#include <graphics.h>

#include <conio.h>

#include <dos.h>

#include <stdlib.h> // For rand()

#include <stdio.h> // For sprintf()

// Paddle settings

int paddleX = 300;

int paddleY = 450;

int paddleWidth = 80;

int paddleHeight = 10;

int paddleSpeed = 20;

// Ball settings

int ballX = 320;

int ballY = 240;

int ballRadius = 10;

int ballSpeedX = 3;

int ballSpeedY = -3;

// Brick settings

int brickWidth = 60;

int brickHeight = 20;

int rows = 5;

int cols = 10;

int bricks[5][10]; // 5 rows, 10 columns of bricks

// Game state

int gameOver = 0;

int score = 0;

void drawPaddle() {

setcolor(WHITE);

rectangle(paddleX, paddleY, paddleX + paddleWidth, paddleY + paddleHeight);

floodfill(paddleX + paddleWidth / 2, paddleY + paddleHeight / 2, WHITE);

}

void drawBall() {

setcolor(RED);

circle(ballX, ballY, ballRadius);

floodfill(ballX, ballY, RED);

}

void drawBricks() {

for (int i = 0; i < rows; i++) {

for (int j = 0; j < cols; j++) {

if (bricks[i][j] == 1) {

int brickX = j \* brickWidth + 100;

int brickY = i \* brickHeight + 50;

setcolor(GREEN);

rectangle(brickX, brickY, brickX + brickWidth, brickY + brickHeight);

floodfill(brickX + brickWidth / 2, brickY + brickHeight / 2, GREEN);

}

}

}

}

void resetBricks() {

for (int i = 0; i < rows; i++) {

for (int j = 0; j < cols; j++) {

bricks[i][j] = 1; // All bricks are visible initially

}

}

}

void drawScore() {

char scoreStr[10];

sprintf(scoreStr, "Score: %d", score);

setcolor(WHITE);

outtextxy(10, 10, scoreStr);

}

void moveBall() {

ballX += ballSpeedX;

ballY += ballSpeedY;

// Ball collision with walls

if (ballX - ballRadius <= 0 || ballX + ballRadius >= getmaxx()) {

ballSpeedX = -ballSpeedX; // Reverse horizontal direction

}

if (ballY - ballRadius <= 0) {

ballSpeedY = -ballSpeedY; // Reverse vertical direction (top wall)

}

// Ball collision with paddle

if (ballY + ballRadius >= paddleY && ballX >= paddleX && ballX <= paddleX + paddleWidth) {

ballSpeedY = -ballSpeedY; // Bounce off the paddle

}

// Ball falls below the paddle

if (ballY + ballRadius > getmaxy()) {

gameOver = 1; // End the game if the ball falls below the screen

}

// Ball collision with bricks

for (int i = 0; i < rows; i++) {

for (int j = 0; j < cols; j++) {

if (bricks[i][j] == 1) {

int brickX = j \* brickWidth + 100;

int brickY = i \* brickHeight + 50;

// Check if the ball hits a brick

if (ballX >= brickX && ballX <= brickX + brickWidth &&

ballY - ballRadius <= brickY + brickHeight && ballY + ballRadius >= brickY) {

bricks[i][j] = 0; // Destroy the brick

ballSpeedY = -ballSpeedY; // Reverse ball direction

score += 10; // Increase score

}

}

}

}

}

void movePaddle() {

if (kbhit()) {

char key = getch();

if (key == 75 && paddleX > 0) { // Left arrow key

paddleX -= paddleSpeed;

}

if (key == 77 && paddleX + paddleWidth < getmaxx()) { // Right arrow key

paddleX += paddleSpeed;

}

}

}

void displayGameOver() {

setcolor(WHITE);

outtextxy(250, 200, "GAME OVER!");

outtextxy(250,220,"VERY GOOD!");

outtextxy(220, 240, "Press 'r' to restart");

}

int main() {

int gd = DETECT, gm;

initgraph(&gd, &gm, "C:\\TURBOC3\\BGI");

resetBricks(); // Initialize bricks

while (!gameOver) {

cleardevice(); // Clear the screen

// Move ball and paddle

moveBall();

movePaddle();

// Draw paddle, ball, bricks, and score

drawPaddle();

drawBall();

drawBricks();

drawScore();

delay(20); // Control game speed

}

// Handle Game Over state

while (gameOver) {

displayGameOver();

if (kbhit()) {

char key = getch();

if (key == 'r' || key == 'R') {

// Reset the game

gameOver = 0;

score = 0;

paddleX = 300;

ballX = 320;

ballY = 240;

ballSpeedX = 3;

ballSpeedY = -3;

resetBricks(); // Reset bricks

}

}

delay(100); // Slow down the game over screen

}

closegraph();

return 0;

}