CSC466 Project Task 5

Random Player Creation

Abstract

In this task I implemented a function to play a game of minesweeper randomly. In order to do this, I added a field to the board class to keep track of revealed tiles. I created a list of unrevealed tiles from taking the set difference of all of the tiles and all of the revealed tiles. I then chose a random tile from this list and revealed it. I then created a function with the purpose of playing n number of games and reporting the statistics of these games.

Demo

```
[1]> ( load "demos.l" )

;; Loading file demos.l ...

;;  Loading file hms.l ...

;;   Loading file ms.l ...

;;   Loaded file ms.l

;;  Loaded file hms.l

;; Loaded file demos.l

T
```

```
[2]> ( demo--random-game )
>>> Testing random game player
Playing game on easy with display option

      A B C D E F G H I J
    +----------------------+
  0 | # X # # # # X 2 1 0 |
  1 | # X # # # # # X 1 0 |
  2 | # # 1 # # # 1 1 1 0 |
  3 | # # # # X 2 1 0 0 0 |
  4 | # 2 # # # X 1 0 0 0 |
  5 | X 2 X # X 2 1 0 0 0 |
  6 | 1 2 1 2 1 1 0 0 0 0 |
  7 | 0 0 0 0 0 0 0 0 0 0 |
  8 | 0 1 1 1 0 0 0 0 0 0 |
  9 | 0 1 X 1 0 0 0 0 0 0 |
    +----------------------+

Playing game on easy with stats option
(73 100)

Playing 100 games on easy
Average number of tiles revealed: 60
Number of wins: 1
```

```
Playing 100 games on medium
Average number of tiles revealed: 52
Number of wins: 0

Playing 100 games on hard
Average number of tiles revealed: 25
Number of wins: 0
NIL
[3]>
```

Code

```
( defclass board ()
    (
        ( width :accessor board-width :initarg :width )
        ( height :accessor board-height :initarg :height )
        ( mines :accessor board-mines :initarg :mines )
        ( tiles :accessor board-tiles :initarg :tiles )
        ( revealed-tiles :accessor board-revealed-tiles :initform '() )
    )
)
( defun play-random-game ( &optional mode difficulty &aux move )
    ( cond
        ( ( or ( null difficulty ) ( equal difficulty 'easy ) )
            ( generate-board 10 10 10 ) )
        ( ( equal difficulty 'medium ) ( generate-board 16 16 40 ) )
        ( ( equal difficulty 'hard ) ( generate-board 24 20 99 ) )
    )
    ( loop
        ( if ( or ( win-p ) ( reveal-random ) )
            ( cond
                ( ( equal mode 'display ) ( display-board ) ( return nil ) )
                ( ( equal mode 'stats )
                    ( return ( list ( length ( board-revealed-tiles *board* ) )
                        ( length ( board-tiles *board* ) ) ) )
                )
                ( t ( return nil ) )
            )
        )
    )
)

( defun reveal-random ( &aux tiles revealed unrevealed  )
    ( setf tiles ( board-tiles *board* ) )
    ( setf revealed ( board-revealed-tiles *board* ) )
    ( setf unrevealed ( set-difference tiles revealed ) )
    ( reveal-tiles ( list ( nth ( random ( length unrevealed ) ) unrevealed ) )
'() )
)
```

```
( defun play-n-random ( n &optional difficulty &aux result revealed total wins )
    ( setf revealed 0 )
    ( setf total 0 )
    ( setf wins 0 )
    ( dotimes ( i n )
        ( setf result ( play-random-game 'stats difficulty ) )
        ( setf revealed ( + revealed ( car result ) ) )
        ( setf total ( + total ( cadr result ) ) )
        ( if ( win-p ) ( setf wins ( + 1 wins ) ) )
    )
    ( format t "Average number of tiles revealed: ~a~%Number of wins: ~a~%" (
floor revealed n ) wins )
)
```