OLED Pong Game using ESP32

Aim:

The aim of this project is to implement a simple and interactive Pong game on an OLED screen using the ESP32 Trainer Kit and push buttons for gameplay interaction.

Description:

This task recreates the classic Pong game on a 128x64 OLED screen using the ESP32. It uses two push buttons to control the player's paddle while the ESP32 autonomously controls the paddle.

Features:

- Real-time Pong game displayed on OLED using ESP32.
- Two push buttons to control the player's paddle (UP and DOWN).
- Paddles move autonomously and intelligently tracks the ball.
- Collision detection and ball bounce effects against walls and paddles.
- Smooth frame updates using Adafruit GFX and SSD1306 libraries.

Procedure:

1. **Code Setup:**

   - Include required libraries:

     SPI.h, Wire.h, Adafruit_GFX.h, and Adafruit_SSD1306.h

   - Initialize display resolution: 128x64.
   - Define paddle dimensions and movement rates.
   - Use timers for paddle and ball updates.

2. **Game Initialization:**

   - Set up display with boundary rectangles.
   - Start with the ball in the center of the screen.
   - Initialize paddle positions.
   - Display a loading screen for 2 seconds.

3. **Gameplay Loop:**

   - Continuously update ball position based on direction and collisions.

- Move CPU paddle to track ball's vertical position.
- Allow player to move their paddle using push buttons.
- Refresh the display with updated graphics for smooth animation.

Working:

- When the ESP32 starts, the OLED initializes and a game border is drawn.
- The game loop continuously checks for push button presses to move the player’s paddle.
- The ball moves diagonally, bouncing off screen edges and paddles.
- The CPU paddle (on the left) follows the ball's Y-axis to attempt blocking it.
- The player paddle (on the right) is controlled using the UP (GPIO 33) and DOWN (GPIO 32) buttons.
- Collision detection ensures the ball bounces back realistically.
- The display refreshes every few milliseconds for smooth visual updates.

Code:

```
#include <SPI.h>

#include <Wire.h>

#include <Adafruit_GFX.h>

#include <Adafruit_SSD1306.h>


#define UP_BUTTON 33

#define DOWN_BUTTON 32


const unsigned long PADDLE_RATE = 33;

const unsigned long BALL_RATE = 16;

const uint8_t PADDLE_HEIGHT = 24;


#define SCREEN_WIDTH 128 // OLED display width, in pixels
```

```
#define SCREEN_HEIGHT 64 // OLED display height, in pixels


// Declaration for an SSD1306 display connected to I2C (SDA, SCL pins)
#define OLED_RESET     4 // Reset pin # (or -1 if sharing Arduino reset pin)
Adafruit_SSD1306 display (SCREEN_WIDTH, SCREEN_HEIGHT, &Wire, OLED_RESET);


void drawCourt();


uint8_t ball_x = 64, ball_y = 32;
uint8_t ball_dir_x = 1, ball_dir_y = 1;
unsigned long ball_update;


unsigned long paddle_update;
const uint8_t CPU_X = 12;
uint8_t cpu_y = 16;


const uint8_t PLAYER_X = 115;
uint8_t player_y = 16;


void setup()
 {
   display.begin(SSD1306_SWITCHCAPVCC, 0x3C);
   display.display();
   unsigned long start = millis();


   pinMode(UP_BUTTON, INPUT);
```

```
    pinMode(DOWN_BUTTON, INPUT);
    digitalWrite(UP_BUTTON,1);
    digitalWrite(DOWN_BUTTON,1);
    display.clearDisplay();
    drawCourt();

    while(millis() - start < 2000);

    display.display();

    ball_update = millis();
    paddle_update = ball_update;
}

void loop()
 {
    bool update = false;
    unsigned long time = millis();

    static bool up_state = false;
    static bool down_state = false;

    up_state |= (digitalRead(UP_BUTTON) == LOW);
    down_state |= (digitalRead(DOWN_BUTTON) == LOW);

    if(time > ball_update) {
```

```
uint8_t new_x = ball_x + ball_dir_x;
uint8_t new_y = ball_y + ball_dir_y;

// Check if we hit the vertical walls
if(new_x == 0 || new_x == 127) {
    ball_dir_x = -ball_dir_x;
    new_x += ball_dir_x + ball_dir_x;
}

// Check if we hit the horizontal walls.
if(new_y == 0 || new_y == 63) {
    ball_dir_y = -ball_dir_y;
    new_y += ball_dir_y + ball_dir_y;
}

// Check if we hit the CPU paddle
if(new_x == CPU_X && new_y >= cpu_y && new_y <= cpu_y + PADDLE_HEIGHT) {
    ball_dir_x = -ball_dir_x;
    new_x += ball_dir_x + ball_dir_x;
}

// Check if we hit the player paddle
if(new_x == PLAYER_X
   && new_y >= player_y
   && new_y <= player_y + PADDLE_HEIGHT)
{
```

```
        ball_dir_x = -ball_dir_x;
        new_x += ball_dir_x + ball_dir_x;
    }

    display.drawPixel(ball_x, ball_y, BLACK);
    display.drawPixel(new_x, new_y, WHITE);
    ball_x = new_x;
    ball_y = new_y;

    ball_update += BALL_RATE;

    update = true;
}

if(time > paddle_update) {
    paddle_update += PADDLE_RATE;

    // CPU paddle
    display.drawFastVLine(CPU_X, cpu_y, PADDLE_HEIGHT, BLACK);
    const uint8_t half_paddle = PADDLE_HEIGHT >> 1;
    if(cpu_y + half_paddle > ball_y) {
        cpu_y -= 1;
    }
    if(cpu_y + half_paddle < ball_y) {
        cpu_y += 1;
    }
```

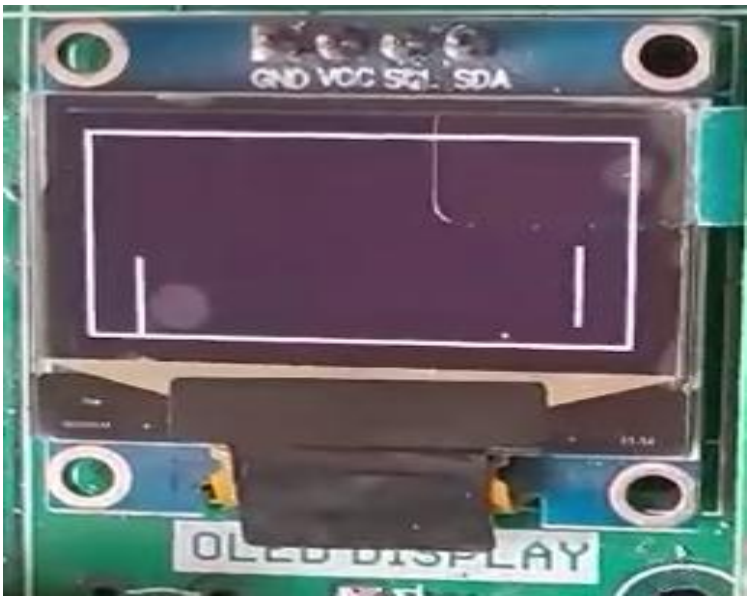
```
    if(cpu_y < 1) cpu_y = 1;
    if(cpu_y + PADDLE_HEIGHT > 63) cpu_y = 63 - PADDLE_HEIGHT;
    display.drawFastVLine(CPU_X, cpu_y, PADDLE_HEIGHT, WHITE);

    // Player paddle
    display.drawFastVLine(PLAYER_X, player_y, PADDLE_HEIGHT, BLACK);
    if(up_state) {
      player_y -= 1;
    }
    if(down_state) {
      player_y += 1;
    }
    up_state = down_state = false;
    if(player_y < 1) player_y = 1;
    if(player_y + PADDLE_HEIGHT > 63) player_y = 63 - PADDLE_HEIGHT;
    display.drawFastVLine(PLAYER_X, player_y, PADDLE_HEIGHT, WHITE);

    update = true;
  }

  if(update)
    display.display();
}

void drawCourt() {
  display.drawRect(0, 0, 128, 64, WHITE);
```

}

Output:

OLED Display:

- A rectangle boundary enclosing the Pong game.
- A ball bouncing between the two paddles.
- The left paddle (CPU) automatically moves up or down to follow the ball.
- The right paddle (Player) responds to button presses.



Gameplay Visual:

- Ball and paddle movements update in real-time.
- Smooth game rendering on a 128x64 OLED screen.
- Interactive and enjoyable gaming experience with just an ESP32 and an OLED.