Bomberman Developer Guide

Libraries

- **racekt/base**: mainly for importing other .rkt files into the big-bang.rkt file.
- **2htdp-image**: for image manipulation and rendering, essential for visualizing the Gamestate.
- **2htdp-universe**: mainly for using big-bang function.
- **racket/vector**: allow more efficient operations on vectors, such as using vector-map.
- **racket/system**: allow access to system-level operations, such as using terminal commands in Racket to open background music.
- **racket/string**: mainly for string-contains?

Public

This part contains data definitions, generate-layout function and its helper functions, some constant definitions and some public functions such as convert, get-symbol,in-bound?.

Render

This part contains the rendering logic, how the game will be shown as an image.
- **render-cell**: each single cell is generated.
- **render-layout**: a random layout is generated, recalls function **render-cell** in it.
- **render**: the main function, which shows the whole game view. If the current layout of the game is the homepage layout, displays the homepage elements. If the layout is not the homepage, it proceeds to render the game page with a bar at the top of the game page generate by function **render-bar**, which displays the current round timer, maximum bomb count, and how many bombs each player has placed.

Keyhandler

The main function is **keyhandler**, pressing q key will quit the game, pressing space key from homepage will enter the intial random generated layout; pressing ”down” ”up” ”left” ”right” player1 will move down, up, left, right; pressing ”w” ”a” ”s” ”d”, player2 will move up, left, down, right; pressing space key in gamepage, player1 puts a bomb; pressing ”g”, player2 puts a bomb.
In the main function, function **put-bomb** and function **move** are recalled to control the movement of player and the put-bomb action, function **move-predicate?** and funciton **put-predicate?** are recalled to restrict them according to the cell types and coordinates.

Timehandler

The **timehandler** function serves as the core logic for the on-tick process, responsible for updating the game state at each tick. It first calls the **updated-layout** and **updated-bomb** functions to update the game layout and the countdown of bombs, while also handling the round timer and the maximum number of bombs by invoking **check-roundtimer?** and **add-maximum**. If any bomb's countdown reaches zero, it calls the **boom** function to handle chain explosion effects. Within **boom**, the **single-boom-range** function calculates the explosion range of a single bomb, and **extend-direction** extends the explosion range in all directions. Subsequently, the player and layout states are updated accordingly. Auxiliary functions such as **check-I?** and **check-D?** are used to determine whether a specific cell is an indestructible or destructible wall, while **chain-explosion** handles chain reactions and updates the bomb list. After processing in **boom** is complete, the **remove-bomb** function removes bombs that have already exploded. Through the close coordination of these functions, **timehandler** dynamically updates all critical game states at each tick.

Big-bang

The **big-bang** function ties together the following components:

- **on-tick:** Calls the **timehandler** function to update the gamestate dynamically at each tick.

- **on-key:** Invokes the **keyhandler** functions to process user inputs, such as player movements and bomb placements.

- **to-draw:** Uses **render** functions to display the current game layout, including players, bombs, and other elements.

- **stop-when:** Determines when the game should end by calling the **end?** function.
- **final:** Displays the appropriate endgame screen when the game ends.

Stop-when

This part describes the logic for determining when the game should stop and how the final game state is displayed. It consists of the following key functions:

end?: This function checks whether the game should stop based on the current game state. The game ends if any of the following conditions are met:

- The player has chosen to quit the game.
- The round timer reaches 0.
- Both players have died.
- Player 1 has died.
- Player 2 has died.

The function returns **#t** if any of these conditions are true and **#f** otherwise.

final: Once the game ends, this function determines the final image to display based on the game state.

- If the player has chosen to quit, it returns **quit-image**.
- If both players have died or the round timer has finished, it returns **tie**.
- If only Player 1 has died, it returns **player2-win**.
- If only Player 2 has died, it returns **player1-win**.

This ensures that the appropriate message is displayed to reflect the game's outcome.

Auxiliary functions such as **check-player1-died?**, **check-player2-died?**, and **check-all-died?** are used by **end?** to evaluate the players' statuses. Additionally, helper functions like **player1-win**, **player2-win**, and **tie** are invoked by **final** to generate specific endgame images based on the result.