Brick Breaker Game in Assembly Language

Mohammad Faizan, Minha Rehman
BS Computer Science
FAST NUCES Islamabad
Islamabad, Pakistan

***Abstract*—This paper presents the implementation of a classic arcade-style Brick Breaker game using x86 Assembly Language under the MASM/TASM environment. The game features real-time keyboard input handling, custom VGA graphics, file operations, multilevel difficulty progression, and sound feedback. It serves as a demonstration of low-level programming capabilities and real-time game development without the use of high-level libraries or frameworks. Our results highlight the effective use of BIOS and DOS interrupts for smooth gameplay and interactive user interface design.**

***Index Terms*—x86 Assembly, MASM, VGA Graphics, Game Development, File Handling, Real-Time Input**

I. Introduction

Assembly language, known for its low-level control over hardware, is rarely used in modern application development due to its complexity. However, it remains valuable for understanding computer architecture and optimizing performance-critical software. This project revisits the classic ”Brick Breaker” arcade game and reimagines it using x86 Assembly language. The game incorporates user interaction, real-time physics, graphical rendering using BIOS interrupts, and persistent score storage using DOS interrupts.

II. Game Architecture

A. Graphics and Display

The game uses VGA mode (INT 10h, mode 10h) to display colored pixels directly to the screen. Bricks, paddle, ball, and interface elements are drawn using pixel plotting routines. Game screens include:

- Introduction Screen
- Name Input Screen
- Menu Screen (Start Game, Instructions, Records)
- Gameplay Screen
- Game Over / You Win Screens

B. Controls and Input

Keyboard inputs are captured in real-time using BIOS (INT 16h) and DOS (INT 21h) interrupts. Paddle movement is controlled via arrow keys, and gameplay can be paused using the Enter key or exited using the Escape key.

C. Game Logic

The gameplay consists of:

- Paddle movement and ball reflection
- Brick collision and scoring
- Life tracking and countdown timer
- Brick durability (1-3 hits required depending on level)
- Color and speed changes across levels
- Winning condition: clear all bricks before timer ends or lives run out

III. Levels and Difficulty

A. Level 1

- Standard paddle size and ball speed
- Bricks break in one hit

B. Level 2

- Faster ball speed and shorter paddle
- Bricks require two hits and change color on the first hit

C. Level 3

- Highest ball speed and smallest paddle
- Bricks need three hits; some are unbreakable
- Includes a special brick that removes five random bricks when hit

IV. File Handling

Player names and scores are saved to ‘player.txt‘ using DOS file interrupts. Each session appends the new score to the file, which can be viewed through the in-game Records menu. This demonstrates low-level file I/O in Assembly, including:

- File open and creation
- Writing string buffers
- File close operations

V. Sound and Feedback

Sound effects are generated using the PC speaker via port manipulation (‘OUT 42h‘, ‘OUT 61h‘). Each brick hit triggers a beep, enhancing gameplay interactivity.

VI. Conclusion

This project successfully demonstrates how a complete graphical game can be developed using low-level x86 Assembly. The integration of graphics, real-time input, file handling, and audio feedback reflects a strong understanding of system-level programming. Future enhancements could include mouse input, dynamic resolution support, and advanced power-ups.

References