Kuncheng Feng
CSC 466

Presentation 3

What is new?

- The gameboard has been remodeled again.
- The ships have been modeled.
- The ships can now be placed on the board with limited restrictions.

Demo

```
[]> (demo)

An 5 x 5 board:
     A   B   C   D   E
   +---+---+---+---+---+
 0 |   |   |   |   |   |
   +---+---+---+---+---+
 1 |   |   |   |   |   |
   +---+---+---+---+---+
 2 |   |   |   |   |   |
   +---+---+---+---+---+
 3 |   |   |   |   |   |
   +---+---+---+---+---+
 4 |   |   |   |   |   |
   +---+---+---+---+---+

An 10 x 10 board:
     A   B   C   D   E   F   G   H   I   J
   +---+---+---+---+---+---+---+---+---+---+
 0 |   |   |   |   |   |   |   |   |   |   |
   +---+---+---+---+---+---+---+---+---+---+
 1 |   |   |   |   |   |   |   |   |   |   |
   +---+---+---+---+---+---+---+---+---+---+
 2 |   |   |   |   |   |   |   |   |   |   |
   +---+---+---+---+---+---+---+---+---+---+
 3 |   |   |   |   |   |   |   |   |   |   |
   +---+---+---+---+---+---+---+---+---+---+
 4 |   |   |   |   |   |   |   |   |   |   |
   +---+---+---+---+---+---+---+---+---+---+
 5 |   |   |   |   |   |   |   |   |   |   |
   +---+---+---+---+---+---+---+---+---+---+
```

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

An 15 x 15 board:

```
     A   B   C   D   E   F   G   H   I   J   K   L   M   N   O
   +---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
 0 |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
   +---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
 1 |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
   +---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
 2 |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
   +---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
 3 |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
   +---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
 4 |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
   +---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
 5 |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
   +---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
 6 |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
   +---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
 7 |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
   +---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
 8 |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
   +---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
 9 |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
   +---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
10 |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
   +---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
11 |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
   +---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
12 |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
   +---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
13 |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
   +---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
14 |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
   +---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
```

An 10 x 10 board with ships placed:

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

```
   +---+---+---+---+---+---+---+---+---+---+---+---+
 3 | 5 |   | 4 |   |   |   |   |   |   |   |   |   |
   +---+---+---+---+---+---+---+---+---+---+---+---+
 4 | 5 |   |   |   | 3 |   |   |   |   |   |   |   |
   +---+---+---+---+---+---+---+---+---+---+---+---+
 5 |   |   |   |   | 3 |   |   |   |   |   |   |   |
   +---+---+---+---+---+---+---+---+---+---+---+---+
 6 |   |   |   |   | 3 |   |   |   |   |   |   |   |
   +---+---+---+---+---+---+---+---+---+---+---+---+
 7 |   |   |   |   |   |   | 1 | 1 |   |   |   |   |
   +---+---+---+---+---+---+---+---+---+---+---+---+
 8 |   | 2 | 2 | 2 |   |   |   |   |   |   |   |   |
   +---+---+---+---+---+---+---+---+---+---+---+---+
 9 |   |   |   |   |   |   |   |   |   |   |   |   |
   +---+---+---+---+---+---+---+---+---+---+---+---+
```

An 12 x 12 board with 4 carriers:

```
     A   B   C   D   E   F   G   H   I   J   K   L
   +---+---+---+---+---+---+---+---+---+---+---+---+
 0 |   |   |   |   |   |   |   |   |   |   |   |   |
   +---+---+---+---+---+---+---+---+---+---+---+---+
 1 |   | 5 |   |   |   |   |   |   |   |   |   |   |
   +---+---+---+---+---+---+---+---+---+---+---+---+
 2 |   | 5 |   |   |   |   | 5 |   |   |   |   |   |
   +---+---+---+---+---+---+---+---+---+---+---+---+
 3 |   | 5 |   |   |   |   | 5 |   |   |   |   |   |
   +---+---+---+---+---+---+---+---+---+---+---+---+
 4 |   | 5 |   |   |   |   | 5 |   |   |   |   |   |
   +---+---+---+---+---+---+---+---+---+---+---+---+
 5 |   | 5 |   |   |   |   | 5 |   |   |   |   |   |
   +---+---+---+---+---+---+---+---+---+---+---+---+
 6 |   |   |   |   |   |   | 5 |   | 5 |   |   |   |
   +---+---+---+---+---+---+---+---+---+---+---+---+
 7 |   |   | 5 |   |   |   |   |   | 5 |   |   |   |
   +---+---+---+---+---+---+---+---+---+---+---+---+
 8 |   |   | 5 |   |   |   |   |   | 5 |   |   |   |
   +---+---+---+---+---+---+---+---+---+---+---+---+
 9 |   |   | 5 |   |   |   |   |   | 5 |   |   |   |
   +---+---+---+---+---+---+---+---+---+---+---+---+
 10|   |   | 5 |   |   |   |   |   | 5 |   |   |   |
   +---+---+---+---+---+---+---+---+---+---+---+---+
 11|   |   | 5 |   |   |   |   |   |   |   |   |   |
   +---+---+---+---+---+---+---+---+---+---+---+---+
```

```
Cell information at (1 1):
Cell row: 1
Cell number: 1
Cell resident: 5
Cell explored: NIL

Cell information at (2 1):
```

```
Cell row: 1
Cell number: 2
Cell resident: NIL
Cell explored: NIL

Ship information of carrier 1:
Ship type: CARRIER
Ship status: FLOATING
Ship cells:
(#<CELL #x1A3C56D5> #<CELL #x1A3C5839> #<CELL #x1A3C599D> #<CELL
#x1A3C5B01>
 #<CELL #x1A3C5C65>)
NIL
```

Code for this demo

```
(defun demo(&aux board carrier battleship cruiser submarine
destroyer c1 c2 c3 c4)
     (format t "~%An 5 x 5 board:~%")
     (display (newBoard 5 5))

     (format t "~%An 10 x 10 board:~%")
     (display (newBoard 10 10))

     (format t "~%An 15 x 15 board:~%")
     (display (newBoard 15 15))

     (format t "~%An 10 x 10 board with ships placed:~%")
     (setf board (newBoard 10 10))
     (setf carrier (newShip 'carrier))
     (placeShip 0 0 0 4 carrier board)
     (setf battleship (newShip 'battleship))
     (placeShip 2 0 2 3 battleship board)
     (setf cruiser (newShip 'cruiser))
     (placeShip 4 4 4 6 cruiser board)
     (setf submarine (newShip 'submarine))
     (placeShip 1 8 3 8 submarine board)
     (setf destroyer (newShip 'destroyer))
     (placeShip 6 7 7 7 destroyer board)
     (display board)

     (format t "~%An 12 x 12 board with 4 carriers: ~%")
     (setf board (newBoard 12 12))
     (setf c1 (newShip 'carrier))
     (setf c2 (newShip 'carrier))
     (setf c3 (newShip 'carrier))
     (setf c4 (newShip 'carrier))
     (placeShip 1 1 1 5 c1 board)
```

```
      (placeShip 6 2 6 6 c2 board)
      (placeShip 2 7 2 11 c3 board)
      (placeShip 8 6 8 10 c4 board)
      (display board)
      (format t "~%Cell information at (1 1): ~%")
      (getInfo (getCell 1 1 board))
      (format t "~%Cell information at (2 1): ~%")
      (getInfo (getCell 2 1 board))
      (format t "~%Ship information of carrier 1: ~%")
      (getInfo c1)
)
```

All of the code

Main.l

```
; File: Main.l

(load "Board.l")
(load "Row.l")
(load "Cell.l")
(load "Ship.l")

; Demo for 2/28/2023 --------------------
(defun demo(&aux board carrier battleship cruiser submarine destroyer c1 c2 c3 c4)
       (format t "~%An 5 x 5 board:~%")
       (display (newBoard 5 5))

       (format t "~%An 10 x 10 board:~%")
       (display (newBoard 10 10))

       (format t "~%An 15 x 15 board:~%")
       (display (newBoard 15 15))

       (format t "~%An 10 x 10 board with ships placed:~%")
       (setf board (newBoard 10 10))
       (setf carrier (newShip 'carrier))
       (placeShip 0 0 0 4 carrier board)
       (setf battleship (newShip 'battleship))
       (placeShip 2 0 2 3 battleship board)
       (setf cruiser (newShip 'cruiser))
       (placeShip 4 4 4 6 cruiser board)
       (setf submarine (newShip 'submarine))
       (placeShip 1 8 3 8 submarine board)
       (setf destroyer (newShip 'destroyer))
       (placeShip 6 7 7 7 destroyer board)
       (display board)

       (format t "~%An 12 x 12 board with 4 carriers: ~%")
       (setf board (newBoard 12 12))
       (setf c1 (newShip 'carrier))
       (setf c2 (newShip 'carrier))
       (setf c3 (newShip 'carrier))
       (setf c4 (newShip 'carrier))
       (placeShip 1 1 1 5 c1 board)
       (placeShip 6 2 6 6 c2 board)
       (placeShip 2 7 2 11 c3 board)
       (placeShip 8 6 8 10 c4 board)
       (display board)
       (format t "~%Cell information at (1 1): ~%")
       (getInfo (getCell 1 1 board))
```

```
        (format t "~%Cell information at (2 1): ~%")
        (getInfo (getCell 2 1 board))
        (format t "~%Ship information of carrier 1: ~%")
        (getInfo c1)
)
```

Board.l

```
; File: Board.l
(defclass board()
        (
                (rows :accessor board-rows :initarg :rows)
                (width :accessor board-width :initarg :width)
        )
)


; Constructor ----------------------------------------------------------
; The board can support up to 26 width, due to user experience reasons.
(defmethod newBoard(width height &aux rows)
        (setf rows (list))
        (dotimes (rowNum height)
                (setf rows (cons (newRow width rowNum) rows))
        )
        (setf rows (reverse rows))
        (make-instance 'board
                :rows rows
                :width width
        )
)
; ---------------------------------------------------------------------


; 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 ship (b board) &aux cells shipType)
        (setf shipType (ship-type ship))
        (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 ship 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))
		)
	)
)
; ---------------------------------------------------------------------------


; All for user experience ---------------------------------------------------
(defmethod display((b board) &aux width rows)
	(setf width (board-width b))
	(setf rows (board-rows b))
	(displayTop width)
	(dotimes (n (length rows))
		(display (nth n rows))
	)
	(newLine width)
)
; ---------------------------------------------------------------------------
```

Row.l

```
; File: Row.l
(defclass row()
	(
		(number :accessor row-number :initarg :number :initform 0)
		(cells :accessor row-cells :initarg :cells :initform nil)
	)
)

(defmethod newRow(width number &aux cells)
	(setf cells (list))
	(dotimes (cellNum width)
		(setf cells (cons (newCell number cellNum) cells))
	)
	(setf cells (reverse cells))
	(make-instance 'row
		:number number
		:cells cells
	)
)

; Display methods -----------------------------------------------------
; Display the top line
(defmethod displayTop(boardWidth)
```

```
        (format t "   ")
        (dotimes (n boardWidth)
                (format t "  ~A " (cellToLetter n))
        )
        (format t " ~%")
)

(setf letters '(a b c d e f g h i j k l m n o p q r s t u v w x y z))

(defmethod cellToLetter(cellNumber)
        (nth cellNumber letters)
)

(defmethod letterToCell(cellLetter)
        (position cellLetter letters)
)

; Repeating cell displays
(defmethod newLine(cellLength)
        (format t "   ")
        (dotimes (n cellLength)
                (format t "+---")
        )
        (format t "+ ~%")
)


(defmethod display((r row) &aux cells cellLength rowNumber)
        (setf cells (row-cells r))
        (setf cellLength (length cells))

        ; +---+---+---+
        (newLine cellLength)

        ; Numbers on the left
        (setf rowNumber (row-number r))
        (if (< 9 rowNumber)
                (format t " ~A" rowNumber)
                (format t " ~A " rowNumber)
        )

        ; |   |   |
        (dotimes (n cellLength)
                (display (nth n cells))
        )
        (format t "| ~%")
)
; ---------------------------------------------------------------------
```

Cell.l

```
; File: Cell.l
; Note: cell-resident should be a positive integer less than 10.
(defclass cell()
        (
                (resident :accessor cell-resident :initform nil)
                (explored :accessor cell-explored :initform nil)
                (cellRow :accessor cell-row :initarg :cellRow :initform 0)
                (cellNum :accessor cell-num :initarg :cellNum :initform 0)
        )
)

(defmethod newCell(rowNumber cellNumber)
        (make-instance 'cell
                :cellRow rowNumber
                :cellNum cellNumber
        )
)
```

```
(defmethod setCellResident((c cell) resident)
	(setf (cell-resident c) resident)
)

; This method display the player's board
(defmethod display((c cell) &aux resident number)
	(setf resident (cell-resident c))
	(setf number (cell-num c))
	(if (equal resident nil)
		(format t "|   ")
		(format t "| ~A " (cell-resident c))
	)
)

; This method is for development purposes
(defmethod getInfo((c cell))
	(format t "Cell row: ~A~%" (cell-row c))
	(format t "Cell number: ~A~%" (cell-num c))
	(format t "Cell resident: ~A~%" (cell-resident c))
	(format t "Cell explored: ~A~%" (cell-explored c))
)
```

Ship.l

```
; File Ship.l
(defclass ship()
	(
		(type :accessor ship-type :initarg :type)
		(cells :accessor ship-cells :initform nil)
		(status :accessor ship-status :initform 'floating)
	)
)

(defmethod setShipCells((s ship) cells)
	(setf (ship-cells s) cells)
)

(defmethod getInfo((s ship))
	(format t "Ship type: ~A ~%" (ship-type s))
	(format t "Ship status: ~A ~%" (ship-status s))
	(format t "Ship cells: ~A ~%" (ship-cells s))
)

(setf (symbol-plist 'shipRep) '(carrier 5 battleship 4 cruiser 3 submarine 2 destroyer 1))
(setf (symbol-plist 'shipSize) '(carrier 5 battleship 4 cruiser 3 submarine 3 destroyer 2))


; Checking for valid ship placement -------------------------------------
; The ship position should be checked valid before placement
(defmethod checkValid(x1 y1 x2 y2 shipType (b board))
	(and
		(checkSize shipType x1 x2 y1 y2)
		(checkBorder x1 y1 b)
		(checkBorder x2 y2 b)
		(checkDiagonal x1 y1 x2 y2)
	)
)

; Note: (0 1 2 3 4) would count as size 5
(defmethod checkSize(type x1 x2 y1 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)
                )
        )
)
; ---------------------------------------------------------------------


; Constructor class, the positions are assumed valid! -------------------
(defmethod newShip(type)
        (make-instance 'ship :type type)
)
; -------------------------------------------------------------------------
```