**Part : B Microproject Report**

1. **Summary**

In this project, we did some animations by using the <graphics.h> header file.

1. **Course action addressed**

* Manipulate visual and geometric information of images.
* Implement Polygon Algorithm.
* Implement standard Algorithms to draw various Graphics Objects using C

1. **Actual methodology**
2. **Algorithm:**

Step 1: Start.

Step 2: Initialize the graphics library.

Step 3: Create a window of a desired size.

Step 4: Draw a circle to represent the sun in the center of the window.

Step 5: Draw a set of concentric circles around the sun to represent the orbits of the planets.

Step 6: Draw each planet along its respective orbit.

Step 7: Add details such as rings (for Saturn), moons, clouds, etc.

Step 8: Use color and shading to make the scene more realistic.

Step 9: Add a light source to simulate the sun’s light.

Step 10: Simulate the motion of the planets by changing their positions on the orbits.

Step 11: End.

**b.)Flowchart:**



**c.) Source code:**

#include <stdio.h>

#include <stdlib.h>

#include <GL/glut.h>

#include <math.h>

#define PI 3.1416

// Define global variables

GLfloat hour = 0.0, minute = 0.0, second = 0.0, millisecond = 0.0;

GLfloat days = 0.0, months = 0.0, years = 0.0;

// function to initialize

void init(void)

{

// glClearColor (red, green, blue, alpha)

glClearColor(0.0, 0.0, 0.0, 0.0);

glMatrixMode(GL\_PROJECTION);

glLoadIdentity();

gluOrtho2D(-1.0, 1.0, -1.0, 1.0);

}

// Function to draw Objects

void drawObjects(void)

{

// Draw Sun

glColor3f(1.0, 1.0, 0.0);

glPushMatrix();

glTranslatef(0.0, 0.0, -1.0);

glutWireSphere(0.1, 20, 16);

glPopMatrix();

// Draw 2D grid along X-Y axis

glColor3f(1.0, 1.0, 1.0);

glPushMatrix();

glTranslatef(0.0, 0.0, -1.0);

glBegin(GL\_LINES);

// Loop to draw grid along X-axis

for (float i = -1; i <= 1; i = i + 0.001) {

glVertex3f(i, -1, 0);

glVertex3f(i, 1, 0);

}

// Loop to draw grid along Y-axis

for (float i = -1; i <= 1; i = i + 0.001) {

glVertex3f(-1, i, 0);

glVertex3f(1, i, 0);

}

glEnd();

glPopMatrix();

// Draw Moon

glColor3f(0.5, 0.5, 0.5);

glPushMatrix();

glTranslatef(0.0, 0.0, -1.0);

glRotatef(days, 0.0, 1.0, 0.0);

glTranslatef(0.2, 0.0, 0.0);

glutWireSphere(0.05, 10, 8);

glPopMatrix();

}

// Display Function

void display(void)

{

glClear(GL\_COLOR\_BUFFER\_BIT);

drawObjects();

glFlush();

}

// Reshape Function

void reshape(int w, int h)

{

glViewport(0, 0, (GLsizei)w, (GLsizei)h);

glMatrixMode(GL\_PROJECTION);

glLoadIdentity();

gluOrtho2D(-1.0, 1.0, -1.0, 1.0);

glMatrixMode(GL\_MODELVIEW);

glLoadIdentity();

}

// Timer Function

void timer(int value)

{

millisecond = millisecond + 1;

if (millisecond >= 100) {

second = second + 1;

millisecond = 0;

}

if (second >= 60) {

minute = minute + 1;

second = 0;

}

if (minute >= 60) {

hour = hour + 1;

minute = 0;

}

days = days + 5;

if (days >= 360) {

months = months + 1;

days = 0;

}

if (months >= 12) {

years = years + 1;

months = 0;

}

glutPostRedisplay();

glutTimerFunc(100, timer, 0);

}

// Main Function

int main(int argc, char\*\* argv)

{

glutInit(&argc, argv);

glutInitDisplayMode(GLUT\_RGB | GLUT\_SINGLE);

glutInitWindowSize(500, 500);

glutInitWindowPosition(100, 100);

glutCreateWindow("3D Solar System Simulation");

init();

glutDisplayFunc(display);

glutReshapeFunc(reshape);

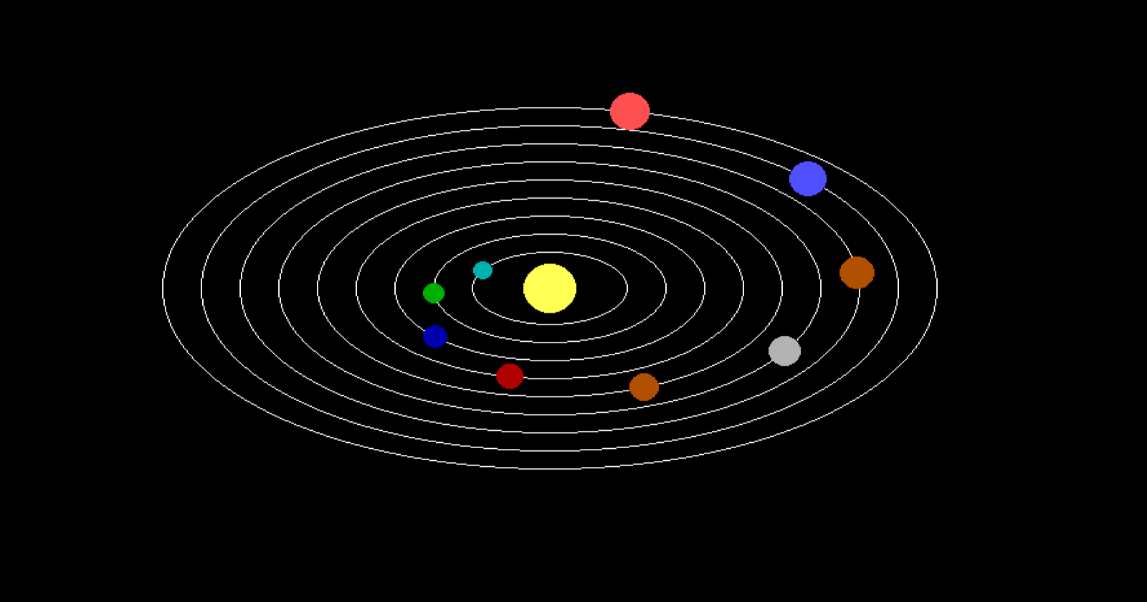
glutTimerFunc(100, timer, 0);

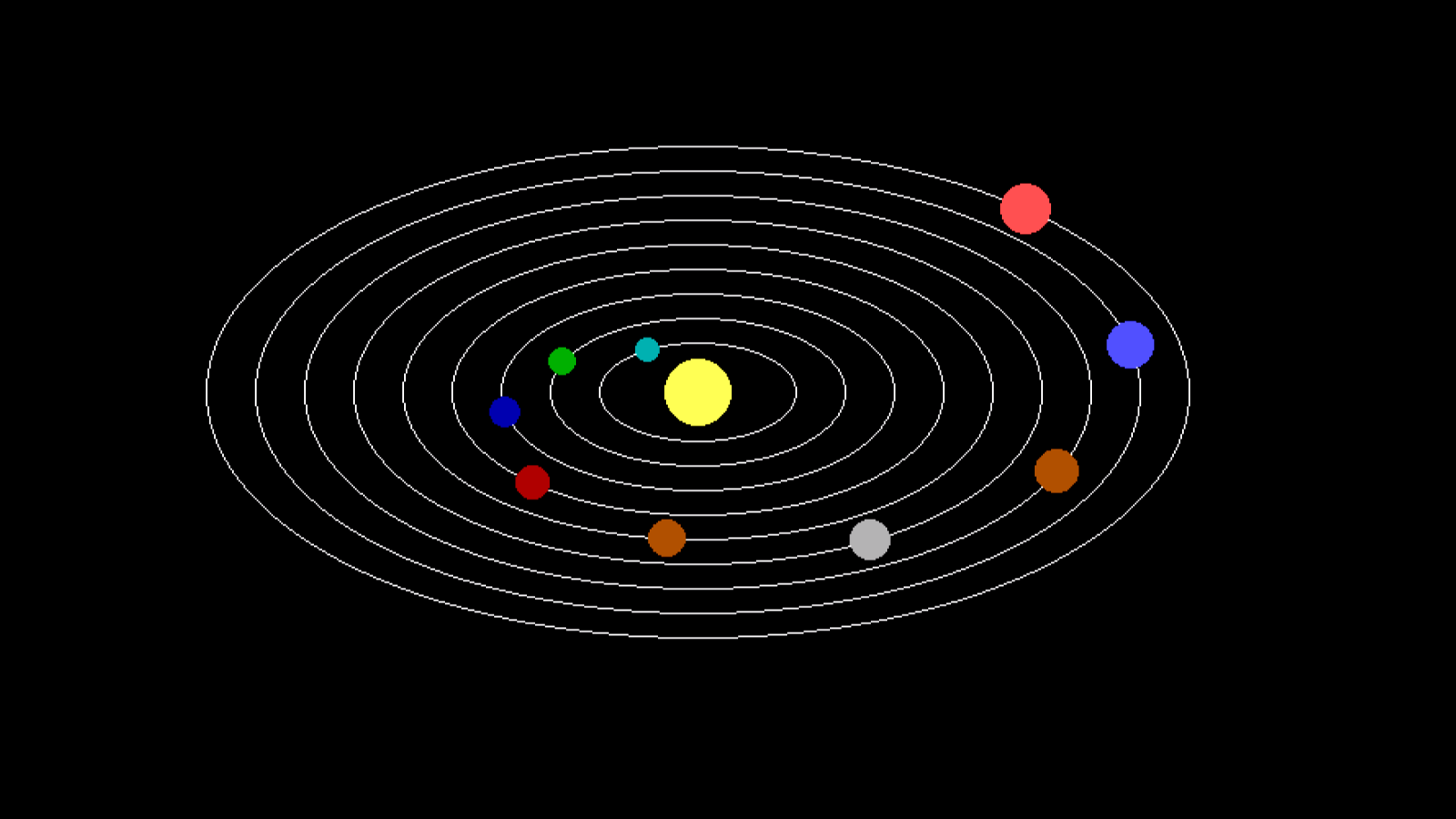
glutMainLoop();

return 0;

}

1. **Microproject output:**

****

****

1. **Actual resources used**
2. Google For References
3. TurboC3 For Coding and Compiling
4. **Skills developed**

* We developed programming skills
* We developed searching skills
* We developed out team work skills
* We developed learning skills

1. **Application of Microproject**

* Animation
* Creating objects