CSE 231
Fall 2008

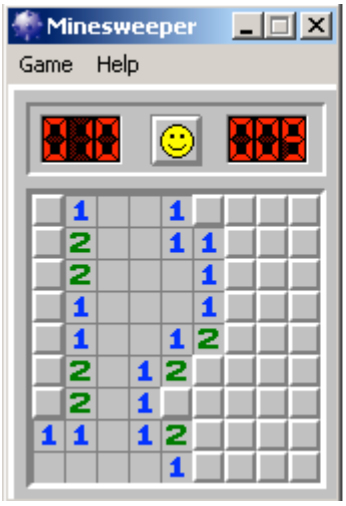
# Programming Project 10

This assignment is worth 60 points (6% of the course grade) and is a little different in that it has 2 parts, a skeleton file which must be turned in by 11:59pm, Monday 11/10 (worth 15 points), and the completed program which must be turned in by 11:59pm, Monday 11/17 (worth 45 points).

The purpose of this project is to work with classes i.e. object oriented programming as opposed to structured programming using plain functions that we have been doing over past couple of projects.

## Assignment Overview

Minesweeper is a popular computer game that comes with Windows Operating System.
You can find the game on a Windows OS at: (Start => Games => Minesweeper)





**How to play?**
Given to the player is a square grid consisting of "n x n" hidden cells. Any hidden cell contains one of the following:
a) A Mine
b) Blank cell
c) A number (say X) indicating that the cell has X mines in its neighborhood. Note that a blank cell could be represented by number “0”.

The mines are randomly placed on the grid.

The rules of the game are as follows:
1) If the player chooses a cell that has a hidden mine, the game is over. Explode!

2) If the player chooses a cell that has a hidden number, then the spot is uncovered and the number is now visible to the player.
3) If the player chooses a cell that doesn't have a mine or number, then ripple effect takes place (see section "Ripple effect" below) uncovering all the blanks in the neighborhood until either the grid boundary is reached or a number is reached along its path.
The objective of the game is to identify, logically, all the mines on the grid.

## Task

Your task is to implement the game in Python using classes. Before implementing the game, please play the game at least 5 times. It would help you understand the project.

## Program Specifications:

1) To reduce the complexity of the program, let us fix the size of the grid to be "5x5" and let the number of mines on the grid be 5.
2) On the grid, the blank cells can be represented with a number 0; indicating that the cell has zero mines in its neighborhood. ***The neighborhood of a cell consists of all the cells that are immediately adjacent to the given cell horizontally, vertically or diagonally.***
3) Program should have all error checks like,
    a) If a user chooses a cell that's not on the grid, report an error and ask the user for a valid cell.
    b) If a user gives non-integer inputs for the row and column positions, notify the user that the input is incorrect and prompt again.

    Any others that you could think of…
4) If the player gives the position of a cell on the grid that's already open, do nothing.
5) Part of the grade on this project will be the appropriateness of your class, methods, and any functions you use. The quality of the code will now matter as well as the performance. No skeleton file is provided, you need to come up with this yourself. Check previous project skeleton files as examples.

***High Level Algorithm:***

1) Create a class for the Minesweeper game. Within the class, create a "n x n" data structure to represent the grid. You will use a list of lists for this.
2) Randomly place the mines on the grid. For placing the mines, you could have a member function called placeMines().
After that, get the counts for each non-mine cell on the grid. You could have a member function called getCounts() for this.

At this stage, the grid ("Actual Grid" below) is ready with the mines and numbers, and the player could start playing.

Actual Grid

```
0  1  1  1  1
1  M  1  2  M
3  3  2  2  M
M  M  1  1  1
2  2  1  0  0
```

Player View

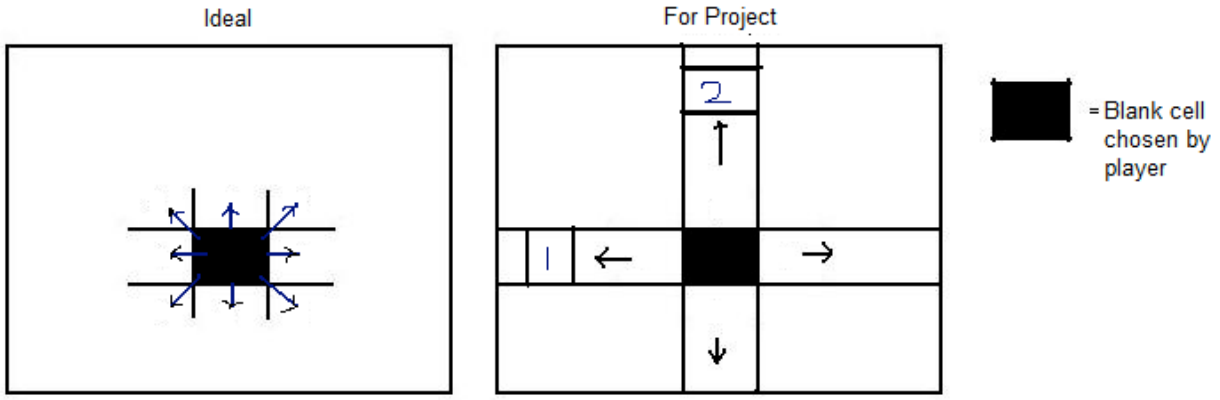
```
H  H  1  0  1
1  H  1  0  H
H  H  2  0  H
H  H  1  0  1
H  H  1  0  0
```

3) Display the grid in the form of the “Player View”, in the beginning it will all be H’s, and as the player uncovers cells you will update this view to reflect that, include row and column numbers to make it easier for the user to play.
4) Prompt the player for the position (row,column) on the grid that he/she wants to explore. Note the rows are 1-5 from top to bottom, the columns are 1-5 from left to right.
5) If the player hits a mine, give a message that a mine was hit and the game is over.
6) If the player hits a number, just show the number.
7) If the player hits a blank, then show the ripple effect (see section "Ripple effect" below)
8) Repeat steps 3-7 until game is over (i.e. all the mine locations have been successfully located) or the player has lost.
9) Ask if the player would like to play again, if so continue, otherwise, exit.

Steps 3-7 could be coded in the main() function of the program.

**Ripple effect:**
**Case 1 (CSE-231 requirement ) – This is expected in your program:**
If the player chooses a blank cell on the grid, consider only the row and column to which the cell belongs. Starting from the cell, move horizontally and vertically away from the cell until you encounter a number or the edge of a grid, whichever comes first. If a number is encountered, it should be uncovered and then the ripple effect should stop.

**Case 2(Ideal case): NOT EXPECTED – but think about it for extra credit.**
In the ideal case, all the neighbors of a cell should be considered. Then, for each of the neighbors, again, all the 8 neighbors should be considered, so on. Additionally, the boundary conditions also have to be dealt with i.e. when we reach en edge or corner of the grid during the ripple effect.




## Deliverables
You must use Handin to turn in 2 files; proj10Skel.py (due 11/10) and proj10.py (due 11/17); be sure to include your section, the date, the project number and comments describing your code. Please be sure to use the specified file name, and save a copy of your proj10.py file to your H: drive as a backup.
You will complete this project in 2 parts:

1. A skeleton, this includes your class(es) definition(s), your methods (just stubs, not filled in with code), and an outline of how you are going to use your class and methods in the main program (this can be just comments).
2. Your final solution, with your skeleton filled in with the proper code, you may change your skeleton a bit as you realize some things will work better another way, but your final program should generally follow your skeleton.

## Tips/Hints:

1) As a starting point, identify the variables and methods (also called as member functions) that you could have in the Minesweeper class. This is called the design of a class. If you get the design right, implementation is just a matter of time (and some skill, of course). The design is what you turn in in the skeleton file.
2) There are 2 views of the grid in this game. One view is the view that the player would be seeing and the second view is the complete view of the grid with mines and numbers. Think about how you are going to implement these 2 views.
eg.

| Actual Grid | | | | Player View | | | |
|---|---|---|---|---|---|---|---|
| 0 | 1 | 1 | 0 | H | H | 1 | 0 |
| 1 | M | 1 | 0 | 1 | H | 1 | 0 |
| 3 | 3 | 2 | 0 | H | H | 2 | 0 |
| M | M | 1 | 0 | H | H | 1 | 0 |

[Hint: weiv hcae rof eno ,stsil fo stsil owt esU]

## *Extra credit (10 points):*

Modify the function that implements ripple effect to make it work for the ideal case mentioned above.

Two ways to do this:

1) Using recursion. Make recursive calls on all the 8 neighbors. Recursion is a useful technique in which a function calls itself to perform a particular operation (or set) on a different object each time. A recursive function should have a *stopping* condition; otherwise the program will go into infinite loop.
2) Using a queue and doing a Breadth First Search.

Just FYI, this is similar to color fill algorithm used in computer graphics for filling polygons with some color, pixel by pixel.