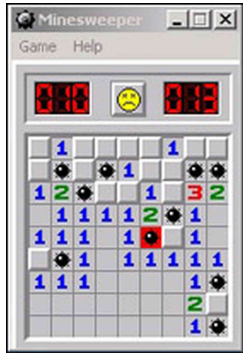


Minesweeper was one of the first games available on Windows.



An ADT for a Minesweeper field:

- `Field(nrows, ncols, nbombs)`
Create field and spread bombs.
- `display(show_bombs)` Display the field (with or without showing the bombs).
- `cell(row, col)` Get contents of cell.
- `uncover(row, col)` Open a cell.
Returns True if there was a bomb.
- `mark(row, col)` Mark a cell.
- `all_visible()` Are all cells marked or visible?
- `num_marks()` Return number of marks on the field.

Creating the field:

```
self._field = [ None ] * nrows
for i in range(nrows):
    self._field[i] = [ '.' ] * ncols
```

Reading a cell:

```
return self._field[row][col]
```

Setting a cell:

```
self._field[row][col] = el
```

There are two strategies for implementing a higher-dimensional array.

- Make one array/List for the rows. Each slot is a reference to one array/List for the columns.
- Make a large array/List with `nrows * ncols` slots.

Creation:

```
self._field = [ '.' ] * (nrows * ncols)
```

Reading a cell:

```
return self._field[row * self._ncols + col]
```

Setting a cell:

```
self._field[row * self._ncols + col] = el
```

For two-dimensional fields, the difference between the two techniques is not that large.

Lists of lists have nicer code for accessing elements, are slightly faster, but need a bit more memory.

For three- and more-dimensional fields, the “one long list” method is better, because it avoids wasting a lot of space. One can speed up accessing elements by precomputing the factors for accessing the dimensions.