#include <SPI.h>

#include <Wire.h>

#include <Adafruit\_GFX.h>

#include <Adafruit\_SSD1306.h>

#define SCREEN\_WIDTH 128

#define SCREEN\_HEIGHT 64

#define OLED\_RESET -1

Adafruit\_SSD1306 display(SCREEN\_WIDTH, SCREEN\_HEIGHT, &**Wire**, OLED\_RESET);

const int paddleWidth = 8;

const int paddleHeight = 20;

const int paddleSpeed = 1;

const int ballSize = 3;

const int ballSpeed = 1;

int paddle1Y, paddle2Y;

int ballX, ballY;

int ballDX, ballDY;

int player1Score, player2Score;

#define BUTTON1\_PIN 4 // Change pin numbers according to your setup

#define BUTTON2\_PIN 5

void setup() {

**Serial**.begin(9600);

  if(!display.begin(SSD1306\_SWITCHCAPVCC, 0x3C)) {

**Serial**.println(F("SSD1306 allocation failed"));

    for(;;);

  }

  display.clearDisplay();

  display.display();

  randomSeed(analogRead(0));

  paddle1Y = paddle2Y = (SCREEN\_HEIGHT - paddleHeight) / 2;

  ballX = SCREEN\_WIDTH / 2;

  ballY = SCREEN\_HEIGHT / 2;

  ballDX = random(2) == 0 ? -ballSpeed : ballSpeed;

  ballDY = random(2) == 0 ? -ballSpeed : ballSpeed;

  pinMode(BUTTON1\_PIN, INPUT\_PULLUP);

  pinMode(BUTTON2\_PIN, INPUT\_PULLUP);

}

void loop() {

  handleInput();

  moveBall();

  movePaddles();

  updateDisplay();

  delay(10);

}

void handleInput() {

  // Read the state of button 1

  int button1State = digitalRead(BUTTON1\_PIN);

  // If button 1 is pressed and the paddle 1 position is within the screen bounds, move the paddle up

  if (button1State == LOW && paddle1Y > 0) {

    paddle1Y -= paddleSpeed;

  }

  // Read the state of button 2

  int button2State = digitalRead(BUTTON2\_PIN);

  // If button 2 is pressed and the paddle 1 position is within the screen bounds, move the paddle down

  if (button2State == LOW && paddle1Y < SCREEN\_HEIGHT - paddleHeight) {

    paddle1Y += paddleSpeed;

  }

}

void movePaddles() {

  // Artificial Intelligence for player 2 (computer)

  int targetY = ballY - paddleHeight / 2;

  if (targetY > paddle2Y) {

    paddle2Y += paddleSpeed;

  } else if (targetY < paddle2Y) {

    paddle2Y -= paddleSpeed;

  }

  // Keep paddles within bounds

  paddle1Y = constrain(paddle1Y, 0, SCREEN\_HEIGHT - paddleHeight);

  paddle2Y = constrain(paddle2Y, 0, SCREEN\_HEIGHT - paddleHeight);

}

void moveBall() {

  // Move the ball

  ballX += ballDX;

  ballY += ballDY;

  // Check for collisions with walls

  if (ballY <= 0 || ballY >= SCREEN\_HEIGHT - ballSize) {

    ballDY = -ballDY;

  }

  // Check for collisions with paddles

  if (ballX <= paddleWidth && ballY + ballSize >= paddle1Y && ballY <= paddle1Y + paddleHeight) {

    ballDX = -ballDX;

  }

  if (ballX >= SCREEN\_WIDTH - paddleWidth - ballSize && ballY + ballSize >= paddle2Y && ballY <= paddle2Y + paddleHeight) {

    ballDX = -ballDX;

  }

  // Check for scoring

  if (ballX < 0) {

    player2Score++;

    resetBall();

  }

  if (ballX > SCREEN\_WIDTH) {

    player1Score++;

    resetBall();

  }

}

void resetBall() {

  ballX = SCREEN\_WIDTH / 2;

  ballY = SCREEN\_HEIGHT / 2;

  ballDX = random(2) == 0 ? -ballSpeed : ballSpeed;

  ballDY = random(2) == 0 ? -ballSpeed : ballSpeed;

}

void updateDisplay() {

  display.clearDisplay();

  // Draw paddles

  display.fillRect(0, paddle1Y, paddleWidth, paddleHeight, WHITE);

  display.fillRect(SCREEN\_WIDTH - paddleWidth, paddle2Y, paddleWidth, paddleHeight, WHITE);

  // Draw ball

  display.fillCircle(ballX, ballY, ballSize, WHITE);

  // Display scores

  display.setTextSize(1);

  display.setTextColor(WHITE);

  display.setCursor(SCREEN\_WIDTH / 4, 0);

  display.print(player1Score);

  display.setCursor(3 \* SCREEN\_WIDTH / 4, 0);

  display.print(player2Score);

  display.display();

}