Kuncheng Feng
CSC 466

What's new?

Ship placement methods now give feedback as to what went wrong.

Quick Interactive Demo

```
[1]> (load "Main.l")
;; Loading file Main.l ...
;;  Loading file Board.l ...
;;  Loaded file Board.l
;;  Loading file Row.l ...
;;  Loaded file Row.l
;;  Loading file Cell.l ...
;;  Loaded file Cell.l
;;  Loading file Ship.l ...
;;  Loaded file Ship.l
;; Loaded file Main.l
T
[2]> (setf board (newBoard 10 10))
#<BOARD #x1A9CF995>
[3]> (display board)
     A   B   C   D   E   F   G   H   I   J
   +---+---+---+---+---+---+---+---+---+---+
 0 |   |   |   |   |   |   |   |   |   |   |
   +---+---+---+---+---+---+---+---+---+---+
 1 |   |   |   |   |   |   |   |   |   |   |
   +---+---+---+---+---+---+---+---+---+---+
 2 |   |   |   |   |   |   |   |   |   |   |
   +---+---+---+---+---+---+---+---+---+---+
 3 |   |   |   |   |   |   |   |   |   |   |
   +---+---+---+---+---+---+---+---+---+---+
 4 |   |   |   |   |   |   |   |   |   |   |
   +---+---+---+---+---+---+---+---+---+---+
 5 |   |   |   |   |   |   |   |   |   |   |
   +---+---+---+---+---+---+---+---+---+---+
 6 |   |   |   |   |   |   |   |   |   |   |
```

```
   +---+---+---+---+---+---+---+---+---+---+
 7 |   |   |   |   |   |   |   |   |   |   |
   +---+---+---+---+---+---+---+---+---+---+
 8 |   |   |   |   |   |   |   |   |   |   |
   +---+---+---+---+---+---+---+---+---+---+
 9 |   |   |   |   |   |   |   |   |   |   |
   +---+---+---+---+---+---+---+---+---+---+
NIL
[4]> (setf ship (newShip 'plane))
Ship type not supported.
NIL
[5]> (setf ship (newShip 'carrier))
#<SHIP #x1A9CA95D>
[6]> (placeShip 0 0 3 0 ship board)
Error: Incorrect size.
NIL
[7]> (placeShip 0 10 4 10 ship board)
Error: Position 1 out of bound.
NIL
[8]> (placeShip 9 9 13 9 ship board)
Error: Position 2 out of bound.
NIL
[9]> (placeShip 0 0 4 4 ship board)
Error: Ship needs to be vertical or horizontal.
NIL
[10]> (placeShip 0 0 0 4 ship board)
T
[11]> (display board)
     A   B   C   D   E   F   G   H   I   J
   +---+---+---+---+---+---+---+---+---+---+
 0 | 5 |   |   |   |   |   |   |   |   |   |
   +---+---+---+---+---+---+---+---+---+---+
 1 | 5 |   |   |   |   |   |   |   |   |   |
   +---+---+---+---+---+---+---+---+---+---+
 2 | 5 |   |   |   |   |   |   |   |   |   |
   +---+---+---+---+---+---+---+---+---+---+
 3 | 5 |   |   |   |   |   |   |   |   |   |
   +---+---+---+---+---+---+---+---+---+---+
 4 | 5 |   |   |   |   |   |   |   |   |   |
```

```
   +---+---+---+---+---+---+---+---+---+---+
 5 |   |   |   |   |   |   |   |   |   |   |
   +---+---+---+---+---+---+---+---+---+---+
 6 |   |   |   |   |   |   |   |   |   |   |
   +---+---+---+---+---+---+---+---+---+---+
 7 |   |   |   |   |   |   |   |   |   |   |
   +---+---+---+---+---+---+---+---+---+---+
 8 |   |   |   |   |   |   |   |   |   |   |
   +---+---+---+---+---+---+---+---+---+---+
 9 |   |   |   |   |   |   |   |   |   |   |
   +---+---+---+---+---+---+---+---+---+---+
NIL
[12]> (placeShip 0 3 4 3 ship board)
Error: Cells already occupied.
NIL
[13]> (display board)
     A   B   C   D   E   F   G   H   I   J
   +---+---+---+---+---+---+---+---+---+---+
 0 | 5 |   |   |   |   |   |   |   |   |   |
   +---+---+---+---+---+---+---+---+---+---+
 1 | 5 |   |   |   |   |   |   |   |   |   |
   +---+---+---+---+---+---+---+---+---+---+
 2 | 5 |   |   |   |   |   |   |   |   |   |
   +---+---+---+---+---+---+---+---+---+---+
 3 | 5 |   |   |   |   |   |   |   |   |   |
   +---+---+---+---+---+---+---+---+---+---+
 4 | 5 |   |   |   |   |   |   |   |   |   |
   +---+---+---+---+---+---+---+---+---+---+
 5 |   |   |   |   |   |   |   |   |   |   |
   +---+---+---+---+---+---+---+---+---+---+
 6 |   |   |   |   |   |   |   |   |   |   |
   +---+---+---+---+---+---+---+---+---+---+
 7 |   |   |   |   |   |   |   |   |   |   |
   +---+---+---+---+---+---+---+---+---+---+
 8 |   |   |   |   |   |   |   |   |   |   |
   +---+---+---+---+---+---+---+---+---+---+
 9 |   |   |   |   |   |   |   |   |   |   |
   +---+---+---+---+---+---+---+---+---+---+
NIL
```

```
[14]>
```

Relative Code:

```
; Checking for valid ship placement
-------------------------------------
; The ship position should be checked valid before placement
(defmethod checkValid(x1 y1 x2 y2 shipType (b board) &aux result)
     (setf result t)
     (cond
          ((not (checkType shipType))
               (format t "Error: Incorrect ship type.~%")
               (setf result nil)
          )
          ((not (checkSize shipType x1 y1 x2 y2))
               (format t "Error: Incorrect size.~%")
               (setf result nil)
          )
          ((not (checkBorder x1 y1 b))
               (format t "Error: Position 1 out of bound.~%")
               (setf result nil)
          )
          ((not (checkBorder x2 y2 b))
               (format t "Error: Position 2 out of bound.~%")
               (setf result nil)
          )
          ((not (checkDiagonal x1 y1 x2 y2))
               (format t "Error: Ship needs to be either vertical
or horizontal.~%")
               (setf result nil)
          )
          ((not (checkResidents x1 y1 x2 y2 b))
               (format t "Error: Cells already occupied.~%")
               (setf result nil)
          )
     )
     result
```

```
)

(defmethod checkType(shipType)
     (not (equal (member shipType shipTypes) nil))
)

; Note: (0 1 2 3 4) would count as size 5
(defmethod checkSize(type x1 y1 x2 y2 &aux shipSize horSize verSize)
     (setf shipSize (get 'shipSize type))
     (setf horSize (+ (abs (- x1 x2)) 1))
     (setf verSize (+ (abs (- y1 y2)) 1))
     (or
          (= shipSize horSize)
          (= shipSize verSize)
     )
)

; (X Y)
;     Y   Y
; X |0 0|0 1|
; X |1 0|1 1|
(defmethod checkBorder(x y (b board))
     (and
          (>= x 0)
          (< x (length (board-rows b)))
          (>= y 0)
          (< y (board-width b))
     )
)

; A ship cannot be placed diagonally
(defmethod checkDiagonal(x1 y1 x2 y2)
     (or
          (and
               (= x1 x2)
               (not (= y1 y2))
          )
          (and
               (not (= x1 x2))
```

```
                (= y1 y2)
            )
        )
)

; All the cells that the ship is going to occupy have to be empty
; The methods used here are defined below.
(defmethod checkResidents(x1 y1 x2 y2 (b board) &aux cells result)
        (if (= x1 x2)
            (setf cells (sameXCells x1 y1 y2 b))
            (setf cells (sameYCells x1 x2 y1 b))
        )
        (setf result t)
        (loop for cell in cells do
            (setf result (and result (checkCell cell)))
        )
        result
)

; Check if the cell is empty
(defmethod checkCell((c cell))
        (equal (cell-resident c) nil)
)
;
-------------------------------------------------------------------------
-----------------


; Placing ships
-------------------------------------------------------------------------
----
(defmethod getCell(x y (b board))
        (nth x (row-cells (nth y (board-rows b))))
)

; Note, ship is placed with this function
; Can't specify ship object because it is loaded after this
; (carrier battleship cruiser submarine destroyer)
(defmethod placeShip(x1 y1 x2 y2 (s ship) (b board) &aux cells
```

```
shipType)
     (setf shipType (ship-type s))
     (cond
          ((checkValid x1 y1 x2 y2 shipType b)
               (if (= x1 x2)
                    (setf cells (sameXCells x1 y1 y2 b))
                    (setf cells (sameYCells x1 x2 y1 b))
               )
               (setResidents cells shipType)
               (setShipCells s cells)
               t          ;; Return true to represent success
          )
          (t
               nil   ;; Return false to represent failure
          )
     )
)

; Mark the resident at the given cells
(defmethod setResidents(cells shipType &aux shipNum)
     (setf shipNum (get 'shipRep shipType))
     (dotimes (n (length cells))
          (setCellResident (nth n cells) shipNum)
     )
)

; Return all the cells between two Ys.
(defmethod sameXCells(x y1 y2 (b board))
     (cond
          ((= y1 y2)
               (list (getCell x y1 b))
          )
          ((< y1 y2) ; Up to down
               (cons (getCell x y1 b) (sameXCells x (+ y1 1) y2 b))
          )
          (t              ; Down to up
               (cons (getCell x y2 b) (sameXCells x y1 (+ y2 1) b))
          )
     )
```

```
)

; Return all the cells between two Xs.
(defmethod sameYCells(x1 x2 y (b board))
     (cond
          ((= x1 x2)
               (list (getCell x1 y b))
          )
          ((< x1 x2) ; Left to right
               (cons (getCell x1 y b) (sameYCells (+ x1 1) x2 y b))
          )
          (t              ; Right to left
               (cons (getCell x2 y b) (sameYCells x1 (+ x2 1) y b))
          )
     )
)
;
------------------------------------------------------------------------
------------
```