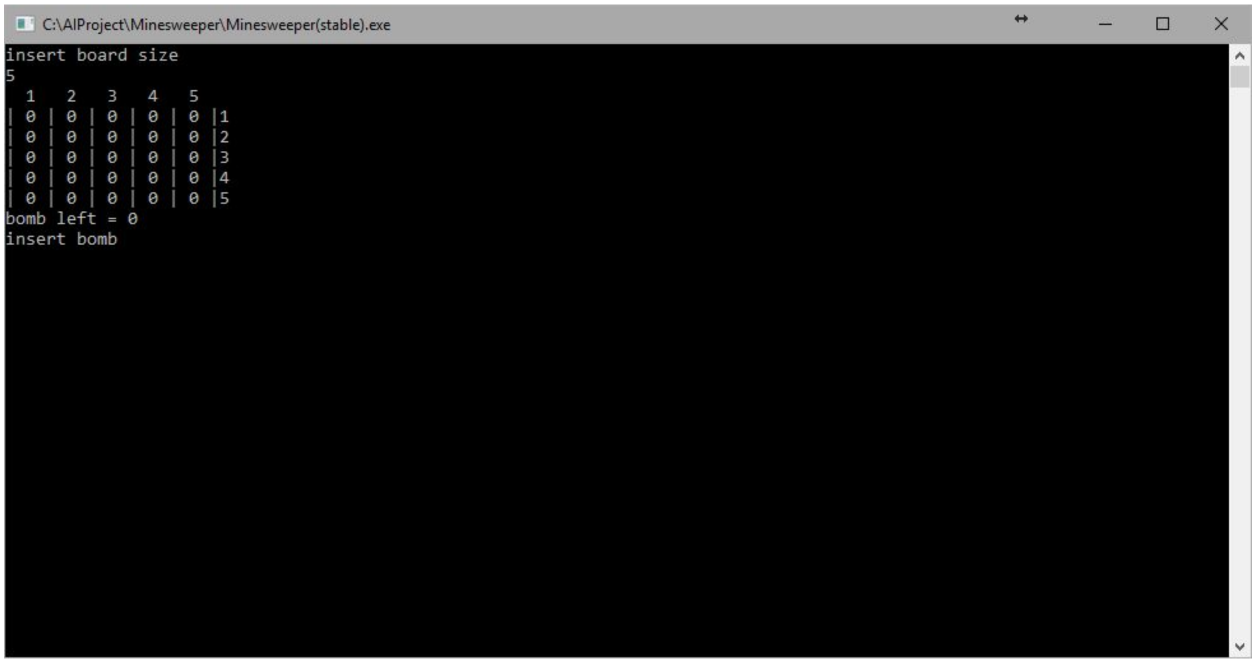
To test the Minesweeper puzzle game we can compile the source code in Code::Blocks 17.12 with MinGW-w64 compiler and run it in command prompt window.

After the program has been compiled and running, there will be a cmd window pops up like this



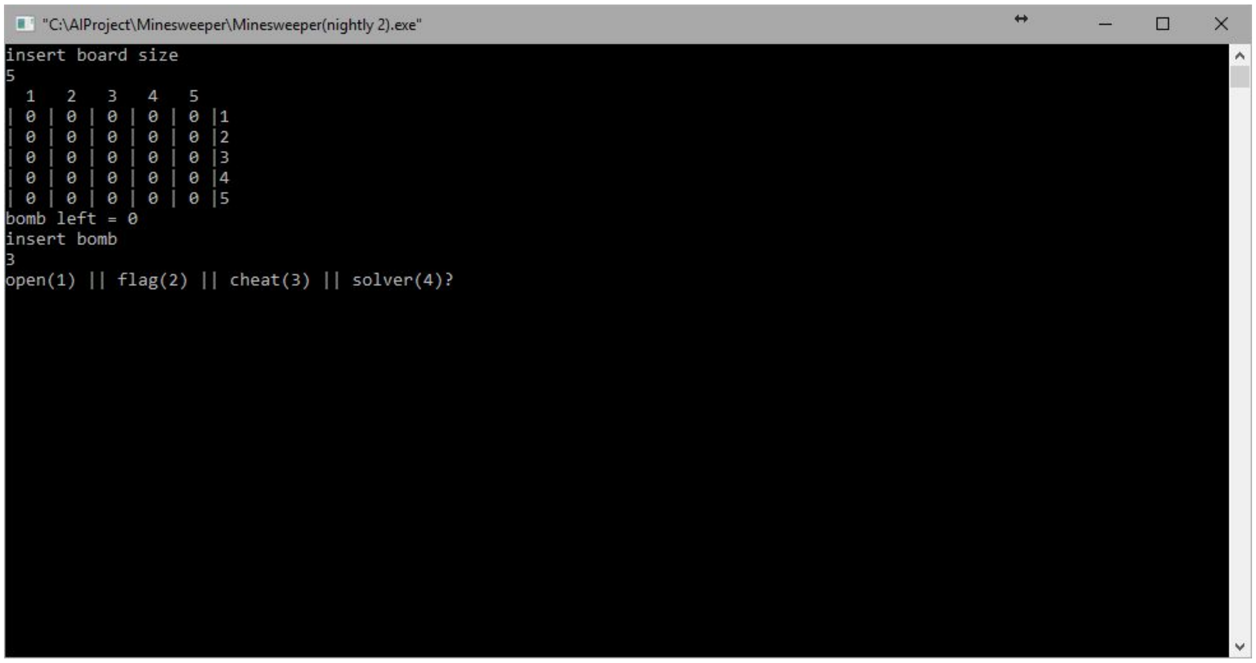
```
insert board size
```

This window is the beginning of this Minesweeper puzzle. First, the program will ask the user to input the size of the board, lets try to make a simple 5x5 board.



```
C:\AIProject\Minesweeper\Minesweeper(stable).exe
insert board size
5
  1   2   3   4   5
| 0 | 0 | 0 | 0 | 0 |1
| 0 | 0 | 0 | 0 | 0 |2
| 0 | 0 | 0 | 0 | 0 |3
| 0 | 0 | 0 | 0 | 0 |4
| 0 | 0 | 0 | 0 | 0 |5
bomb left = 0
insert bomb
```

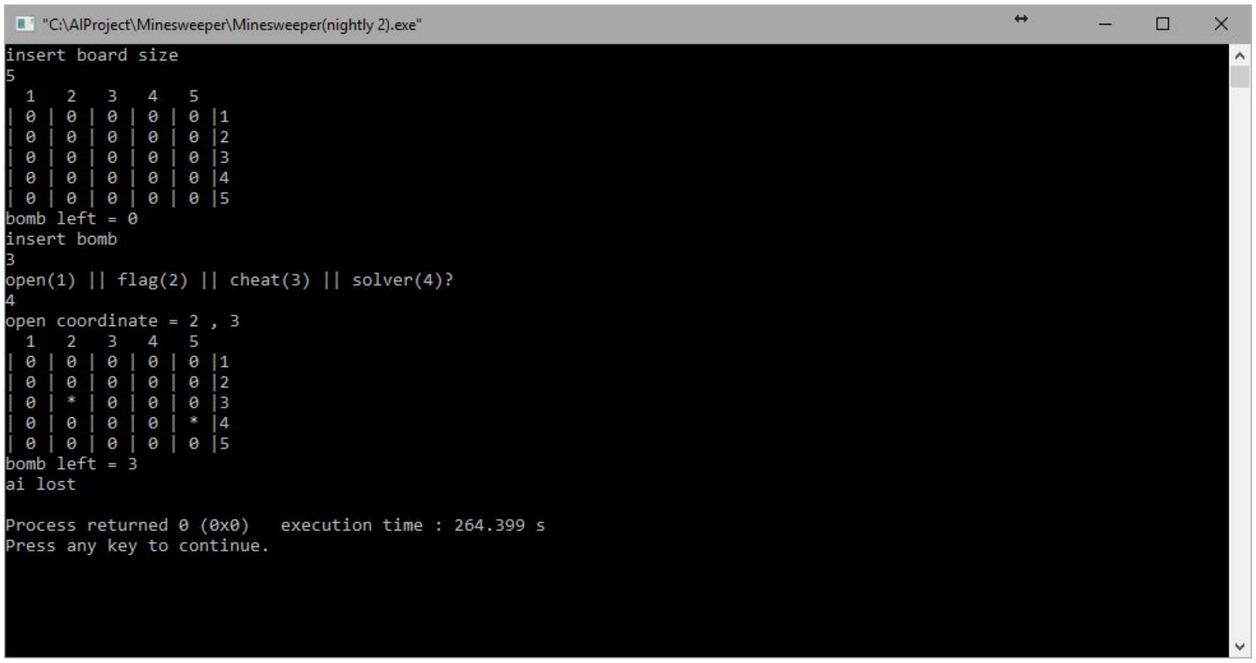
After the board has been printed now the program will ask the user how many mines will be planted inside the board randomly. In this case we will try to add 3 mines.



```
"C:\AIProject\Minesweeper\Minesweeper(nightly 2).exe"
insert board size
5
  1   2   3   4   5
| 0 | 0 | 0 | 0 | 0 |1
| 0 | 0 | 0 | 0 | 0 |2
| 0 | 0 | 0 | 0 | 0 |3
| 0 | 0 | 0 | 0 | 0 |4
| 0 | 0 | 0 | 0 | 0 |5
bomb left = 0
insert bomb
3
open(1) || flag(2) || cheat(3) || solver(4)?
```

And then there is a choice selection if the user want to (1)open, (2)flag, (3) cheat, or (4)solver

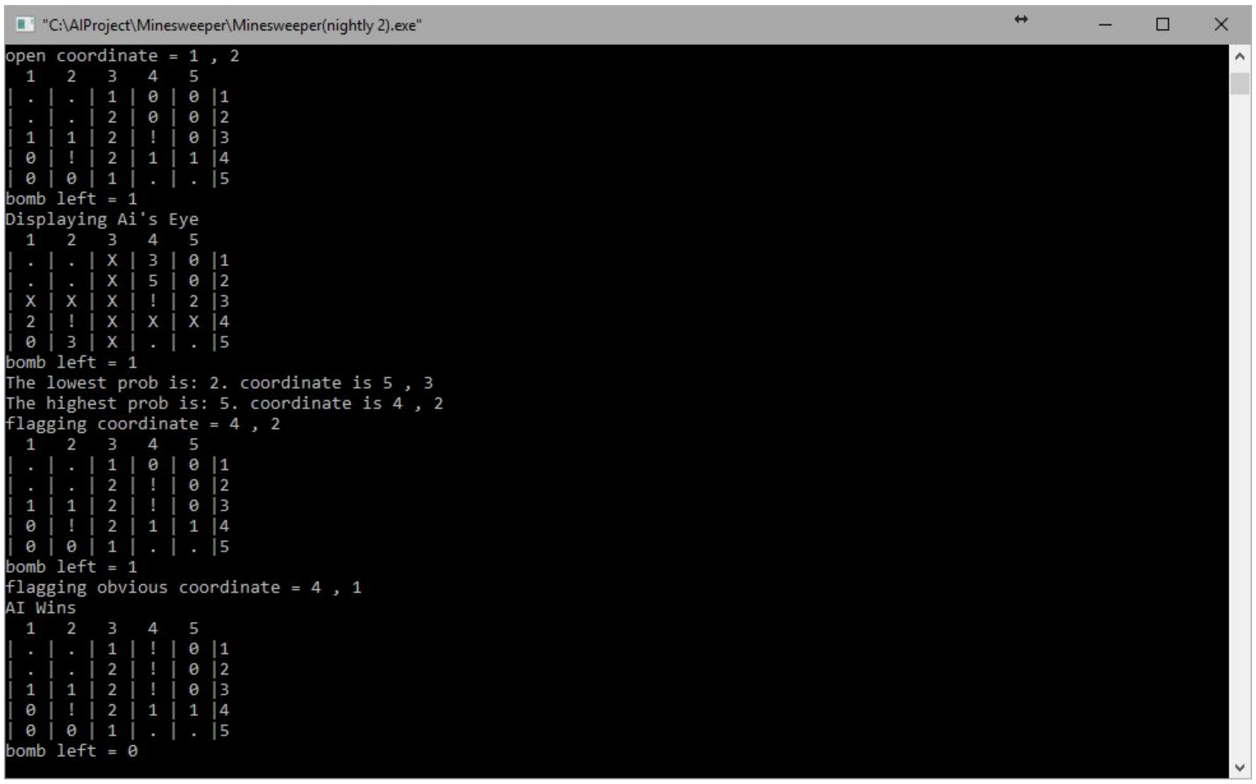
(1)Open	**=**	**User plays**
(2)Flag	**=**	**Flag the probability mines contained board with '!' symbol**
(3)Cheat	**=**	**Shown the place of the mines all over the board**
(4)Solver	**=**	**The AI of this puzzle game**



```
insert board size
5
  1   2   3   4   5
| 0 | 0 | 0 | 0 | 0 |1
| 0 | 0 | 0 | 0 | 0 |2
| 0 | 0 | 0 | 0 | 0 |3
| 0 | 0 | 0 | 0 | 0 |4
| 0 | 0 | 0 | 0 | 0 |5
bomb left = 0
insert bomb
3
open(1) || flag(2) || cheat(3) || solver(4)?
4
open coordinate = 2 , 3
  1   2   3   4   5
| 0 | 0 | 0 | 0 | 0 |1
| 0 | 0 | 0 | 0 | 0 |2
| 0 | * | 0 | 0 | 0 |3
| 0 | 0 | 0 | 0 | * |4
| 0 | 0 | 0 | 0 | 0 |5
bomb left = 3
ai lost

Process returned 0 (0x0)   execution time : 264.399 s
Press any key to continue.
```

This is the result if the AI lose. The AI choice is to open coordinate 2, 3, which is there is a mines planted in there.



"C:\AIProject\Minesweeper\Minesweeper(nightly 2).exe"

```
open coordinate = 1 , 2
  1   2   3   4   5
| . | . | 1 | 0 | 0 |1
| . | . | 2 | 0 | 0 |2
| 1 | 1 | 2 | ! | 0 |3
| 0 | ! | 2 | 1 | 1 |4
| 0 | 0 | 1 | . | . |5
bomb left = 1
Displaying Ai's Eye
  1   2   3   4   5
| . | . | X | 3 | 0 |1
| . | . | X | 5 | 0 |2
| X | X | X | ! | 2 |3
| 2 | ! | X | X | X |4
| 0 | 3 | X | . | . |5
bomb left = 1
The lowest prob is: 2. coordinate is 5 , 3
The highest prob is: 5. coordinate is 4 , 2
flagging coordinate = 4 , 2
  1   2   3   4   5
| . | . | 1 | 0 | 0 |1
| . | . | 2 | ! | 0 |2
| 1 | 1 | 2 | ! | 0 |3
| 0 | ! | 2 | 1 | 1 |4
| 0 | 0 | 1 | . | . |5
bomb left = 1
flagging obvious coordinate = 4 , 1
AI Wins
  1   2   3   4   5
| . | . | 1 | ! | 0 |1
| . | . | 2 | ! | 0 |2
| 1 | 1 | 2 | ! | 0 |3
| 0 | ! | 2 | 1 | 1 |4
| 0 | 0 | 1 | . | . |5
bomb left = 0
```

This is the result if the AI wins after finding lowest and highest probability and flagging the obvious box.

There is also a function called ‘AI’s eye’ to show how the AI see the board.

Conclusion

At this point we already know that there is no other way to improve the win rate. The algorithm uses every last piece of information available, and only fails when it's provably certain that guessing is needed.

- **Probability guessing solves it about 20%-30% of the time.**
- **Probability + Obvious guessing can rig the win rate up to 40%-50% most of the time.**

Bibliography

https://github.com/luckytoilet/MSolver

https://luckytoilet.wordpress.com/2012/12/23/2125/

https://luckytoilet.wordpress.com/about/

http://play-minesweeper.com/