SMT. Indira Gandhi College of Engineering

Plot No 1, Sector S, Ghansoli, Navi-Mumbai-400 701

A MINIPROJECT

OF

COMPUTER GRAPHICS

Simulation of SOLAR SYSTEM

Submitted By:

Om Chaudhari D66

Bhavesh Choudhari D67

Guided By:

Asst. Prof. Nirosha Uppu Ma'am

INSTITUTE VISION AND MISSION

Vision

- **To serve and have a transformative impact on society by constantly pursuing excellence in technical education, innovation and entrepreneurship for human development with strong ethical values.**

Mission

- **Serve and help transform society by graduating talented, broadly educated engineers, equipped with state of art technology resources for developing sustainable solutions.**

- **Academic excellence in Science, Engineering and Technology through dedication to duty, commitment to research, innovation in learning and faith in human values.**

- **Cultivate the spirit of entrepreneurship and the connection between academia and industry that fosters problem solving through collaboration**

- **Enable the students to develop into outstanding professionals with high ethical standards capable of creating, developing and managing global engineering enterprises.**

SMT. Indira Gandhi College of Engineering

Plot No 1, Sector S, Ghansoli, Navi-Mumbai-400 701

CERTIFICATE

This is to certify that Mr/Miss _____ of Class SE (AIML) SEM III Roll No. _____Exam No._____ has performed the experiments mentioned during the academic year 2023-2024 and has successfully completed all the practicals in the subject of COMPUTER GRAPHICS as prescribed by the University of Mumbai

_____ _____

Mrs. Nirosha Uppu **Mrs. Deshpande**

PRACTICAL IN-CHARGE **HOD**

_____

Mr. Sunil Chavan

PRINCIPAL

Program Outcomes (PO):

Engineering Graduates will be able to:

1. **Engineering knowledge: Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.**
2. **Problem analysis: Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.**
3. **Design/development of solutions: Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations**
4. **Conduct investigations of complex problems: Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.**
5. **Modern tool usage: Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations.**
6. **The engineer and society: Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.**
7. **Environment and sustainability: Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.**
8. **Ethics: Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.**
9. **Individual and team work: Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.**

10. **Communication: Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.**
11. **Project management and finance: Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments**
12. **Life-long learning: Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.**

SUBMISSION

We, students of 3rd semester of second year AIML in computer science engineering humbly submit that we have completed the miniproject work as prescribed in this report by own skill and the study in academic session 2023-24, as per the instruction and guidance of **NIROSHA MA'AM**

The work carried out in the project is our and not copied the report on any appreciable part from any other literature in contravention of the academic ethics. The teacher has approved our contribution.

The students associated in the Mini project are:

1) Bhavesh Choudhari
2) Om Chaudhari
3) Sumit Dey

PART B

Simulation of Solar System

1.0 **Brief Description**

In this program, We created an animated simulation of planets orbiting a central sun using graphics in C, with predefined functions for drawing shapes and managing the display.

Function Name	Description
initgraph	Initializes graphics system with specified driver and mode
graphresult	Returns result of most recent graphics operation
getmaxx	Returns maximum x-coordinate of current graphics mode
getmaxy	Returns maximum y-coordinate of current graphics mode
setcolor	Sets current drawing color
setfillstyle	Sets current fill pattern and color
circle	Draws a circle on the screen
floodfill	Fills an area with current fill pattern and color
ellipse	Draws an ellipse on the screen
pieslice	Draws a pie-shaped section of an ellipse
delay	Pauses program for specified number of milliseconds
kbhit	Checks if a keyboard key has been pressed
cleardevice	Clears the graphics screen
closegraph	Closes graphics system and deallocates resources

2.0 **Aim for the micro-Project**

This micro-project aims at:

1. Create a replica of solar system
2. Create and apply graphics to animate the revolution of planets in Solar System.

ABSTRACT

This C program, rooted in graphics programming, explores the captivating realm of planetary motion. Utilizing the Turbo C++ environment, it orchestrates a dynamic display of nine planets revolving around a central sun. The heart of this simulation lies in the parametric equations governing ellipses, dictating each planet's precise trajectory. Vivid colors and distinct fill patterns enhance the visual representation, injecting life into the celestial dance. Predefined functions, from drawing shapes to screen manipulation, are our trusty tools in this endeavor. The simulation continues until a key press, offering an engaging glimpse into the intricacies of celestial mechanics. This program, as an engineering student, not only fascinates but also lays the groundwork for more ambitious simulations and educational tools concerning planetary motion and orbital mechanics.

Actual Resource Used:

Sr No.	Name of resource	Specifications	Quantity
1	Hardware: Computer System	**Processor:** 11th Gen Intel(R) Core(TM) i5-1155G7 @ 2.50GHz 2.50 GHz **RAM:** 8.00 GB (7.65 GB usable)	1
2	Operating System	Windows 10	1
3	Software	Turbo C	1

DESCRIPTION

This C program is like a virtual space show created by an engineering enthusiast. It's all about making planets move around a virtual sun using computer graphics. Think of it as a little space simulation!

We use a special tool called Turbo C++ to do this. It helps us draw things on the screen. In our show, we have nine planets orbiting a central 'sun'. We calculate how each planet moves in its path using some math tricks called parametric equations. It's like figuring out a special dance routine for each planet.

To make things look cool, we add colors and patterns. So, when you run the program, you'll see colorful planets dancing around.

We have some pre-made tools in Turbo C++ that help us with tasks like drawing shapes, filling colors, and clearing the screen. These tools save us a lot of time and effort.

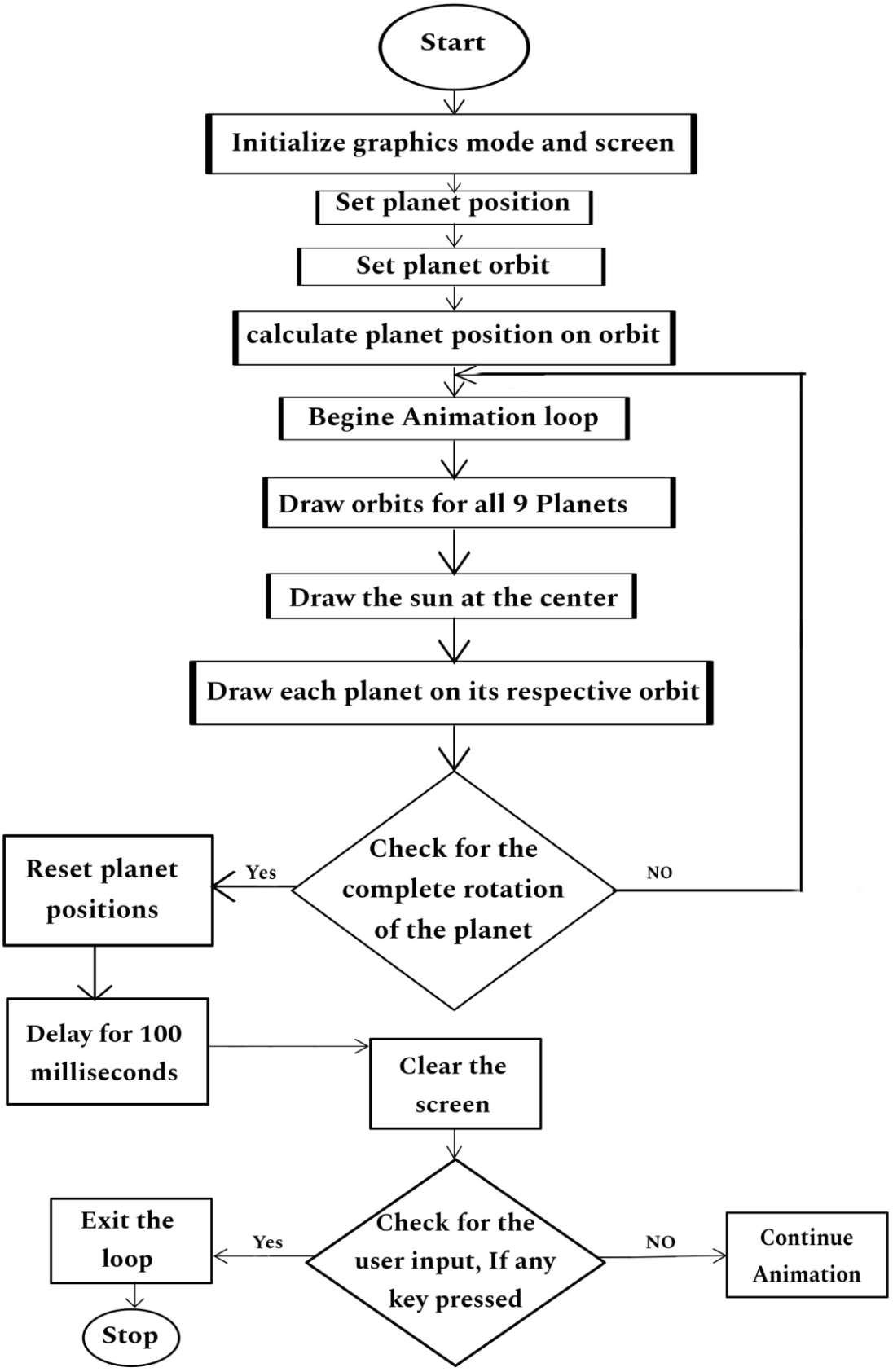
The show keeps going until you press a key, giving you a chance to watch the planets for as long as you'd like. It's a fun way to explore a bit about how planets move in space.

For me, as an engineering student, this program is not just fun, but also a great way to learn about graphics programming and a bit of physics about how planets move around stars. It sets the stage for more complex simulations and educational tools about space.

ALGORITHM:

1. Start
2. Initialize graphics mode and screen.
3. Set the positions of the solar system's celestial bodies (planets, sun).
4. Define the orbits and radius of the planets.
5. Calculate the positions of planets on their orbits using trigonometric functions.
6. Continuously loop until a key is pressed:
    a. Draw the orbits for all 9 planets.
    b. Draw the sun at the center of the solar system.
    c. Draw each planet on its respective orbit.
    d. Check for a complete rotation of the planets.
    e. Delay for 100 milliseconds and clear the screen.
7. Deallocate memory used for the graphic screen and exit.
8. Stop

Flowchart:


Start
Initialize graphics mode and screen
Set planet position
Set planet orbit
calculate planet position on orbit
Begine Animation loop
Draw orbits for all 9 Planets
Draw the sun at the center
Draw each planet on its respective orbit
Check for the complete rotation of the planet
Yes
NO
Reset planet positions
Delay for 100 milliseconds
Clear the screen
Check for the user input, If any key pressed
Yes
NO
Exit the loop
Stop
Continue Animation

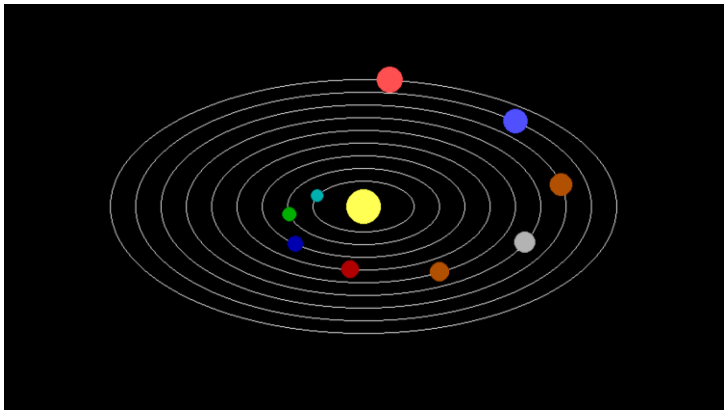
CODE:

```
#include <stdio.h>
#include <conio.h>
#include <graphics.h>
#include <dos.h>
#include <math.h>
void planetMotion(int xrad, int yrad, int midx, int midy, int x[60], int y[60]) {
    int i, j = 0;
    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;
}
int main() {
    int gdriver = DETECT, gmode, err;
    int i = 0, midx, midy;
    int xrad[9], yrad[9], x[9][60], y[9][60];
    int pos[9], planet[9], tmp;
    initgraph(&gdriver, &gmode, "C:/TURBOC3/BGI");
    err = graphresult();
    if (err != grOk) {
        printf("Graphics Error: %s",
                grapherrormsg(err));
        return 0;
    }
    midx = getmaxx() / 2;
    midy = getmaxy() / 2;
    planet[0] = 7;
    for (i = 1; i < 9; i++) {
```

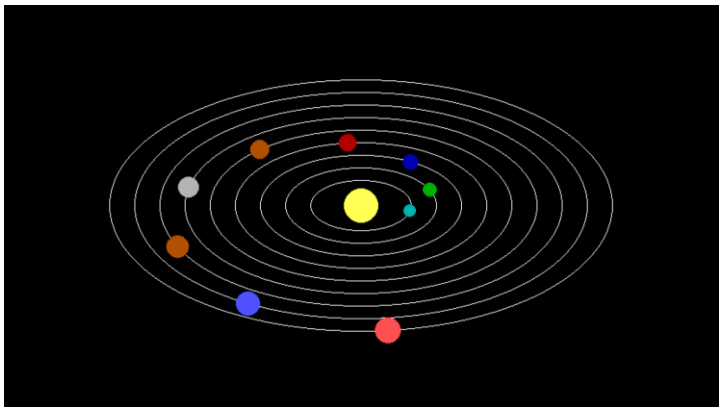
```
        planet[i] = planet[i - 1] + 1;
}
for (i = 0; i < 9; i++) {
        pos[i] = i * 6;
}
xrad[0] = 60, yrad[0] = 30;
for (i = 1; i < 9; i++) {
        xrad[i] = xrad[i - 1] + 30;
        yrad[i] = yrad[i - 1] + 15;
}
for (i = 0; i < 9; i++) {
        planetMotion(xrad[i], yrad[i], midx, midy, x[i], y[i]);
}
while (!kbhit()) {
        setcolor(WHITE);
        for (i = 0; i < 9; i++) {
                ellipse(midx, midy, 0, 360, xrad[i], yrad[i]);
        }
        setcolor(YELLOW);
        setfillstyle(SOLID_FILL, YELLOW);
        circle(midx, midy, 20);
        floodfill(midx, midy, YELLOW);
        setcolor(CYAN);
        setfillstyle(SOLID_FILL, CYAN);
        pieslice(x[0][pos[0]], y[0][pos[0]], 0, 360, planet[0]);
        setcolor(GREEN);
        setfillstyle(SOLID_FILL, GREEN);
        pieslice(x[1][pos[1]], y[1][pos[1]], 0, 360, planet[1]);
        setcolor(BLUE);
        setfillstyle(SOLID_FILL, BLUE);
        pieslice(x[2][pos[2]], y[2][pos[2]], 0, 360, planet[2]);
```

```
            setcolor(RED);
            setfillstyle(SOLID_FILL, RED);
            pieslice(x[3][pos[3]], y[3][pos[3]], 0, 360, planet[3]);
            setcolor(BROWN);
            setfillstyle(SOLID_FILL, BROWN);
            pieslice(x[4][pos[4]], y[4][pos[4]], 0, 360, planet[4]);
            setcolor(LIGHTGRAY);
            setfillstyle(SOLID_FILL, LIGHTGRAY);
            pieslice(x[5][pos[5]], y[5][pos[5]], 0, 360, planet[5]);
            setcolor(BROWN);
            setfillstyle(SOLID_FILL, BROWN);
            pieslice(x[6][pos[6]], y[6][pos[6]], 0, 360, planet[6]);
            setcolor(LIGHTBLUE);
            setfillstyle(SOLID_FILL, LIGHTBLUE);
            pieslice(x[7][pos[7]], y[7][pos[7]], 0, 360, planet[7]);
            setcolor(LIGHTRED);
            setfillstyle(SOLID_FILL, LIGHTRED);
            pieslice(x[8][pos[8]], y[8][pos[8]], 0, 360, planet[8]);
            for (i = 0; i < 9; i++) {
                  if (pos[i] <= 0) {
                        pos[i] = 59;
                  } else {
                        pos[i] = pos[i] - 1;
                  }
            }
            delay(100);
            cleardevice();
      }
      closegraph();
      return 0;
}
```

OUTPUT SCREENSHOT:



The screenshot exhibits nine planets orbiting a central sun in vibrant colors. This dynamic display offers a simplified yet captivating portrayal of planetary motion within a DOS-based graphics environment.



The screenshot displays planets in motion, tracing their orbits around the central sun. This dynamic animation vividly illustrates planetary movement, providing an engaging visual representation of celestial mechanics.

CONCLUSIUON:

To sum it up, this program is all about showing how planets move in space. By using graphics and math, it makes nine planets go around a pretend sun. As an engineering student, it's like a hands-on lesson in how things move in space. It's cool to see the planets circling around. This project connects what I learn in class with a real-life simulation. It shows how science and technology work together to teach us about the universe. It's been a fun and educational experience, and it makes me more curious about space and engineering.