Kuncheng Feng
CSC 466 progress report

Play the game

Abstract

The game's basic mechanics are all in place, and now it supports a human player playing against a truly randomly playing machine.

Main playing function

The function displays some information that might be helpful to the player, sets up the game, and enters the playing loop.

```
(defun play(&aux human ai board1 board2 ships1 ships2)
    ; Some info to help the player get started.
    (format t "Welcome to the Battleship game.~%")
    (format t "Each player have 5 ships on the board,~%")
    (format t "to win, you have to sink all of the other player's ship
before it sinks all of yours.~%")
    (format t "At each turn, you will be shown a marker map on the left,
and your board on the right.~%")
    (format t "When you are asked to enter a position, enter an letter
for X, followed by space, and then a number for Y.~%")
    (format t "For example: B 7~%~%")
    (format t "Enter anything to start ...")
    (read)

    (setf board1 (newBoard 10 10))
    (setf board2 (newBoard 10 10))
    (setf ships1 (reverse (generateShips)))
    (setf ships2 (reverse (generateShips)))

    (setf human (newHumanPlayer board1 board2 ships1))
    (setf ai (newRandomPlayer board2 board1 ships2))

    (playerPlaceShips human)
    (playerPlaceShips ai)
```

```
    (takeTurn human ai)
)
```

(generateShips) creates 5 instances of ships, one of each type.

```
(defun generateShips(&aux ships)
    (setf ships (list))
    (loop for ship in *shipTypes* do
        (setf ships (cons (newShip ship) ships))
    )
    ships
)
```

The HumanPlayer class and the RandomlyPlayer keep the same values, but their methods will be different.

```
(defclass humanPlayer()
    (
        (thisBoard :accessor player-board :initarg :thisBoard)
        (otherBoard :accessor player-otherBoard :initarg :otherBoard)
        (ships :accessor player-ships :initarg :ships)
    )
)


; Constructor ----------------------------------------------------------
(defmethod newHumanPlayer((this board) (other board) (ships list))
    (make-instance 'humanPlayer
        :thisBoard this
        :otherBoard other
        :ships ships
    )
)
; ---------------------------------------------------------------------
```

```
(defclass randomPlayer()
    (
        (thisBoard :accessor player-board :initarg :thisBoard)
        (otherBoard :accessor player-otherBoard :initarg :otherBoard)
        (ships :accessor player-ships :initarg :ships)
    )
)


; Constructor ----------------------------------------------------------
(defmethod newRandomPlayer((this board) (other board) (ships list))
    (make-instance 'randomPlayer
        :thisBoard this
        :otherBoard other
        :ships ships
    )
)
; ---------------------------------------------------------------------
```

Placing ships

The HumanPlayer will ask for user input, while the other will randomly generate it.

```
(defmethod playerPlaceShips((player humanPlayer) &aux board ships)
    (setf board (player-board player))
    (setf ships (player-ships player))

    ; Loop through each ship and have them placed.
    (loop for ship in ships do
        (humanPlaceShip ship board)
    )

    (format t "All ships have been placed.~%~%~%")
)

; Note that this method repeats until success.
(defmethod humanPlaceShip((s ship) (b board) &aux shipType x1 y1 x2 y2)
    (display b)
```

```
    (setf shipType (ship-type s))
    (format t "Now placing: ~A, size: ~A~%" shipType (get '*shipSize*
shipType))
    (format t "Enter position 1: ")
    (setf x1 (read))
    (setf x1 (letterToCell x1))
    (setf y1 (read))
    (format t "Enter position 2: ")
    (setf x2 (read))
    (setf x2 (letterToCell x2))
    (setf y2 (read))

    (if (checkValid x1 y1 x2 y2 s b)
        (placeShip x1 y1 x2 y2 s b)
        (humanPlaceShip s b)
    )
)
```

```
(defmethod playerPlaceShips((player randomPlayer) &aux board ships)
    (setf ships (player-ships player))
    (setf board (player-board player))

    ; Loop through the ships that have to be placed
    (loop for ship in ships do
        (randomlyPlaceShip ship board)
    )
)

(defmethod randomlyPlaceShip((s ship) (b board) &aux pos)
    (setf pos (getRandomPosition s b))
    (placeShip (first pos) (second pos) (third pos) (fourth pos) s b)
)

(defmethod getRandomPosition((s ship) (b board) &aux size maxX maxY x1
y1 x2 y2)
    ; Note! Ship positioning is inclusive, the -1 is needed here!
    (setf size (- (get '*shipSize* (ship-type s)) 1))
```

```
    ; Ship is either placed (left to right) or (small Y to big Y).
    (setf maxX (- (board-width b) size))
    (setf maxY (- (length (board-rows b)) size))

    (loop
        (setf x1 (random maxX))
        (setf y1 (random maxY))

        ; Roll a number of either 0 or 1;
        ; 0 will result in horizontal placement,
        ; 1 will result in vertical placement.
        (cond
            ((= (random 2) 0)
                (setf x2 (+ x1 size))
                (setf y2 y1)
            )
            (t
                (setf x2 x1)
                (setf y2 (+ y1 size))
            )
        )

        (when
            (equal (checkValidNoText x1 y1 x2 y2 s b) t)
            (return (list x1 y1 x2 y2))
        )
    )
)
```

Each player takes a turn

```
(defun takeTurn(player1 player2 &aux p1Win p2Win)
    (playerOpenFire player1)
    (playerOpenFire player2)

    (setf p1Win (isPlayerDefeated player2))
    (setf p2Win (isPlayerDefeated player1))

    (cond
        ((and p1Win p2Win)
            (format t "It's a draw.~&")
        )
        (p1Win
            (format t "You won!~%")
        )
        (p2Win
            (format t "You lost!~%")
        )
        (t
            (takeTurn player1 player2)
        )
    )
)
```

Human player open fire

```
(defmethod playerOpenFire((p humanPlayer) &aux myBoard enemyBoard pos
cell ship)
    (setf myBoard (player-board p))
    (setf enemyBoard (player-otherBoard p))

    (format t "Your markers and your board: ~%")
    (displayBoth myBoard enemyBoard)

    ; Give a little feedback on where the enemy recently fired.
    (setf pos (board-recent myBoard))
    (if (not (equal pos nil))
        (format t "Enemy fired at ~A, ~A~%" (cellToLetter (first pos))
```

```
(second pos))
    )

    (setf pos (getPlayerInput enemyBoard))
    (fireAtBoard (first pos) (second pos) enemyBoard)

    ; Give a little feedback on the result of recent firing.
    (setf cell (getCell (first pos) (second pos) enemyBoard))
    (setf ship (cell-resident cell))
    (if (isCellHit cell)
        ; If a cell is hit, it have a residence.
        (if (isShipSunk ship)
            (format t "You sunk a ~A.~%" (ship-type ship))
            (format t "It's a hit!~%")
        )
        (format t "It missed.~%")
    )
)

(defmethod getPlayerInput((b board) &aux x y)
    (format t "Enter target location: ")
    (setf x (read))
    (setf x (letterToCell x))
    (setf y (read))

    (cond
        ((checkBorder x y b)
            (list x y)
        )
        (t
            (format t "Position out of bound.~%")
            (getPlayerInput b)
        )
    )
)
```

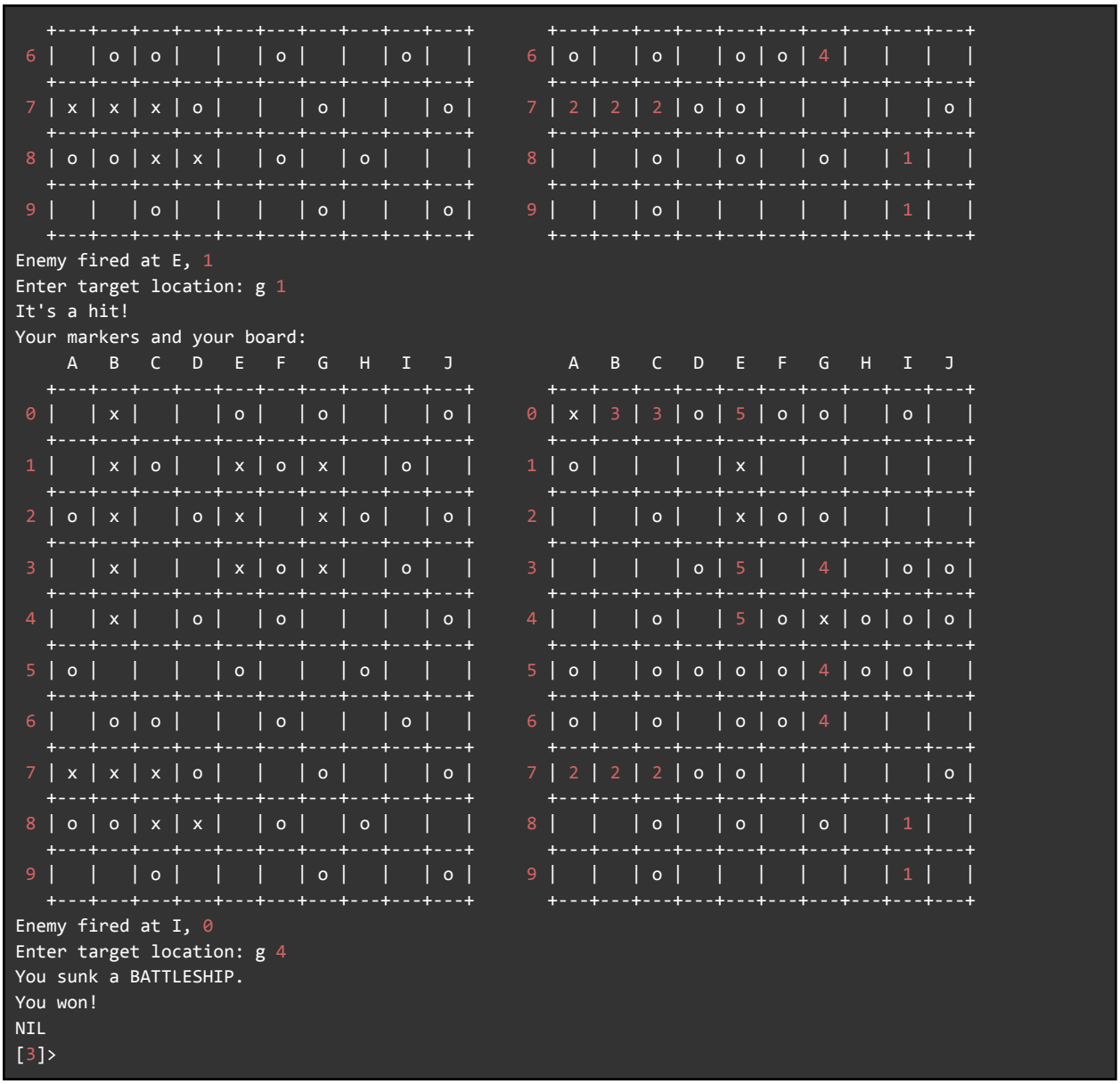
Random player open fire

```
(defmethod playerOpenFire((p randomPlayer) &aux enemyBoard x y)
    (setf enemyBoard (player-otherBoard p))
    (setf x (random (board-width enemyBoard)))
    (setf y (random (length (board-rows enemyBoard))))
    (fireAtBoard x y enemyBoard)
)
```

Demo

Since the game turned out to have a lot of text, only the last bit is shown.

```
Your markers and your board:
    A   B   C   D   E   F   G   H   I   J          A   B   C   D   E   F   G   H   I   J
  +---+---+---+---+---+---+---+---+---+---+      +---+---+---+---+---+---+---+---+---+---+
0 |   | x |   |   | o |   | o |   |   | o |    0 | x | 3 | 3 | o | 5 | o | o |   | o |   |
  +---+---+---+---+---+---+---+---+---+---+      +---+---+---+---+---+---+---+---+---+---+
1 |   | x | o |   | x | o |   |   | o |   |    1 | o |   |   |   | x |   |   |   |   |   |
  +---+---+---+---+---+---+---+---+---+---+      +---+---+---+---+---+---+---+---+---+---+
2 | o | x |   | o | x |   |   | o |   | o |    2 |   |   | o |   | x | o | o |   |   |   |
  +---+---+---+---+---+---+---+---+---+---+      +---+---+---+---+---+---+---+---+---+---+
3 |   | x |   |   | x | o | x |   | o |   |    3 |   |   |   | o | 5 |   | 4 |   | o | o |
  +---+---+---+---+---+---+---+---+---+---+      +---+---+---+---+---+---+---+---+---+---+
4 |   | x |   | o |   | o |   |   |   | o |    4 |   |   | o |   | 5 | o | x | o | o | o |
  +---+---+---+---+---+---+---+---+---+---+      +---+---+---+---+---+---+---+---+---+---+
5 | o |   |   |   | o |   |   | o |   |   |    5 | o |   | o | o | o | o | 4 | o | o |   |
  +---+---+---+---+---+---+---+---+---+---+      +---+---+---+---+---+---+---+---+---+---+
6 |   | o | o |   |   | o |   |   | o |   |    6 | o |   | o |   | o | o | 4 |   |   |   |
  +---+---+---+---+---+---+---+---+---+---+      +---+---+---+---+---+---+---+---+---+---+
7 | x | x | x | o |   |   | o |   |   | o |    7 | 2 | 2 | 2 | o | o |   |   |   |   | o |
  +---+---+---+---+---+---+---+---+---+---+      +---+---+---+---+---+---+---+---+---+---+
8 | o | o | x | x |   | o |   | o |   |   |    8 |   |   |   | o |   | o |   | o |   | 1 |   |
  +---+---+---+---+---+---+---+---+---+---+      +---+---+---+---+---+---+---+---+---+---+
9 |   |   |   | o |   |   |   | o |   |   | o |  9 |   |   |   | o |   |   |   |   |   | 1 |   |
  +---+---+---+---+---+---+---+---+---+---+      +---+---+---+---+---+---+---+---+---+---+
Enemy fired at A, 5
Enter target location: g 2
It's a hit!
Your markers and your board:
    A   B   C   D   E   F   G   H   I   J          A   B   C   D   E   F   G   H   I   J
  +---+---+---+---+---+---+---+---+---+---+      +---+---+---+---+---+---+---+---+---+---+
0 |   | x |   |   | o |   | o |   |   | o |    0 | x | 3 | 3 | o | 5 | o | o |   | o |   |
  +---+---+---+---+---+---+---+---+---+---+      +---+---+---+---+---+---+---+---+---+---+
1 |   | x | o |   | x | o |   |   | o |   |    1 | o |   |   |   | x |   |   |   |   |   |
  +---+---+---+---+---+---+---+---+---+---+      +---+---+---+---+---+---+---+---+---+---+
2 | o | x |   | o | x |   | x | o |   | o |    2 |   |   | o |   | x | o | o |   |   |   |
  +---+---+---+---+---+---+---+---+---+---+      +---+---+---+---+---+---+---+---+---+---+
3 |   | x |   |   | x | o | x |   | o |   |    3 |   |   |   | o | 5 |   | 4 |   | o | o |
  +---+---+---+---+---+---+---+---+---+---+      +---+---+---+---+---+---+---+---+---+---+
4 |   | x |   | o |   | o |   |   |   | o |    4 |   |   | o |   | 5 | o | x | o | o | o |
  +---+---+---+---+---+---+---+---+---+---+      +---+---+---+---+---+---+---+---+---+---+
5 | o |   |   |   | o |   |   | o |   |   |    5 | o |   | o | o | o | o | 4 | o | o |   |
  +---+---+---+---+---+---+---+---+---+---+      +---+---+---+---+---+---+---+---+---+---+
```


Enemy fired at E, 1
Enter target location: g 1
It's a hit!
Your markers and your board:
Enemy fired at I, 0
Enter target location: g 4
You sunk a BATTLESHIP.
You won!
NIL
[3]>