Report On

BOUNCING BALL.

Submitted in partial fulfillment of the requirements of the Course project in Semester III of Second Year Artificial Intelligence and Data Science

by
Mohammed Ali Jaffari(16)
Gautam Chaudhari(04)
Ayush Gupta(14)

Supervisor
Mrs. Sejal D’Mello

University of Mumbai

Vidyavardhini's College of Engineering & Technology

Department of Artificial Intelligence and Data Science

(2023-24)

Vidyavardhini's College of Engineering & Technology
Department of Artificial Intelligence and Data Science

CERTIFICATE

This is to certify that the project entitled “BOUNCING BALL” is a bonafide work of "Gautam Chaudhari(04), Ayush Gupta(14), Mohammed Ali Jaffari(16)" submitted to the University of Mumbai in partial fulfillment of the requirement for the Course project in semester III of Second Year Artificial Intelligence and Data Science engineering.

Supervisor

Mrs. Sejal D’Mello

Dr. Tatwadarshi P. N.
Head of Department

Table of Contents

OVERVIEW

Title: Bouncing Ball (C)

Overview:

This C program utilizes the Turbo C graphics library to create a visually engaging animation of a bouncing ball. It commences by initializing the graphics environment, a crucial step in ensuring proper graphical output. The program configures various attributes of the bouncing ball, including its initial position (x and y coordinates), radius, and movement speeds (deltaX and deltaY) for horizontal and vertical motion. These parameters dictate the behavior of the animated ball.The central element of the program is a continuous loop that orchestrates the animation. Within this loop, the screen is consistently refreshed, delivering a seamless visual experience. The ball's position is dynamically updated by incrementing its x and y coordinates, driven by the deltaX and deltaY values. This mechanism imparts the illusion of a bouncing ball.

In conclusion, this program serves as an instructive example of graphics programming, demonstrating the fundamental concepts of graphical initialization, animation, collision detection, and resource cleanup.

PROGRAM:

2.1 Code:

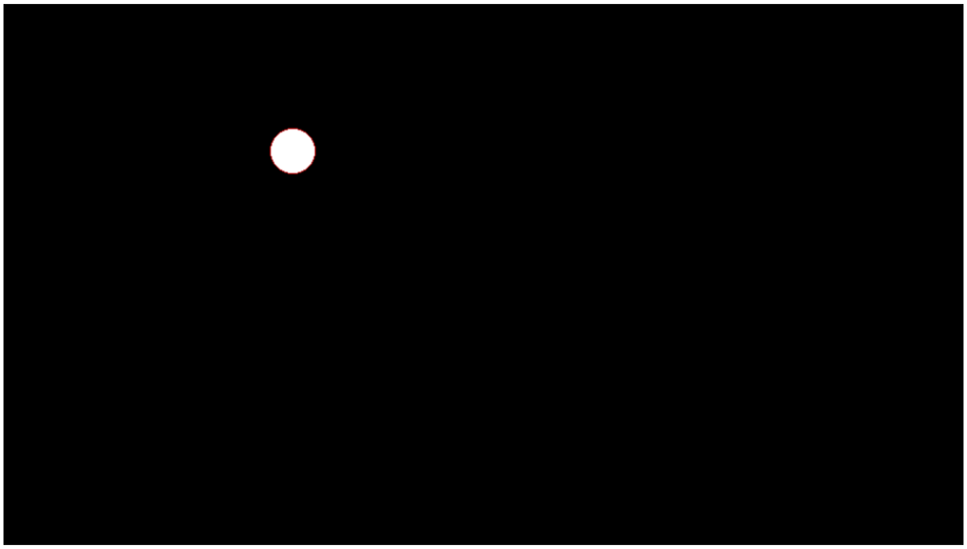
```
#include <graphics.h>
#include <conio.h>
int main() {
    int x,y,radius,deltaX,deltaY;
    int gd = DETECT, gm;
    initgraph(&gd, &gm, "C:\\Turboc3\\BGI");
     x = 320;     // Initial x-coordinate of the ball
     y = 240;      // Initial y-coordinate of the ball
     radius = 20;   // Radius of the ball
     deltaX = 5;   // Horizontal movement speed
     deltaY = 5;    // Vertical movement speed
       while (!kbhit()) {
        setbkcolor(BLACK);
        cleardevice();

        setcolor(RED);
        circle(x, y, radius);
        floodfill(x, y, RED);
       delay(100);        // Delay for smooth animation

        x += deltaX;
        y += deltaY;

// Check for collision with screen boundaries
        if (x - radius <= 0 || x + radius >= getmaxx()) {
            deltaX = -deltaX; // Reverse horizontal direction
        }
        if (y - radius <= 0 || y + radius >= getmaxy()) {
            deltaY = -deltaY; // Reverse vertical direction
        }
    }
    closegraph();
    return 0;
}
```

2.2 Output:



TECHNICALITIES:

3.1Technologies Used:

The C program you provided uses the Turbo C graphics library for creating a graphical animation. Here are the key technologies and components involved:

- C Programming Language: The program is written in the C programming language, a widely used language for system and application development.

- Turbo C: Turbo C is an integrated development environment (IDE) that was popular in the 1980s and early 1990s. It includes a C compiler and graphics library for DOS-based systems.

- BGI (Borland Graphics Interface): The program utilizes the BGI graphics library, specifically designed for Turbo C. BGI provides functions for graphics and animation, allowing developers to create visual applications.

- Graphics.h Header: The graphics.h header file is included to access graphics-related functions, such as initgraph, setbkcolor, cleardevice, circle, and floodfill. These functions are part of the BGI graphics library and enable graphical output.

- Conio.h Header: The conio.h header file is included for keyboard input handling. It provides functions like kbhit to detect keyboard input and is often used in conjunction with the Turbo C environment.

- DOS-Based Environment: This program is designed for DOS-based systems, as Turbo C was primarily used in such environments. It relies on DOS for system interaction and screen handling.

- Screen Buffer: The program utilizes a screen buffer to create a graphical window and display the animation. It updates the buffer to achieve animation effects.

- Keyboard Input: The program detects keyboard input using functions from conio.h, allowing the user to exit the animation loop by pressing a key

3.2 Explanation:

1. **Header Files Inclusion:** The program includes two header files, <graphics.h> and <conio.h>. <graphics.h> provides functions for graphics programming, while <conio.h> provides functions for console input.
2. **Main Function:** The main function serves as the entry point of the program.
3. **Variable Declarations:** The program declares several integer variables to control the animation:

- x and y store the current position of the ball.
- radius defines the radius of the ball.
- deltaX and deltaY represent the horizontal and vertical movement speeds.

4. **Graphics Initialization:** The program initializes the graphics environment using the initgraph function. It uses the DETECT parameter to automatically detect the graphics mode and the gm variable to store the graphics mode.
5. **Animation Loop:** The core of the program is a while loop, which continues until a key is pressed (detected by kbhit()). Inside the loop:

- The screen background is set to black using setbkcolor(BLACK).
- The cleardevice() function clears the screen to prepare it for the next frame.

6. **Drawing the Ball:** The ball is drawn as a red circle using the setcolor(RED), circle(x, y, radius), and floodfill(x, y, RED) functions. These functions create and fill the ball shape.
7. **Delay for Smooth Animation:** A delay(100) is introduced to create a delay of 100 milliseconds between frames, providing a smooth animation effect.
8. **Updating Ball Position:** The x and y coordinates of the ball are updated based on the deltaX and deltaY values. This motion simulates the ball's movement.
9. **Collision Detection:** The program checks for collisions with the screen boundaries. If the ball reaches the left or right edges (x - radius <= 0 or x + radius >= getmaxx()), the horizontal direction (deltaX) is reversed. Likewise, if the ball reaches the top or bottom edges (y - radius <= 0 or y + radius >= getmaxy()), the vertical direction (deltaY) is reversed.

10. **Cleanup and Exit:** When a key is pressed (i.e., a key hit is detected by kbhit()), the program exits the animation loop. It proceeds to close the graphics environment using closegraph() and returns 0 to indicate successful program execution.