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 to destroy obstacles and try to eliminate opponents.

Map: The map is a x*y rectangle with x*y 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:

- 'I: 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 fix cross-shape area of 9 cells. The explosion will be obstacled by ‘I. 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)

(require racket/base)

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:;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;layout is one of the following:
; -- (Vector of (Vector of Symbol))
; -- #false

;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:
; -- there will be a function(generate-layout base)
; to randomizes the symbols 'W(it has probability to become 'D) in base to create initial-layout in each game

; -- 'S and 'W are both walkable cells.

; but , in the (generate-layout base) function

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


;(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 '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 '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 'S 'S 'S)))

;#false represents the the layout before game start

;for example, in start page of the game


;;;;;;;;;;;;;;;;;;;;;;;;;;;;;layout;;;;;;;end;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
```

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;; bomb ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;bomb is one of the following:
; -- '()  ;; no existing boomstate in game
; -- list of bombstate structure
;represents all of the existing bombstate
;including both exploding and unexploed bomb

;;;;;;;;;;;;;;;;;;;;;;;;; bombstate ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
(define-struct bombstate [cor countdown])

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;; countdown ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;countdown is a nonnegative Integer in a interval(undecided)
;represents countdown of each bombstate

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;; cor ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
(define-struct cor [column row])
;(make-cor Integer Integer)
;column is a Integer , represents the location of column
;row is a Integer, represents the location of row

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;bomb;;;;;;;;;;;;;;end;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
```

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;player1:;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
(define-struct player1 [posn dead? bombcount])
;(make-player1 posn Boolean Number)

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;posn:;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;Posn represents the position of player1
;it is a precise position , different from the cor (row and
column location)

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;dead?:;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;dead? is a Boolean
;represents whether player1 has dead

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;bombcount:;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;bombcount is a Number
;represents the the amounts of the bomb that player1 put in
the game
;and still exist (unexploded or exploding)

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;player1;;;;;;;;;;;;end;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
```

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;player2;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
(define-struct player2 [posn dead? bombcount])
Similar to player1
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;player2;;;;;;;;;;end;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;


;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;roundtimer:;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;roundtimer is a nonnegative Interger in the
interval(undecided)
;represents the countdown of one game
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;roundtimer;;;;;;;end;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;


;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;maximum;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;maximum is a nonnegative Interger in the interval(undecided)
;the bombcount of each player should less than or equal
maximum
;it will raise along with the roundtimer countdowns in one game
```