Bomberman

Documentation

Scientific Python - Project work

László Levente POZSONYI (VKVCZO)
István SZÉKELY (GY2W5Z)

11 May 2022

Abstract

In this summary and documentation we briefly go through the most important properties and features of our project work.

1 The goal of our project

The goal of our project was to implement a game mimicking the famous game called Bomberman. During the development, we practiced the basics of Python and studied more advanced modules as well. Furthermore, we got more familiar with git and working in a team overall.

2 Implemented features

Here is a list of features we implemented:

- GUI for starting screen and the game itself
- Selection of number of real and AI players
- Loading maps
- Movement of players
- Placing bombs
- Explosion of bombs
- Damage of bombs

- Death of players
- Handling end of the game
- AI for controlling players

3 The structure of the project

The design of the project follows the object oriented principles.

The project consists of the following folder and files:

- `maps`: folder to store the maps
- `ai_input.py`: code to implement the AI players
- `bomb.py`: code to implement the bomb
- `game.py`: code to implement the GUI
- `game_logic.py`: code to implement the logic and rules of the game
- `initialize.py`: code to start the game if GUI is not used (for debugging and developing purposes)
- `player.py`: code to implement a player, whether it is real or AI
- `player_input.py`: code to implement the control of a real player
- `text_ui.py`: code to implement command line interface (for debugging and developing purposes)

3.1 ai_input.py

In this file the AIInput class is implemented, which is responsible for handling the AI players. In that class the following functions are implemented:

- `__handle_input`: This function controls the AI players' movement. In each iteration it collects the allowed movements for all AI players and selects one from them randomly, with weighted chances.

3.2 bomb.py

In this file the Bomb class is implemented, which is responsible for handling the bomb. In that class the following functions are implemented:

- `__timer`: This function is a timer for the detonation, running on its own thread.

3.3 game.py

In this file the Game class is implemented, which is responsible for the GUI part of the game with the help of `tkinter`. In this class the following functions are implemented:

- `__refresh`: This function updates the game board by creating the cells of the board and also at the end of the game this function updates the GUI to inform the players about the results.
- `start`: This function starts the actual game, when the play button is pressed.
- `build_menu`: This function draws the main menu of the game. Here the number of real players and the number of AI players can be set and after that the game can be started by pressing the `PLAY` button, or the program can be closed by pressing the `EXIT` button.
- `build_board`: This function draws the game board. The cells' colors are referring to the following types:
  - black: impenetrable wall
  - blue: penetrable wall
  - grey: free cell
  - white: actual explosion

  The smaller colored rectangles referring to the players and the smaller black rectangles are referring to the bombs.

3.4 game_logic.py

In this file the GameLogic class is implemented, which is responsible for handling the logic of the game and enforcing its rules. In that class the following functions are implemented:

- `__load_map`: This function loads a map.
- `__create_players`: This function creates the real and AI players.
- `__text_ui`: This function handles the command line interface, running on its own thread.
- `handle_input`: This function handles the inputs coming from real and AI players.
- `__move_player`: This function moves the players on the game board.
- `__place_bomb`: This function places a bomb on the game board.
- `detonate`: This function detonates a bomb in a given position.
- `__detonate_check`: This function is a helper of `detonate`, helping handling the detonation.
- `__remove_explosion`: This function removes the beams of an exploded bomb.

3.5 initialize.py

This file is only a short script, starting the game if CLI is used.

3.6 player.py

In this file the Player class is implemented, which is responsible for the representation of the players. Each player has a limited amount of lives and an alive property. In that class the following functions are implemented:

- `hp_loss`: This function manages the player's lives. In case of damage it decreases the number of lives and if the life counter hits zero the player is dead and out of the game.

3.7 player_input.py

In this file the PlayerInput class is implemented, which is responsible for handling the real players. The game can handle 2 real players at a time. In that class the following functions are implemented:

- `__handle_input`: In this function in each iteration the program reads the input from the keyboard and according to that read data moves the appropriate player on the board.

3.8 text_ui.py

In this file the TextUI class is implemented, which is responsible for writing to the CLI. In that class the following functions are implemented:

- `update`: This function updates the game board to be drawn.
- `draw`: This function writes the game board to the CLI.

4 Conclusion

In our opinion, the project was successful, we learned plenty of new things, which is the main objective of the project work. We were able to implement most of the features, though sometimes small hiccups and bugs can occur here and there.