

## Tier List Player

### Abstract

This player is a true improvement from the Random Player Plus Plus, it now organize locations into 5 tiers:

- 0: Explored tier
- 1: Avoid tier
- 2: Normal tier
- 3: Preferred tier
- 4: Critical tier

When it explores a location, that location will be moved into tier 0. And if it misses, the adjacent locations will be moved to tier 1, however if it hits something the adjacent locations will be moved to tier 3, a consecutive hit will move the next inferred location into tier 4.

### Demo

I feel that the demo should come first as it contains a lot of code.

### Manual Demo

```
[8]> (play)
Welcome to the Battleship game.
Each player have 5 ships on the board,
to win, you have to sink all of the other player's ship before it sinks all of yours.
At each turn, you will be shown a marker map on the left, and your board on the right.
When you are asked to enter a position, enter an letter for X, followed by space, and then a number for Y.
For example: B 7

Enter anything to start ...a
Available AIs:
1 - RANDOMPLAYER
2 - RANDOMPLAYERPLUS
3 - RANDOMPLAYERPLUSPLUS
4 - TIERLISTPLAYER
Choose your opponent: 4
  A  B  C  D  E  F  G  H  I  J
0 +---+---+---+---+---+---+---+---+---+
  |   |   |   |   |   |   |   |   |   |
  +---+---+---+---+---+---+---+---+---+
1 |   |   |   |   |   |   |   |   |   |
```

```

+---+---+---+---+---+---+---+---+---+---+
2 | | | | | | | | | | |
+---+---+---+---+---+---+---+---+---+---+
3 | | | | | | | | | | |
+---+---+---+---+---+---+---+---+---+---+
4 | | | | | | | | | | |
+---+---+---+---+---+---+---+---+---+---+
5 | | | | | | | | | | |
+---+---+---+---+---+---+---+---+---+---+
6 | | | | | | | | | | |
+---+---+---+---+---+---+---+---+---+---+
7 | | | | | | | | | | |
+---+---+---+---+---+---+---+---+---+---+
8 | | | | | | | | | | |
+---+---+---+---+---+---+---+---+---+---+
9 | | | | | | | | | | |
+---+---+---+---+---+---+---+---+---+---+

```

Now placing: CARRIER, size: 5

Enter position 1: a 0

Enter position 2: e 0

```

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

```

Now placing: BATTLESHIP, size: 4

Enter position 1: b 2

Enter position 2: e 2

```

      A  B  C  D  E  F  G  H  I  J
+---+---+---+---+---+---+---+---+---+---+
0 | 5 | 5 | 5 | 5 | 5 | | | | | |
+---+---+---+---+---+---+---+---+---+---+
1 | | | | | | | | | | |
+---+---+---+---+---+---+---+---+---+---+
2 | | 4 | 4 | 4 | 4 | | | | | |
+---+---+---+---+---+---+---+---+---+---+
3 | | | | | | | | | | |
+---+---+---+---+---+---+---+---+---+---+
4 | | | | | | | | | | |
+---+---+---+---+---+---+---+---+---+---+
5 | | | | | | | | | | |
+---+---+---+---+---+---+---+---+---+---+

```

```

6 | | | | | | | | | | |
+---+---+---+---+---+---+---+---+---+
7 | | | | | | | | | | |
+---+---+---+---+---+---+---+---+
8 | | | | | | | | | | |
+---+---+---+---+---+---+---+---+
9 | | | | | | | | | | |
+---+---+---+---+---+---+---+---+

```

Now placing: CRUISER, size: 3

Enter position 1: c 4

Enter position 2: e 4

```

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

```

Now placing: SUBMARINE, size: 3

Enter position 1: f 3

Enter position 2: h 3

```

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

```

Now placing: DESTROYER, size: 2

All ships have been placed.

[illegible]

It missed.

[illegible][illegible]

```
Enemy fired at I, 7
Enter target location: d 0
It missed.
```

Your markers and your board:

[illegible]

```
Enemy fired at H, 3
Enter target location: f 0
It missed.
```

[illegible]

Your markers and your board:

[illegible]

Enemy fired at H, 2

[illegible]

```
Enter target location: h 0
It missed.
```

## Your markers and your board:

[illegible]

Enemy fired at G, 3

```
Enter target location: c 1
```

It's a hit!

[illegible]

## Your markers and your board:

[illegible]

Enemy fired at F, 3

Enter target location: b 1

It's a hit!

Your markers and your board:

[illegible]

Enemy fired at E, 3

```
Enter target location: a 1
```

It missed.

[illegible]

Your markers and your board:

[illegible]

Enemy fired at H, 4

[illegible]

We can see that not only does it prioritize the inferred ship locations, it also backtracks to previous known locations.

Let's put it against other AIs:

Against random player:

```
[2]> (getStatistics)
Available AIs:
1 - RANDOMPLAYER
2 - RANDOMPLAYERPLUS
3 - RANDOMPLAYERPLUSPLUS
4 - TIERLISTPLAYER
Enter a corresponding number to choose AI 1: 1
Enter a corresponding number to choose AI 2: 4
Enter the number of iterations: 100

100 games played:
Player 1 (RANDOMPLAYER) victories: 0
Player 2 (TIERLISTPLAYER) victories: 100
Draws: 0
NIL
```

Against random player plus

```
[3]> (getStatistics)
Available AIs:
1 - RANDOMPLAYER
2 - RANDOMPLAYERPLUS
3 - RANDOMPLAYERPLUSPLUS
4 - TIERLISTPLAYER
Enter a corresponding number to choose AI 1: 2
Enter a corresponding number to choose AI 2: 4
Enter the number of iterations: 100

100 games played:
```



```
Player 1 (RANDOMPLAYERPLUS) victories: 44
Player 2 (TIERLISTPLAYER) victories: 55
Draws: 1
NIL
[4]> (getStatistics)
Available AIs:
1 - RANDOMPLAYER
2 - RANDOMPLAYERPLUS
3 - RANDOMPLAYERPLUSPLUS
4 - TIERLISTPLAYER
Enter a corresponding number to choose AI 1: 2
Enter a corresponding number to choose AI 2: 4
Enter the number of iterations: 100

100 games played:
Player 1 (RANDOMPLAYERPLUS) victories: 43
Player 2 (TIERLISTPLAYER) victories: 55
Draws: 2
NIL
```

Against random player plus plus

```
[5]> (getStatistics)
Available AIs:
1 - RANDOMPLAYER
2 - RANDOMPLAYERPLUS
3 - RANDOMPLAYERPLUSPLUS
4 - TIERLISTPLAYER
Enter a corresponding number to choose AI 1: 3
Enter a corresponding number to choose AI 2: 4
Enter the number of iterations: 100

100 games played:
Player 1 (RANDOMPLAYERPLUSPLUS) victories: 43
Player 2 (TIERLISTPLAYER) victories: 56
Draws: 1
NIL
```

Against itself

```
[6]> (getStatistics)
Available AIs:
1 - RANDOMPLAYER
2 - RANDOMPLAYERPLUS
3 - RANDOMPLAYERPLUSPLUS
4 - TIERLISTPLAYER
Enter a corresponding number to choose AI 1: 4
Enter a corresponding number to choose AI 2: 4
Enter the number of iterations: 100

100 games played:
Player 1 (TIERLISTPLAYER) victories: 40
Player 2 (TIERLISTPLAYER) victories: 57
Draws: 3
NIL
```

## Code

Reason why it's called tier list player:

```
(defclass tierListPlayer()
  (
    (name :accessor player-name :initform 'TierListPlayer)
    (thisBoard :accessor player-board :initarg :thisBoard)
    (otherBoard :accessor player-otherBoard :initarg :otherBoard)
    (ships :accessor player-ships :initarg :ships)
    (t0 :accessor player-t0 :initform '())
    (t1 :accessor player-t1 :initform '())
    (t2 :accessor player-t2 :initarg :locations)
    (t3 :accessor player-t3 :initform '())
    (t4 :accessor player-t4 :initform '())
  )
)

; Constructor
```

```

(defmethod newTierListPlayer((this board) (other board) (ships list) &aux x y
  locations)
  (setf x (board-width this))
  (setf y (length (board-rows other)))
  (setf locations (generateAllLocations x y))
  (make-instance 'tierListPlayer
    :thisBoard this
    :otherBoard other
    :ships ships
    :locations locations
  )
)
;
-----
-----

```

When choose locations, it chooses from higher tier first:

```

(defmethod getNextLocation((p tierListPlayer) &aux t1 t2 t3 t4 location)
  (setf t1 (player-t1 p))
  (setf t2 (player-t2 p))
  (setf t3 (player-t3 p))
  (setf t4 (player-t4 p))

  (setf location nil)
  (cond
    ; Pick from t4 if not empty
    ((not (equal t4 nil))
      (setf location (randomFromList t4))
    )
    ; Pick from t3 if not empty, given that t4 is empty
    ((not (equal t3 nil))
      (setf location (randomFromList t3))
    )
    ; Pick from t2 if not empty, given that t3 is empty
    ((not (equal t2 nil))
      (setf location (randomFromList t2))
    )
    ; Pick from t1 if no other options, the game should end when t1 runs out
  )
)

```

```

        (t
          (setf location (randomFromList t1))
        )
      )

      location
    )

```

Here is the upgrade:

```

(defmethod modifyLists((l location) hit (p tierListPlayer))
  (moveToTier l p 0)
  (if hit
    (doWhenHit l p)
    (doWhenNotHit l p))
  )
)

```

If not hit, move neighbors to t1 (avoid)

```

(defmethod doWhenNotHit((l location) (p tierListPlayer))
  (moveNeighborsToTier l p 1)
)

; Move all neighbors to a tier, except the explored ones.
(defmethod moveNeighborsToTier((l location) (p tierListPlayer) tier &aux
adjacents)
  (setf adjacents (getAdjacentsFromTiers l p))
  (loop for adjacent in adjacents do
    (if (isUnexplored adjacent p)
      (moveToTier adjacent p tier)
    )
  )
)
)

```

If hit

```

; If shot landed a hit
(defmethod doWhenHit((l location) (p tierListPlayer) &aux adjacent opposite)
  ; Move all unexplored (not t0) neighbors to preferred (t3) list.
  (moveNeighborsToTier l p 3)
)

```

```

; If a neighbor counts as a consecutive hit, then move the opposite neighbor
(?) into critical tier (t4).
; | X | X | ? |
(if (isConsecutiveHit (getAdjacentFromTiers 'left 1 p) p)
    (setf opposite (getAdjacentFromTiers 'right 1 p))
)
; | ? | X | X |
(if (isConsecutiveHit (getAdjacentFromTiers 'right 1 p) p)
    (setf opposite (getAdjacentFromTiers 'left 1 p))
)
; | ? |
; | X |
; | X |
(if (isConsecutiveHit (getAdjacentFromTiers 'below 1 p) p)
    (setf opposite (getAdjacentFromTiers 'above 1 p))
)
; | X |
; | X |
; | ? |
(if (isConsecutiveHit (getAdjacentFromTiers 'above 1 p) p)
    (setf opposite (getAdjacentFromTiers 'below 1 p))
)

; Move the opposite (and qualified) neighbor into critical tier (t4)
(if (isQualifiedOpposite opposite p)
    (moveToTier opposite p 4)
)
)

```

A shot is a consecutive hit if:

```

(defmethod isConsecutiveHit(location (p tierListPlayer) &aux board)
    (setf board (player-otherBoard p))

    ; This location need to:
    (and
        ; Be not null
        (not (equal location nil))
        ; Have achieved an hit
        (isLocationHit location board)
    )
)

```

```

        ; Have an unsunk ship
        (isLocationSunk location board)
    )

    ; Can't just ask if a neighbor have an unsunk ship, that would be cheating.
)

```

An opposite is qualified for critical list if

```

(defmethod isQualifiedOpposite(location (p tierListPlayer))
    ; This location need to:
    (and
        ; Be not null
        (not (equal location nil))
        ; Be unexplored
        (isUnexplored location p)
    )
)

```

The move tier function

```

; Remove this location from all tiers, then write into the desired one.
(defmethod moveToTier((l location) (p tierListPlayer) target)
    (removeFromTiers l p)
    (addToTier l p target)
)

; Remove this location from *ALL* tiers of this player.
(defmethod removeFromTiers((l location) (p tierListPlayer))
    (setf (player-t0 p) (remove l (player-t0 p)))
    (setf (player-t1 p) (remove l (player-t1 p)))
    (setf (player-t2 p) (remove l (player-t2 p)))
    (setf (player-t3 p) (remove l (player-t3 p)))
    (setf (player-t4 p) (remove l (player-t4 p)))
)

; Add this location to the desired tier of this player.
(defmethod addToTier((l location) (p tierListPlayer) target)
    (cond

```

```
((= target 0)
  (setf (player-t0 p) (cons 1 (player-t0 p)))
)
((= target 1)
  (setf (player-t1 p) (cons 1 (player-t1 p)))
)
((= target 2)
  (setf (player-t2 p) (cons 1 (player-t2 p)))
)
((= target 3)
  (setf (player-t3 p) (cons 1 (player-t3 p)))
)
(t
  (setf (player-t4 p) (cons 1 (player-t4 p)))
)
)
)
```