Report On

Solar System

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

by
Aaryan Gole (Roll No. 12)
Aryan Gaikwad (Roll No. 09)
Nitish Jha (Roll No. 18)

Supervisor
Prof. Sneha Yadav

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 “Solar System” is a bonafide work of " Aaryan Gole (Roll No. 12), Aryan Gaikwad (Roll No. 09), Nitish Jha (Roll No. 18) " 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

Prof. Sneha Yadav

Dr. Tatwadarshi P. N.
Head of Department

Table of Contents

1. OVERVIEW

The "Solar System Simulation" program in C leverages graphics.h library to create an animated representation of the solar system. This application provides an engaging visual depiction of planetary orbits, offering an educational tool for students and enthusiasts interested in astronomy.

Key Features:

Graphics Initialization: The program initializes the graphics system, setting up the screen for visual rendering.

Planetary Motion Calculation: Utilizing mathematical calculations, the positions of the eight planets are determined within their respective orbits. These calculations account for radii, angles, and orbits.

Dynamic Animation: The program continuously updates and redraws the solar system, creating the illusion of planetary motion. Each iteration simulates the planets' movement along their orbits.

Rotation Tracking: The program monitors each planet's position and resets it when a full rotation is completed. This ensures that the simulation accurately represents celestial motion.

Animation Control: A controlled delay is introduced between frames, allowing users to adjust the speed of the simulation.

Graceful Termination: The program gracefully exits upon user input, closing the graphics window and releasing any allocated resources.

This Solar System Simulation program serves as an informative and visually engaging educational tool, providing users with an interactive platform to explore and understand the dynamics of our solar system. Its intuitive interface and dynamic animation make it an accessible resource for learners of all levels.

2. PROGRAM

2.1 ALGORITHM

1. Initialize graphics mode and set up the screen.

2. Calculate the positions of 8 planets on their orbits.

3. Continuously update and redraw the solar system for animation.

4. Check if a planet completes a full rotation and reset its position.

5. Add a delay for animation.

6. Close the graphics window and deallocate resources when a key is pressed.

7. Exit the program.

2.2 CODE

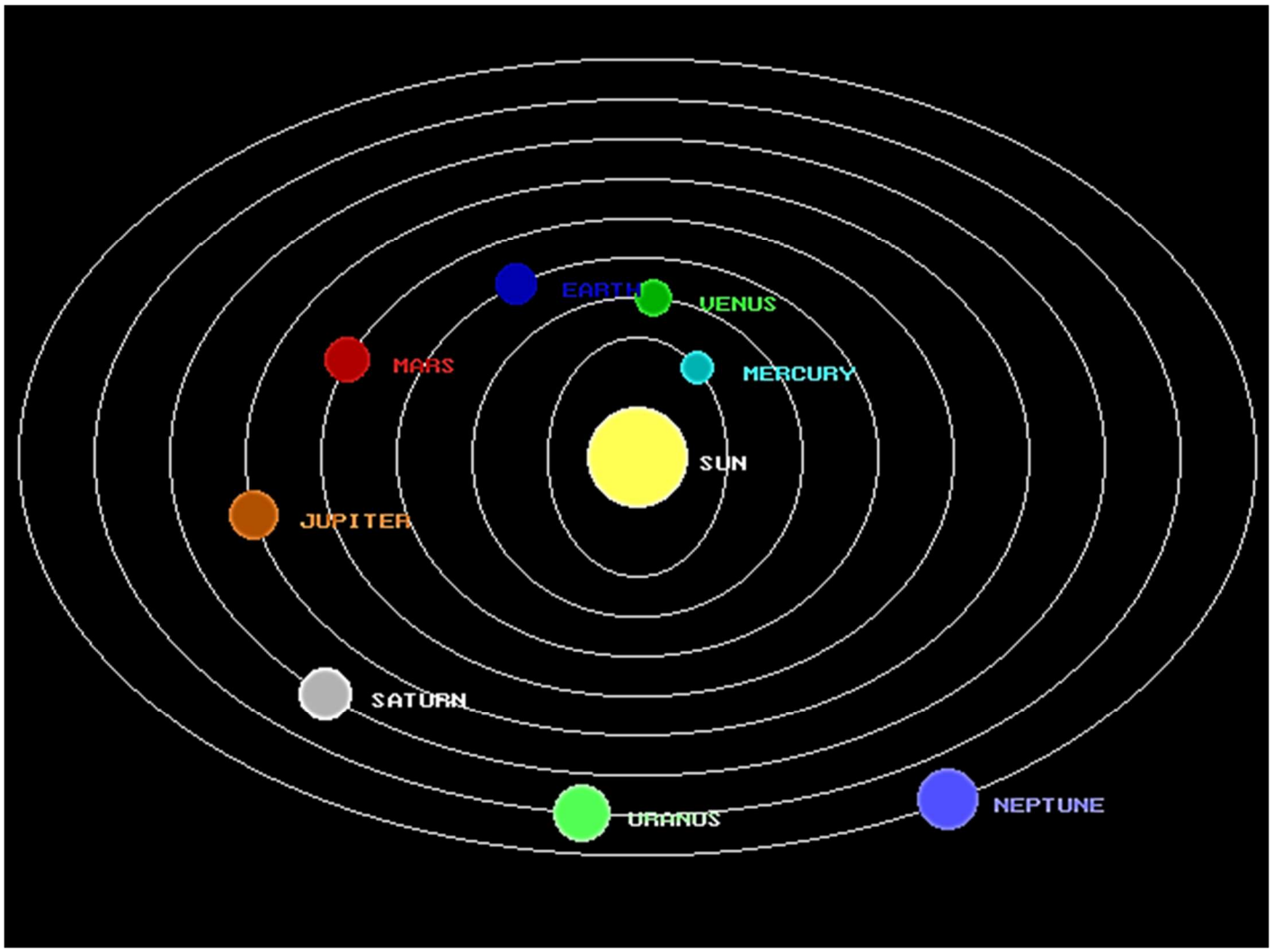
```
#include <conio.h>
#include <dos.h>
#include <graphics.h>
#include <math.h>
#include <stdio.h>

// Function to manipulates the position of planets on the orbit
void planetMotion(int xrad, int yrad,
                  int midx, int midy,
                  int x[70], int y[70])
{
    int i, j = 0;

    // Positions of planets in their corresponding orbits
    for (i = 360; i > 0; i = i - 6) {
        x[j] = midx - (xrad * cos((i * 3.14) / 180));
        y[j++] = midy - (yrad * sin((i * 3.14) / 180));
    }

    return;
}

// Driver Code
int main()
{

    // Initialize graphic driver
    int gdriver = DETECT, gmode, err;
    int i = 0, midx, midy;
    int xrad[8], yrad[8], x[8][70], y[8][70];
    int pos[8], planet[8], tmp;

    initgraph(&gdriver, &gmode, "C:\\Turboc3\\BGI");
    err = graphresult();

    if (err != grOk) {

        // Error occurred
        printf("Graphics Error: %s",
            grapherrormsg(err));
        return 0;
    }
```

```
// Mid positions at x and y-axis
midx = getmaxx() - 320;
midy = getmaxy() - 250;

// Manipulating radii of all the eight planets
planet[0] = 8;
for (i = 1; i < 8; i++) {
    planet[i] = planet[i - 1] + 1;
}

// Offset position for the planets on their corresponding orbit
for (i = 0; i < 8; i++) {
    pos[i] = i * 6;
}

// Orbits for all 8 planets
xrad[0] = 45, yrad[0] = 60;
for (i = 1; i < 8; i++) {
    xrad[i] = xrad[i - 1] + 38;
    yrad[i] = yrad[i - 1] + 20;
}
// Positions of planets on their corresponding orbits
for (i = 0; i < 8; i++) {
    planetMotion(xrad[i], yrad[i],
                midx, midy, x[i],
                y[i]);
}
while (!kbhit()) {
    // Drawing 8 orbits
    setcolor(WHITE);
    for (i = 0; i < 8; i++) {
        setcolor(WHITE);
        ellipse(midx, midy, 0, 360,
                xrad[i], yrad[i]);
    }
    // Sun at the mid of solar system
    outtextxy(midx, midy, "    SUN");
    setcolor(YELLOW);
    setfillstyle(SOLID_FILL, YELLOW);
    circle(midx, midy, 25);
    floodfill(midx, midy, YELLOW);
    // Mercury in first orbit
    setcolor(CYAN);
    setfillstyle(SOLID_FILL, CYAN);
    outtextxy(x[0][pos[0]],
            y[0][pos[0]],
```

```
        "   MERCURY");

pieslice(x[0][pos[0]],
        y[0][pos[0]],
        0, 360, planet[0]);

// Venus in second orbit
setcolor(GREEN);
setfillstyle(SOLID_FILL, GREEN);
outtextxy(x[1][pos[1]],
        y[1][pos[1]],
        "   VENUS");
pieslice(x[1][pos[1]],
        y[1][pos[1]],
        0, 360, planet[1]);

// Earth in third orbit
setcolor(BLUE);
setfillstyle(SOLID_FILL, BLUE);
outtextxy(x[2][pos[2]],
        y[2][pos[2]],
        "   EARTH");
pieslice(x[2][pos[2]],
        y[2][pos[2]],
        0, 360, planet[2]);

// Mars in fourth orbit
setcolor(RED);
setfillstyle(SOLID_FILL, RED);
outtextxy(x[3][pos[3]],
        y[3][pos[3]],
        "   MARS");
pieslice(x[3][pos[3]],
        y[3][pos[3]],
        0, 360, planet[3]);

// Jupiter in fifth orbit
setcolor(BROWN);
setfillstyle(SOLID_FILL, BROWN);
outtextxy(x[4][pos[4]],
        y[4][pos[4]],
        "   JUPITER");
pieslice(x[4][pos[4]],
        y[4][pos[4]],
        0, 360, planet[4]);
// Saturn in sixth orbit
setcolor(LIGHTGRAY);
```

```
        setfillstyle(SOLID_FILL, LIGHTGRAY);
        outtextxy(x[5][pos[5]],
                y[5][pos[5]],
                "   SATURN");
        pieslice(x[5][pos[5]],
                y[5][pos[5]],
                0, 360, planet[5]);
        // Uranus in seventh orbit
        setcolor(LIGHTGREEN);
        setfillstyle(SOLID_FILL, LIGHTGREEN);
                outtextxy (x [6] [pos [6]],
                        y [6] [pos [6]],
                        "   URANUS");
                pieslice (x [6] [pos [6]],
                        y [6] [pos [6]],
                        0, 360, planet [6]);
        // Neptune in eighth orbit
        setcolor (LIGHTBLUE);
        setfillstyle (SOLID_FILL, LIGHTBLUE);
        outtextxy (x [7] [pos [7]],
                y [7] [pos [7]],
                "   NEPTUNE");
        pieslice (x [7] [pos [7]],
                y [7] [pos [7]],
                0, 360, planet [7]);

        // Checking for one complete
        // rotation
        for (i = 0; i < 8; i++) {
            if (pos[i] <= 0) {
                pos[i] = 59;
            }
            else {
                pos[i] = pos[i] - 1;
            }
        }
        // Sleep for 100 milliseconds
        delay (100);

        // Clears graphic screen
        cleardevice ();
    }
    // Deallocate memory allocated for graphic screen
    // or closing the graphics window
    closegraph();
    return 0;
}
```

2.3 OUTPUT


EARTH
VENUS
MARS
MERCURY
SUN
JUPITER
SATURN
URANUS
NEPTUNE

3. TECHNICALITIES

3.1 TECHNOLOGIES UTILIZED

- C Programming Language:

  C is a widely-used, general-purpose programming language known for its efficiency, low-level system access, and flexibility. It is commonly used for system-level programming and applications where performance is critical.

- graphics.h Library:

  graphics.h is a C library that provides functions for graphics programming. It is used for creating graphical applications, including drawing shapes, handling colors, and managing screen displays. In this program, graphics.h is used to render the graphical elements of the solar system simulation.

- conio.h Library:

  conio.h is a header file in the C standard library that provides functions for console input/output operations. In this program, it is used for functions like kbhit() to detect keyboard input, allowing the program to exit when a key is pressed.

- dos.h Library:

  dos.h is a header file in the C standard library that provides access to various DOS (Disk Operating System) functions. In this program, it is used to handle low-level operations, such as setting the graphics mode and accessing system time.

- math.h Library:

  math.h is a standard C library that provides mathematical functions and operations. In this program, it is used for mathematical calculations involving trigonometric functions (cos and sin), as well as constants like PI.

- stdio.h Library:

  stdio.h is a standard C library that provides functions for standard input and output operations. In this program, it is used for functions like printf() to display messages and information to the console.

- initgraph(), graphresult(), getmaxx(), getmaxy():

  These functions are part of the graphics.h library. initgraph() initializes the graphics system, graphresult() checks for any errors during initialization, and getmaxx() and getmaxy() return the maximum x and y coordinates of the screen, respectively.

- setcolor(), setfillstyle(), outtextxy(), circle(), ellipse(), pieslice(), floodfill(), cleardevice():

  These are functions provided by the graphics.h library for drawing shapes, setting colors, filling shapes, and handling text. They are used to render the orbits, planets, and labels in the simulation.

Overall, the program combines the power of the C programming language with graphics.h and other standard libraries to create a visual representation of the solar system. It uses graphics.h to draw and animate the orbits and planets, while other libraries like conio.h and dos.h provide additional functionalities for input handling and system-level operations.

3.2 EXPLANATION

Step 1: We first include necessary libraries

```
#include <conio.h>
#include <dos.h>
#include <graphics.h>
#include <math.h>
#include <stdio.h>
```

These are header files that provide various functions for handling console input/output (conio.h), low-level operations (dos.h), graphics programming (graphics.h), mathematical operations (math.h), and standard input/output (stdio.h).

Step 2: We define Main Function, and initialize Graphics and Graphics Mode and then handle graphics initialization problems.

Step 3: Calculate Mid Points and initialize Planet Radii and positions:

```
midx = getmaxx() - 320;
midy = getmaxy() - 250;
int xrad[8], yrad[8], x[8][70], y[8][70];
int pos[8], planet[8], tmp;
```

Step 4: Calculate Planet Radii and Positions and then make an Animation loop and then draw orbits, sun & planets.

```
int xrad[8], yrad[8], x[8][70], y[8][70];
int pos[8], planet[8], tmp;
while (!kbhit()) {
    // Code for drawing orbits, planets, and animation
}
```