**Project PF1 – Bomberman**

**Information**

This is the team project for programming fundamentals.

Title: Bomberman

Team members: Chengyu Yu, Elena Hu

**Functionality**

This project aims to develop a 2D battle game based on the gameplay of Bomberman for MacOS keyboard. Players will place bombs in a map with randomly generated elements to destroy obstacles and try to eliminate opponents.

Map: The map is a 15x11 rectangle with 165 cells. The maps will be generated randomly. It consists of a random combination of four basic elements, one unit of one element occupies one cell: ‘U: fix position and indestructible blocks; ‘D: random position and destructible blocks; ‘W: random position aisle; ‘S: fix position for starting areas.

- ‘U: fix position and indestructible blocks: players can neither pass through these blocks nor place bombs on them. Players cannot destroy them. The position is fixed and will remain unchanged, even after random map generation.

- ‘D: random position and destructible blocks: players cannot pass through these blocks nor place bombs on them. Players can destroy them. After they are destroyed, they become ‘W.The position will change after the random map generation.

- ‘W random position aisles: players can walk through the aisle. Players can place the bombs here. The position will change after the random map generation.

- ‘S fix starting areas: there are two ‘S areas, one on the top-left corner, one on the bottom-right corner. Each of them occupies 4 cells. When the game starts, the players will appear on different ‘S. Their function and render are as same as ‘W aisles, but the position won’t change after random map generation.

Movement: player 1 moves with ‘up’, ‘down’, ‘left’ and ‘right’ keys and place the bombs with ‘return’ key; player 2 moves with ‘w’, ‘s’, ‘a’ and ‘d’ keys and place the bombs with ‘space’ key.

Non-exploded Bombs: Bomb will occupy one ‘W or ‘S cell and make the cell become ‘B. Players cannot pass through bombs or place another bomb on bombs. The bombs will explode 3 seconds after the player places it. Each player can have a maximum of ‘n’ bombs on the map, and ‘n’ will increase along with the round-timer countdown of one game.

Exploded Bombs: Exploded bombs will occupy a cross-shape area of 9 cells. Players can walk in this area. It lasts 1 second. All destructible blocks and players in this area during this period will be destroyed/defeated.

Game end conditions: 1) if a player is defeated, game over. The player defeated loses. 2) if after round time, no player is defeated, the game ends in a draw.

**Resources**

(require 2htdp/image)

(require 2htdp/universe)

We do not plan to use external libraries for any core functionality, but we are considering introducing external libraries, if available, to implement background music.

**Data structures**

;gamestate is a structure

(define-struct gamestate [layout bomb player1 player2 roundtimer maximum])

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;layout:;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

;;data types:

;layout is one of the following:

; -- (Vector of (Vector of Symbol))

; -- #false

;;;;;;;;;;;;;;;;;interpretation for (Vector of (Vector of Symbol));;;;;;;;;;;;;;;;;;;;;;;

;the layout of the game is a 2D vector grid composed of same-size-square-shape cells.

;each cell contains a symbol.

;each symbol represents a specific element in the game.

;for each symbol:

;'S represents the safe area for players join the game at the start

;'W represents the walkable cell

;'I represents the indestructible cell

;'D represents the destructible cell

;'B represents the cell with unexploded bomb

;'E represents the cell with exploding bomb

; additional notes:

; -- each game has a random layout,there will be a function(generate-layout base)

; to randomizes the symbols 'W in base to create initial-layout

; -- in fact , 'S and 'W are both walkable cells.

; but , in the (generate-layout base) function, which randomizes the symbols 'W

; 'S cells remain unchanged to provide a fixed safe area for players starting the game.

; -- similar , 'I is also unchangeble in the (generate-layout base) function

;examples:

(define base

(vector

(vector 'S 'S 'S 'W 'W 'W 'W 'W 'W 'W 'W 'W 'W 'W 'W)

(vector 'S 'W 'W 'W 'W 'W 'W 'W 'W 'W 'W 'W 'W 'W 'W)

(vector 'S 'W 'I 'W 'I 'W 'I 'W 'I 'W 'I 'W 'I 'W 'W)

(vector 'W 'W 'W 'W 'W 'W 'W 'W 'W 'W 'W 'W 'W 'W 'W)

(vector 'W 'W 'I 'W 'I 'W 'I 'W 'I 'W 'I 'W 'I 'W 'W)

(vector 'W 'W 'W 'W 'W 'W 'W 'W 'W 'W 'W 'W 'W 'W 'W)

(vector 'W 'W 'I 'W 'I 'W 'I 'W 'I 'W 'I 'W 'I 'W 'W)

(vector 'W 'W 'W 'W 'W 'W 'W 'W 'W 'W 'W 'W 'W 'W 'W)

(vector 'W 'W 'I 'W 'I 'W 'I 'W 'I 'W 'I 'W 'I 'W 'S)

(vector 'W 'W 'W 'W 'W 'W 'W 'W 'W 'W 'W 'W 'W 'W 'S)

(vector 'W 'W 'W 'W 'W 'W 'W 'W 'W 'W 'W 'W 'S 'S 'S)))

;(define initial-layout (generate-layout base))

;#false represents the the layout before game start

;for example, in start page of the game

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;layout:;;;;;;;end;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;; bomb ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

;data types:

;bomb is one of the following:

; -- '() ;; no existing boomstate in game

; -- list of bombstate structure

;interpretation

;represents all of the existing bombstate

;including both exploding and unexploed bomb

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;; bombstate ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

;data types:

;bombstate is a structure

(define-struct bombstate [cor countdown])

;interpretation:

;;;;;;;;;;;;;;;;;;;;;;;;;;;; countdown ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

;data types:

;countdown is a Integer in the interval

;from 4(included)

;to 0(included)

;interpretation:

;represents the countdown of the each bombstate

;4 is the time for bomb just added to game

;1 is the time for bomb start to explode

;0 is the time for bomb end explode

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;; cor ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

;data types:

;cor is a structure

(define-struct cor [column row])

;(make-cor Integer Integer)

;interpretation:

;represents the cell location in the game

;column is a Integer , represents the location of column

;row is a Integer, represents the location of row

;examples

;data example for cor:

(define first-row-and-column (make-cor 0 0))

;data example for bombstate:

(define added-bomb-to-00 (make-bombstate first-row-and-column 5))

;data exmample for bomb

(define initial-bomb '())

(define one-bomb (list added-bomb-to-00))

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;bomb;;;;;;;;;;;;;end;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;player1:;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

;;data types:

;player1 is a structure

(define-struct player1 [posn dead? bombcount])

;(make-player1 posn Boolean Number)

;interpretation

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;posn:;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

;Posn represents the position of player1

;it is a precise position , different from the cor (row and column location)

;additional notes:

;to achieve some functions such as put-bomb/check-dead?

;use (round posn) this kind of function

;which convert posn -> cor to get the nearest cor

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;dead?:;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

;dead? is a Boolean

;represents whether player1 has dead

;#t represents this player1 died

;#f represents this player1 alive

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;bombcount:;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

;bombcount is a Number

;represents the the amounts of the bomb that player1 put in the game

;and still exist (unexploded or exploding)

;example

(define initial-player1 (make-player1

(make-posn 0 0)

#false

0))

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;player1;;;;;;;;;;;;;end;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;player2;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

;;data types:

;player2 is a structure

(define-struct player2 [posn dead? bombcount])

;(make-player2 posn Boolean Number)

;interpretation is as similar as player1

;just replace all "player1" with "player2"

;example

(define initial-player2 (make-player1

(make-posn 200 200)

#false

0))

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;player2;;;;;;;;;;;end;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;roundtimer:;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

;data types:

;roundtimer is a Interger in the interval:

;from 120(included)

;to 0(included)

;interpretation

;represents the countdown of one game

;examples;

(define initial-roundtimer 120)

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;roundtimer;;;;;;;end;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;maximum;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

;

;data types:

;maximum is a Interger in the interval:

;from 3(included)

;to ?(undecided)

;interpretation

;represents the maximum amounts of bomb that each player can put

;the bombcount of each player should less than or equal maximum

;additional note:

;it is not a fixed constant

;it will raise along with the roundtimer countdowns in one game

;to reduce the probability of tie.

;examples

(define initial-maximum 3)

;;;;;;;;;;;example of the gamestate;;;;;;;;;;;;;;;;;;;;

;(define initial-gamestate

; initial-layout

; initial-player1

; initial-player2

; initial-roundtimer

; initial-maximum)