



PP: Online Gaming

# Let's play Minesweeping

Hongjin Zhu & Jialin Li  
hz2291@nyu.edu  
jl10897@nyu.edu



# Rules Recap

How do we play minesweeping game in the childhood?



## Input

User clicks the left/right of the mouse (input 0/1 in our system), start sweeping!

## Position + Dig

If there's no mine, then search through the surrounding 8 boxes, if no mine again, search till the non-zero number exist.

## Position + Dig

If there is a mine, GAME OVER!

## Position + Place Flag

Place a flag at the input position/



# Video Demo



<https://www.bilibili.com/video/BV19h411Z73c>



# Core Code of Minesweeping

The core code we use to implement the Minesweeping

## Single.py

The main body of minesweeping, including functions for "accepting input", "judging result", "setting board"

```
Single.py X
C: > Users > gzLij > Desktop > Solutions for UP3 > Single.py > ...
1  import random
2  from translate_board import encrypted_board
3
4  # set up the board, randomly placing the mines
5  > def set_board(r, c, n): ...
21
22  class Solution:
23  >   def __init__(self, board, click, numFlags, endGame = False): ...
28
29  >   def updateBoard(self): ...
75  >   def set_click(self, new_click): ...
77  >   def get_flag(self): ...
79  >   def get_board(self): ...
81  >   def get_game_status(self): ...
83
84  >   def play(self): ...
102
103  # original settings (Will be commented out in online mode)
104  row = 16
105  column = 16
106  num_of_mines = 16
107  Flags = num_of_mines
108  bo = set_board(row, column, num_of_mines)
109
110
111
112  a = Solution(bo, [], Flags)
113  a.play()
```

# Core Code of Minesweeping

The core code we use to implement the Minesweeping Game

## Single.py

Recursion

```
def dfs(self,i, j):
    direction = ((1, 0), (-1, 0), (0, 1), (0, -1), (1, 1), (-1, 1), (-1, -1), (1, -1))
    self.m, self.n = len(self.board), len(self.board[0])

    cnt = 0
    for x, y in direction:
        x, y = x + i, y + j
        if 0 <= x < self.m and 0 <= y < self.n and self.board[x][y] == 'M':
            cnt += 1

    if cnt == 0:
        self.board[i][j] = 'B'
        for x, y in direction:
            x, y = x + i, y + j
            if 0 <= x < self.m and 0 <= y < self.n and self.board[x][y] == 'E':
                self.dfs(x, y)
    else:
        self.board[i][j] = str(cnt)
```

# Core Code of Minesweeping

The core code we use to implement the Minesweeping

## Translate\_board.py

A side-function designed for clients' view. Hide the locations of mines from client by using another set of representation

```
translate_board.py X
C: > Users > gzLij > Desktop > Solutions for UP3 > translate_board.py
1  import copy
2  '''这个file用来加密每个需要发给用户的board,使得雷对于他们不直接可见'''
3  # Printing the Minesweeper Layout
4  def encrypted_board(lst):
5      llst = copy.deepcopy(lst)
6      string = ''
7      for i in range(len(llst)):
8          for j in range(len(llst[i])):
9              if llst[i][j] == 'E':
10                 llst[i][j] = '|_|'
11             elif llst[i][j] == 'M':
12                 llst[i][j] = '|_|'
13             elif llst[i][j] == 'e':
14                 llst[i][j] = '|F|'
15             elif llst[i][j] == 'X':
16                 llst[i][j] = '|F|'
17             elif llst[i][j] == 'm':
18                 llst[i][j] = '|X|'
19             elif llst[i][j] == 'B':
20                 llst[i][j] = '|O|'
21             else:
22                 llst[i][j] = '|' + llst[i][j] + '|'
23
24         for n in llst:
25             string_col = ''
26             for m in n:
27                 string_col += m
28             string += string_col + '\n'
29
30     return string
31
```

# Chat\_Server.py

Memorize the game information set up in the first round (initiation)

Collect and distribute information back and forward

```
if msg["message"] == 'g':
    row = 16
    column = 16
    num_of_mines = 16
    Flags = 16
    bo = set_board(row, column, num_of_mines)
    self.play_once = Solution(bo, Flags, [])
    board_translated = encrypted_board(self.play_once.get_board())
    msg["message"] += '\nCurrent Board Looks like this: \n'
    msg["message"] += '\n' + board_translated + '\n'
    msg["message"] += 'next player please input a clicker'

else:
    try:
        clicker = msg["message"].split(',')
        for i in range(len(clicker)):
            clicker[i] = int(clicker[i])
        a = self.play_once
        a.set_click(clicker)
        if a.get_game_status() == False:
            board_translated = encrypted_board(a.updateBoard())
            flags_left = a.get_flag()
            msg["message"] += "\nCurrent Board Looks like this: \n"
            msg["message"] += board_translated
            msg["message"] += "\nNumber of flags remained: " + str(flags_left)
        else:
            msg["message"] += "\nGame Over!"
    except:
        msg["message"] += " "
```

initializing  
game

play game

## Core Code in Connecting

The main changes we made in the chat\_system structure



# Chat\_Server.py

THE SERVER PLAYS THE GAME!

(instead of the client)

---Same for other online platforms

# Client\_State\_Machine.py

We plan to build our game under the S\_Chating State, hoping to increase interaction between the players.

The click that the user inputs during the game can also be found in the chat-history.



# Core Code in Connecting

The main changes we made in the chat\_system structure





# Class Knowledge Application



## OOP

Visualize your competitive advantages using a quadrant for easy scanning.

## Recursion - Single.py

Each round when searching for the boundary, we use recursion to ensure finding a box with a number that is not zero.

## Utilize Class/ Function from other Python files

```
Import class from file.py  
from file.py import *
```

# Problems Encountered



## Being confused by local/global variables

OOP Nested function, clarifying the scopes

## Variable changes

Solved Using OOP in chat\_server.py

## Multi-player Display

Manipulate the "msg += " statement, accept any input, use the "try...Except" statement to avoid invalid input

## "User Interface"

encrypting the message sent to users



# Thank You!

Questions Welcomed!

