

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | | | | | | | | |
|  | | PING PONG GAME | | | | |  | |
|  |  | | | | | | |  |
|  | | | |  |  | | | |
|  | | | | Bilal Mohsin (F23-0646) |  | | | |
|  | | | | 08-12-2024—COAL—Dr Usama |  | | | |
|  | | |  | | |  | | |



|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | | |  |  | | |  |
|  | INSTRUCTIONS | | | | | | |  |
|  |  | | |  |  | | |  |
|  |  | |  | | |  | |  |
|  |  |  |  **Objective**: Use the paddles to hit the ball back and forth. Missing the ball increases the opponent's score. The first to reach the set score wins.   **Player Controls**:   * **Names**: First enter the names of both players one by one and then you will the menu of game. * **Player 1**: Use the keys 'W' (move up) and 'S' (move down) to control the left paddle. * **Player 2**: Use the arrow keys '↑' (move up) and '↓' (move down) to control the right paddle. * **Winning**: Whenever any players reaches max score he/she wins. We can easily change game limit (max score)    **Ball Movement**: The ball moves continuously and reflects off the walls and paddles. Missing the ball resets its position.   **Pause/Exit**:   * Press 'P' again to pause the game. * Press 'ESC' to exit the game at any time. | | |  |  |  |

#### Game Architecture

The **Ping Pong Game** is implemented using **NASM assembly** for text-mode VGA graphics. The game operates in **Mode 03h (80x25 text mode)**, utilizing **BIOS interrupts** for keyboard input and direct memory manipulation for rendering graphics. Paddle and ball logic rely on **basic arithmetic** (for movement) and **branching instructions** (for collision detection). The game also includes beautiful menus, borders, and smooth paddle-ball interactions for engaging gameplay.

#### Core Features

1. **Paddle Control**
   * **Keyboard Input**: The game uses INT 0x16 to detect keyboard input for paddle movement.
     + **Player 1**: Uses the 'W' and 'S' keys for moving the left paddle up and down.
     + **Player 2**: Uses the arrow keys (↑, ↓) for moving the right paddle up and down.
   * The paddle's position is updated in memory, and the paddle is redrawn on the screen using video memory manipulation.
2. **Ball Movement**
   * The ball's trajectory is calculated based on its **current direction** (up-right, up-left, down-right, down-left).
   * **Collision Detection**:
     + Walls: The ball bounces off the top and bottom screen borders by reversing its vertical direction.
     + Paddles: The ball reflects horizontally when hitting a paddle, with detection based on the ball’s row and column position compared to the paddle’s.
   * If the ball passes a paddle, the opponent scores, and the ball resets to the center of the screen.
3. **Score Counting**
   * Each player’s score is tracked using a **word variable** in memory.
   * Scores are displayed on the game board in real-time by directly writing to VGA memory.
   * When the ball passes a paddle, the opponent’s score is incremented, and the ball resets to the initial position.
4. **Game-Winning Conditions**
   * The game checks the scores after each ball reset.
   * If a player’s score reaches the winning score (e.g., 5), the game ends, displaying a **“Winner”** message with the player’s name.
5. **Background Implementation**
   * The game does not include a scrolling background but features **beautiful static borders** and **menus**:
     + The borders are drawn using characters like '='.
     + The menu provides an interactive way for players to enter their names and start the game.

### **Challenges and Solutions**

#### Challenges

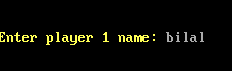
* The most difficult issue was the **ball bypassing the paddle** due to errors in collision detection logic, which relied on complex row-column comparisons.
* The reflection of ball IMO was the most difficult logic to implement. Even though I created that perfect logic very easily but making it to work with other functions was a headache.
* Debugging the program was particularly challenging as tools like **AFD** cannot effectively debug VGA text mode or direct memory manipulation.

#### Solutions

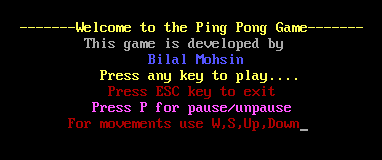
* After 1.5 days of troubleshooting, the issue was identified and I laughed when I found out that the issue was my debugging function that I created to check something else. I didn’t remove it and it caused my mental health to be taken away. I am so dumb
* Somehow CHAT GPT was able to help me in putting the ball movement function in right place.
* To overcome the lack of debugging tools, **test messages** were printed at key problem areas to trace and verify the program's behavior step by step.

#### Some screen shots of my game

**1)enter name** (it is for both players)

****

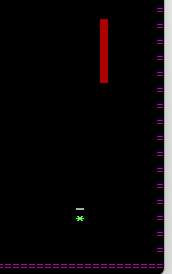
**2)Menu** (showing name and instructions)

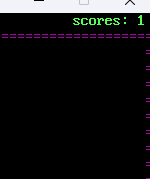


**3)pausing the ball**

****

**4)Missing the paddle** (score increase)

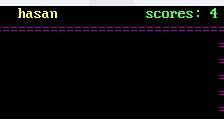
about to miss the paddle

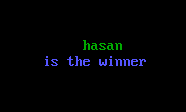
score increases

**5)Paddle moved**

****

**6) winning condition** (5 scores 🡪this limit can be changed)

Hassan about to win

Winning message