Breakython

Hugo Luís Andrade Silva (ha2385)
Larissa Navarro Passos de Araujo (ln2307)
Vinícius de Moraes Rego Cousseau (vd2299)

1. Description

Breakout is an arcade game developed by Atari in 1972. The purpose of the game is to clear the brick lines, using a ball that travels across the screen. When a brick is hit, the ball bounces away and the brick is destroyed. The player loses when the ball drops to the bottom of the screen. In this project, we intend to implement a Python version of Breakout, with a modularized approach, allowing for easy addition of levels or new game modes. Our Breakout version will have power-up items, which will be chosen randomly and have effects such as: shrinking the paddle, making the paddle larger, making the ball keep hitting bricks instead of bouncing away and speeding the ball up.

2. System Behavior

- As soon as the player executes the program, a menu is going to pop-up
- The menu is going to present the following options:
    1. Play game: starts the game
    2. Quit game: exits the program
- The menu will be controlled by the UP and DOWN arrow keys and the ENTER button.
- If the player chooses to start the game, the window will become a screen with the paddle, ball and bricks.
- As soon as the game starts, the player will be able to control the paddle through the LEFT and RIGHT arrow keys, pause the game using the ESC key and make the ball start moving using the ENTER key.
- If the ESC key is pressed, a menu is going to pop up with the following options:
    1. Resume game: continues the game from the point where the ESC key was pressed
    2. Start new game: resets the game
    3. Quit game: same as in the previous menu
- This menu will be controlled by the UP and DOWN keys. ENTER will be used to select the options. The user will also be able to press ESC to exit the menu.
- The ball will stay on the paddle until the ENTER key is pressed.
- Once the ball starts moving, it will follow a diagonal path that can go either upwards or downwards. It will start going upwards as soon as the player presses ENTER
- The ball will change its trajectory whenever it hits a brick or wall, the brick will disappear if it is hit and its resistance is 1.
- If the ball hits the lower part of a brick or hits the ceiling, it starts going downwards
- If the ball hits one of the side walls or hits a brick from the side, it changes the horizontal component of its movement
- Whenever the ball gets close to hitting the floor, the user will have to position the paddle underneath it so that the ball starts going upwards again
- The player will control the paddle and the objective is to make all bricks disappear
- The game will go on until either the player wins or the ball hits the floor. The player wins in the first case and loses in the second one.
- The game will also have power ups. A power up is going to appear at a random location and stay there for a fixed amount of time, if the player hits it, he gets the power up, if he/she doesn't, the power up disappears.
- In case the power up is hit, the ball won't bounce back or change its trajectory in any way
- The power ups include:
    1. Making the paddle wider
    2. Making the paddle smaller
    3. Making the ball go faster
    4. Making the ball go slower

5. Turning the ball into a "super-ball" that will not bounce back when a block is hit. Instead, it will keep hitting other blocks.

3. Architecture

The game will be divided in three main modules: game_elements, game and gui. The game_elements module will contain the classes used to represent the main game elements. The game module will contain functions which, when called, will modify the state of the current game_elements as it was specified by the user's input in the GUI. The gui module will contain the graphical interface and, through user input, will use the corresponding game function.

- **game_elements**:
  The classes contained in this module will be:
  - Ball: class that describes the Breakout ball. This class will possess attributes such as *position* (x, y) relative to the screen, *velocity* and whether it is a *power_ball* (a powered-up ball that, instead of bouncing after it hits a brick, continues in its initial trajectory) or not.
  - Paddle: class that describes the Breakout paddle. This class will possess attributes such as *x_position* and *size*, which can be modified by power ups.
  - Brick: class that represents a Breakout brick. This class will contain attributes such as *position* (x,y) relative to the screen, *resistance*, which indicates how many hits a brick has to suffer in order to break and *color*, that contains the brick color.
  - PowerUp: class that represents a Power Up brick. This special brick appears in a random position for a limited time, and the player has to hit it once to free the Power Up. There will be 5 subclasses:
    - EnlargePaddle;
    - ShrinkPaddle;
    - PowerBall;
    - SpeedUp;
    - SlowDown.
  - Level: class that represents a level in a Breakout game. It will instantiate a number of Bricks, a Paddle and a Ball, and randomly create PowerUp objects. The level will have a number attribute that will determine the difficulty (number of bricks, ball velocity, resistance, etc.), and will keep tracks of which bricks are still up.

- **game**:
  The game module will contain functions responsible for updating the Level class, such as move_paddle_left, move_paddle_right, menu functions and functions that verify if the player lost a life or if the level is finished. It will also keep track of player lives.

- **gui**:
  The gui module will contain the user interface classes, the game loop, and will control the user interaction.

4. Library modules and 3rd party libraries

- TKInter: standard GUI package;
- pickle: for possible data persistence;
- math: mathematical functions;
- collections: container datatypes;
- pdb: debugger module

5. Work-flow

The work will be divided in the following way: Hugo will be responsible for the gui design, Larissa and Vinícius will implement the game and game_elements modules, according to the final complexity of these modules. The code will be tested using the pdb module, and each member will analyze other members' code during the implementation. For version control, the group chose to use git/GitHub. Finally, the documentation will be created automatically using pydoc.