Raphael Kalfon

raphael.kalfon@post.runi.ac.il

Concept / Idea of the App

The “Brick Breaker” game is a classic arcade-style game where the player controls a paddle (trampoline) at the bottom of the screen, bouncing a ball to break all the bricks placed at the top. The goal is to clear all the bricks without letting the ball fall off the screen. The player uses the left and right arrow keys to move the trampoline and bounce the ball to destroy the bricks.

Architecture

System Architecture Overview:

The game architecture is designed around several core components: the **Game**, **Ball**, **Brick**, **Trampoline**, and the **Main** components.

1. **Main Class**:
   - The entry point for the game. It initializes and starts the game.
   - It displays the welcome message and instructions to the player.
   - It handles the game flow, including level progression, game over, and restarting.
2. **Game Class**:
   - This class coordinates the game logic. It initializes the trampoline, ball, and bricks and manages the main game loop.
   - The game loop processes user input (left and right arrow keys to move the trampoline), updates the positions of the ball and paddle, checks for collisions, and redraws the game screen.
3. **Ball Class**:
   - Responsible for handling the ball’s movement, collisions with the trampoline and bricks.
   - It detects if it hits the edge of the screen or a brick and changes direction accordingly.
4. **Brick Class**:
   - Represents the bricks to be destroyed. Each brick has a position, sprite, and a flag indicating whether it is destroyed.
   - The draw() method is used to render the brick, and destroy() marks the brick as destroyed when hit by the ball.
5. **Trampoline Class**:
   - Controls the paddle (trampoline) that the player moves left or right.
   - It detects keypresses (left and right arrows) and moves the trampoline accordingly.
   - It is also responsible for detecting collisions with the ball, which changes the ball’s direction when they collide.

Main Logic Breakdown:

1. **Main**:
   - Initializes the game by calling game.init() and game.run().

- Provides an interface for the user to start a new game, view the score, and proceed to the next level.

2. **Game**:

- Manages the game state: initializes the trampoline and ball, creates the bricks, and manages the level progression.
- The run() method contains the main game loop, handling user inputs, ball movements, and checking collisions.

3. **Ball**:

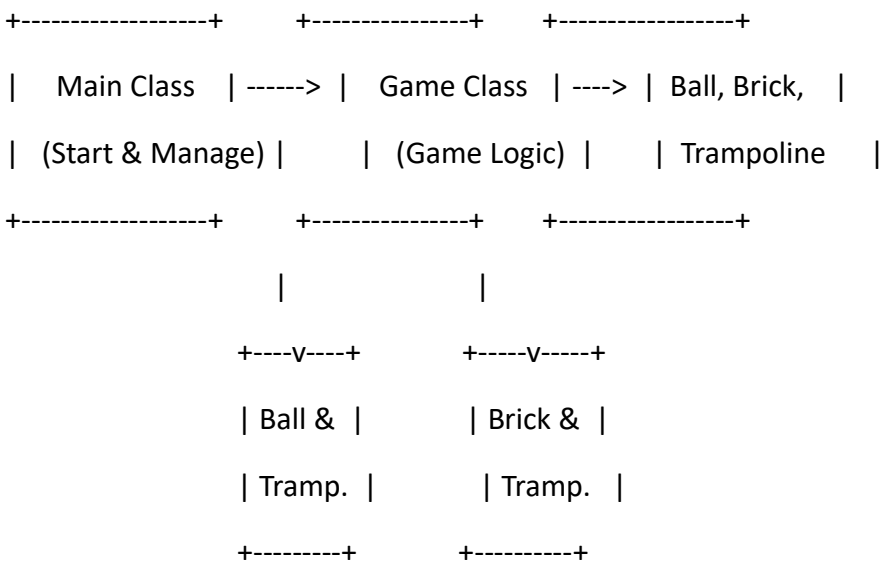
- The ball starts at the position of the trampoline and moves with a fixed velocity.
- The ball bounces off the screen borders and bricks, and the direction changes based on collision detection with bricks and the trampoline.

4. **Brick**:

- The bricks are arranged at the top of the screen in rows. When the ball collides with a brick, that brick is destroyed, and the ball's direction changes.

5. **Trampoline**:

- The trampoline moves horizontally on the screen in response to the player's keyboard input. It acts as the barrier that keeps the ball in play. If the ball hits the trampoline, it bounces upward.



Motivation

I chose to create the "Brick Breaker" game because it's a fun and simple way to practice game development concepts, such as collision detection, user input handling, and game loops. It provides an engaging challenge for the player while allowing the programmer to focus on core game mechanics like object movement, interaction, and game state management. This project also allowed me to practice efficient use of object-oriented programming to structure the game and its components effectively.