Project Report

Maryam Mohammadi

Kharazmi University

Fall - 1403

Algorithm

Class: Player

The Player class represents the player character in the game.

Attributes:

- **image:** The image representing the player character.
- **rect:** A rectangle object used to position the player on the screen.
- **speed:** The speed at which the player can move.
- **lives:** The number of lives the player has in the game.
- **velocity:** The current velocity of the player's movement.
- **acceleration:** The acceleration value used to increase movement speed.
- **deceleration:** The deceleration value used to slow the player down when no input is given.

Methods:

- **__init__(self):** Constructor to initialize the player's attributes.
- **update(self):** Updates the player's position based on key presses, handling movement and acceleration.
- **shoot(self):** Creates and adds a bullet to the game when the player shoots.

Class: Enemy

The Enemy class represents an enemy character in the game.

Attributes:

- **image: The image representing the enemy.**
- **rect:** A rectangle object used to position the enemy on the screen.
- **base_speed:** The base speed at which the enemy moves.
- **velocity:** The current velocity of the enemy's movement, which increases over time.

Methods:

- **__init__(self):** Constructor to initialize the enemy's attributes, including random position and speed.
- **update(self**): Moves the enemy down the screen and resets its position when it goes off-screen, also increasing its speed over time.

Class: Bullet

The **Bullet** class represents a bullet fired by the player.

Attributes:

- image: The image representing the bullet.
- rect: A rectangle object used to position the bullet on the screen.
- speed: The speed at which the bullet moves.

Methods:

- __init__(self, x, y): Constructor to initialize the bullet's position based on the player's position when the shot is fired.
- update(self): Moves the bullet upward and removes it from the screen when it exits.

Class: Army

The **Army** class represents an item that increases the player's lives when collected.

Attributes:

- image: The image representing the army item.
- rect: A rectangle object used to position the army item on the screen.
- speed: The speed at which the army item falls.

Methods:

- __init__(self): Constructor to initialize the army item's position and speed.
- update(self): Moves the army item downward and adds a life to the player when it reaches the bottom of the screen.

Class: Game

The **Game** class manages the overall flow and mechanics of the game.

Attributes:

- screen: The game window where everything is drawn.
- all_sprites: A group containing all game objects for easy updating and drawing.
- player: The player object.
- enemies: A group containing all enemy objects.
- bullets: A group containing all bullet objects.
- armies: A group containing all army items.
- coins: The player's score in coins.
- enemy_spawn_time: The time interval between spawning new enemies.

Methods:

- __init__(self): Initializes all attributes, creates the player, and sets up game groups and resources.
- run(self): The main game loop that handles events, updates all game objects, checks for collisions, and draws everything on the screen.
- draw_lives(self): Displays the number of lives on the screen.
- draw_coins(self): Displays the number of coins on the screen.
- check_collisions(self): Checks for collisions between bullets and enemies, as well as between the player and enemies or army items.

Services Provided:

- **Enemy Management:** The Enemy class handles the creation, movement, and behavior of enemies, including random spawning and speed adjustment.
- **Player Controls:** The Player class provides core game mechanics for player movement, shooting, and interaction with the environment (like collecting items and avoiding enemies).
- **Bullet Handling:** The Bullet class manages the creation, movement, and collision of bullets fired by the player, including removal of bullets once they exit the screen.

- **Item Management:** The Army class handles the spawning of items that increase the player's lives. Items spawn randomly and are collected by the player when they reach the bottom of the screen.
- **Collision Detection:** The game tracks collisions between bullets and enemies, bullets and the environment, and player collisions with enemies and collectible items, affecting gameplay mechanics like health and score.
- **Score and Health Display:** The score (coins) and health (lives) are displayed on the screen, updating in real-time as the player progresses through the game.
- **Game Over and Restart Logic:** The game includes a condition for game over when the player's lives reach zero, along with the option to restart or exit the game.

Potential Future Additions:

- **Adding more stages:** by increasing the difficulty and adding new obstacles.
- **Upgrading powers:** for example, the player can buy stronger weapons.
- **Multiplayer mode:** to play with other players online.
- **Better graphics:** adding new animations and improving visual effects.

Examples:

- ## Dynamic Difficulty Curve:

Gradually increase the difficulty of enemies by making them faster, more accurate, and more intelligent. As players progress through stages, enemies could evolve with new patterns or behaviors, such as evading bullets or attacking in groups.

Introduce new enemy types with varying behaviors, such as enemies that teleport, shoot in patterns, or split into smaller enemies when defeated.

Create new environments or themes for each stage (e.g., desert, space, or underwater) with unique obstacles or interactive elements.

- ## Upgrading Powers:

❖ Weapon Upgrades:

Introduce a system where the player can purchase or unlock new weapons (e.g., laser guns, homing missiles, or flamethrowers) using in-game currency or collectibles.

Implement upgradeable weapons that can be enhanced during gameplay (e.g., increasing damage, fire rate, or adding elemental effects like fire or ice).

Create a power-up system where temporary boosts (e.g., invincibility, rapid fire, shield) are available throughout the levels.

- ## Character Abilities:

Allow the player to upgrade their movement speed, health, and shield capacity.

Introduce special abilities such as slow-motion (bullet time), temporary invisibility, or a teleportation skill that helps the player avoid enemy attacks.