Programming in C++

Submitted By:

Hiranaya Yadav (CS&DF)

Submitted To:

Aasish Acharya Sir

The British College, Kathmandu

Kathmandu, Nepal

Date: 21 April, 2025

Documentation of “Block Bash” Game

Documentation:

Block Bash

Introduction:

Block Bash is a 2D brick breaker-style arcade game developed using **C++** and the **SFML (Simple and Fast Multimedia Library)**. Players control a paddle to bounce a ball and break colored blocks arranged in rows. The game includes features such as dynamic level progression, scoring system, sound effects, and high score tracking.

This project is built using object-oriented programming principles to encourage modular and maintainable game development. Key classes such as Ball, Paddle, and Block abstract core gameplay mechanics while providing opportunities to extend functionality.

Tools & Technologies Used:

- **Programming Language:** C++
- **Graphics Library:** SFML
- **IDE/Compiler:** Code::Blocks / Visual Studio / Clion
- **Assets:** ball.png, paddle.png, background.jpg, multiple block textures
- **Audio:** hit.wav, gameover.wav, music.ogg
- **Font:** arial.ttf
- **File Handling:** highscore.txt (to store persistent high scores)

Game Logic Overview:

1. Game Entities:

- **Paddle:** Controlled using left/right arrow keys. It reflects the ball upward.
- **Ball:** Moves autonomously. Bounces off walls, paddle, and blocks.
- **Block:** Breaks upon collision with the ball and increases score.

2. Ball Mechanics:

- Initially launched diagonally.
- Bounces on paddle and walls.
- After every level-up (per 100 points), the ball's speed increases and direction is randomized slightly to avoid repetitive upward movement.

3. Level Progression:

- Score increases by 10 per block destroyed.
- Every 100 points, the game progresses to the next level.
- New levels generate additional rows of blocks.
- Ball velocity increases per level with randomized angle.

4. Scoring:

- Block destroyed: +10 points.
- High score is saved between game sessions.

5. Game Over:

- Triggered when the ball touches the bottom of the screen.

- Game shows "Game Over" screen with final score.
- Player can restart by pressing **R**.

Graphics and Rendering:

- **Background:** Set using a static image.
- **Paddle & Ball:** Rendered via SFML Sprite.
- **Blocks:** Rendered via RectangleShape with texture.
- **Text Elements:** Display score, level, high score, and game over screen using SFML Text.

Sound and Music:

- **Background Music:** Looped music using SFML Music class.
- **Hit Sound:** Plays when the ball hits a block.
- **Game Over Sound:** Plays when the game ends.

File Handling:

- High score stored in highscore.txt file.
- Reads high score at game start, writes new high score on game over.

Controls:

- **Left/Right Arrow Keys:** Move paddle left/right.

- **R Key:** Restart game on Game Over screen.

Key Features Summary:

- Dynamic level-up and ball speed increase.
- Real-time collision detection.
- Randomized ball trajectory on level-up to avoid vertical movement.
- Score tracking and persistent high score saving.
- Smooth paddle controls and responsive physics.
- Background music and sound effects.
- Clean, object-oriented code structure.

Uses:

- OOP design (Ball, Paddle, Block classes)
- 2D game physics and collision handling
- File I/O operations
- Real-time input handling and rendering
- Modular and scalable architecture

Conclusion:

Block Bash effectively demonstrates how to use **C++ and SFML** to create an interactive, visually appealing 2D arcade game. The implementation of gameplay logic, sound effects, real-time rendering, and file handling offers valuable hands-on experience with game development fundamentals. Its modular, object-oriented structure makes it an ideal base for further enhancements such as power-ups, different paddle types, multi-ball gameplay, and more. This project is a strong foundation for students aspiring to delve deeper into game development and interactive programming.