

Harry Scells
Craig Graci
30 January 2015

Ttt Lm: Visualisation, Analysis And Statistics

Code	2
Demo	4
Results	5

Code

This code outlines the two methods that were implemented.

```
1. ; get the position of player in a game
2. (defun pos (obj lst)
3.   (loop for i in lst and position from 0 when (eql i obj) return
         position)
4. )
5.
6. ; get the player name from the play made
7. (defun playern (num)
8.   (cond ((eq 0 (mod num 2)) 'X) (t 'O))
9. )
10.
11.; draws a single row of the board
12.(defun drawrow (moves lst)
13.  (dolist (move moves)
14.    (format t "~S~S " (playern(pos move lst)) (+ 1 (pos move
        lst))))
15.  )
16.)
17.
18.(defun visualize (lst)
19.  (drawrow '(nw n ne) lst)
20.  (format t "~%")
21.  (drawrow '(w c e) lst)
22.  (format t "~%")
23.  (drawrow '(sw s se) lst)
24.  (format t "~%")
25.)
26.
27.; sum the scores of a list
28.(defun sumscore (sublst lst)
29.  (setf sum 0)
30.  (dolist (l sublst)
31.    (setf sum (+ sum (pos l lst))))
32.  )
33.  (return-from sumscore sum)
34.)
```

```

35.(defun analyze (lst)
36.  ; extract out each players moves
37.  (setf playo (list (nth 1 lst) (nth 3 lst) (nth 5 lst) (nth 7
    lst)))
38.  (setf playx (list (nth 0 lst) (nth 2 lst) (nth 4 lst) (nth 6
    lst) (nth 8 lst)))
39.
40.  ; get a list of all the winning conditions
41.  (setf wcond '(
42.    (nw n ne) (e c w) (sw s se)
43.    (nw w sw) (n c s) (ne e se)
44.    (ne c sw) (nw c se)
45.  ))
46.
47.  ; set the score to the highest possible score (lower is
    better)
48.  (setf scorex 45)
49.  (setf scoreo 45)
50.
51.  ; move through the winning conditions and find the best play
52.  (dolist (w wcond)
53.    (cond
54.      ((eq (length (intersection w playx)) 3) (setf scorex
        (sumscore w lst)))
55.      ((eq (length (intersection w playo)) 3) (setf scoreo
        (sumscore w lst)))
56.    )
57.  )
58.
59.  ; determine the winner or if the game is a draw
60.  (cond
61.    ((> scorex scoreo) (return-from analyze 'l))
62.    ((< scorex scoreo) (return-from analyze 'w))
63.  )
64.  (return-from analyze 'd)
65.)

```

Demo

```
[1]> (demo-va)
(N SE NW S SW C NE W E)
X3 X1 X7
O8 O6 X9
X5 O4 O2
W
NIL
[2]> (demo-va)
(N NW SE E W SW C S NE)
O2 X1 X9
X5 X7 O4
O6 O8 X3
D
NIL
[3]> (stats 5 t)
Begin gathering statistics ...
(W S SE NE NW E N SW C)
X5 X7 O4
X1 X9 O6
O8 O2 X3
(S E NE C SE SW NW N W)
X7 O8 X3
X9 O4 O2
O6 X1 X5
(NW W SW N S C SE NE E)
X1 O4 O8
O2 O6 X9
X3 X5 X7
(E W S C NW N SW SE NE)
X5 O6 X9
O2 O4 X1
X7 X3 O8
(C NW W NE SE E N SW S)
O2 X7 O4
X3 X1 O6
O8 X9 X5
End gathering statistics
((W 0.6) (L 0.0) (D 0.4))
[4]> (stats 1000 nil)
((W 0.624) (L 0.274) (D 0.102))
```

Results

It can be seen that by having player X move first, it gives that player a much higher advantage. In having five possible moves as opposed to four, there is a much higher chance of winning by pure chance, purely due to the fact that there are more possible spaces to enter. When run over a large data set it is clear to see that this is evident.