CHAPTER - 1

INTRODUCTION

1.1 Aim

The main aim of this project is to visually present the users with the satellite motions around a planet. This project gives a better understanding of the different types of orbits of satellites around a planet and it also allows the user to view the planets in different angles for better understanding of the revolution of satellites around it.

1.2 Overview

The planetary motions and the motion of satellites are governed by Kepler's laws of planetary motion and Newton's law of gravitation. While Copernicus rightly observed that the planets revolve around the Sun, it was Kepler who correctly defined their orbits and formulated these laws.

There have been several misconceptions about these motions starting from people who believe in geocentric model to people who believe that the earth is flat. Some people who believe in heliocentric model don't have a clear picture about it. This project aims to enlighten the user by simulating and allowing them to visualize the motion of a satellite around a planet and helps them to get a better comprehension about satellite motions.

1.3 Outcome

The outcome of the project is that it visualizes the different types of motions of a satellite around a planet. It presents the users with the option of viewing it from different angles and controlling the type of motion of the satellite, hence giving a clear picture about it.

CHAPTER – 2

DESIGN AND IMPLEMENTATION

2.1 Algorithm

```
Input:  User gives input through the mouse buttons.

Output: Different views or motions of the satellites are selected based on the input.

Step 1: Open and run the project.

Step 2: Record Input from the user. (Left mouse click or Right mouse click)

Step 3: If the input is left mouse click then:

        3a: If the input is 'Move Artificial Satellite Forward' then:

                Make artificial satellite move one step in anti-clockwise direction.

        3b: If the input is 'Move Artificial Satellite Backward' then:

                Make artificial satellite move one step in clockwise direction.

Step 4: If the input is right mouse click then:

        4a: If the input is 'Retrograde Motion' then:

                Make satellite revolve in retrograde motion.

        4b: If the input is 'Non Retrograde Motion' then:

                Make satellite revolve in non-retrograde motion.

        4c: If the input is 'Angular Motion' then:

                Make satellite revolve at an angle not perpendicular to the axis.

        4d: If the input is 'Non Angular Motion' then:

                Make satellite revolve at an angle perpendicular to the axis.

        4e: If the input is 'Views' then:

                If the input is 'Front View' then:

                    Display the front view.
```

```
            If the input is ‘Top View’ then:
                  Display the top view.
            If the input is ‘Side View’ then:
                  Display the side view.
      4f: If the input is ‘Quit’ then:
            Terminate the program.
Step 5: Stop.
```

2.2 Flowchart

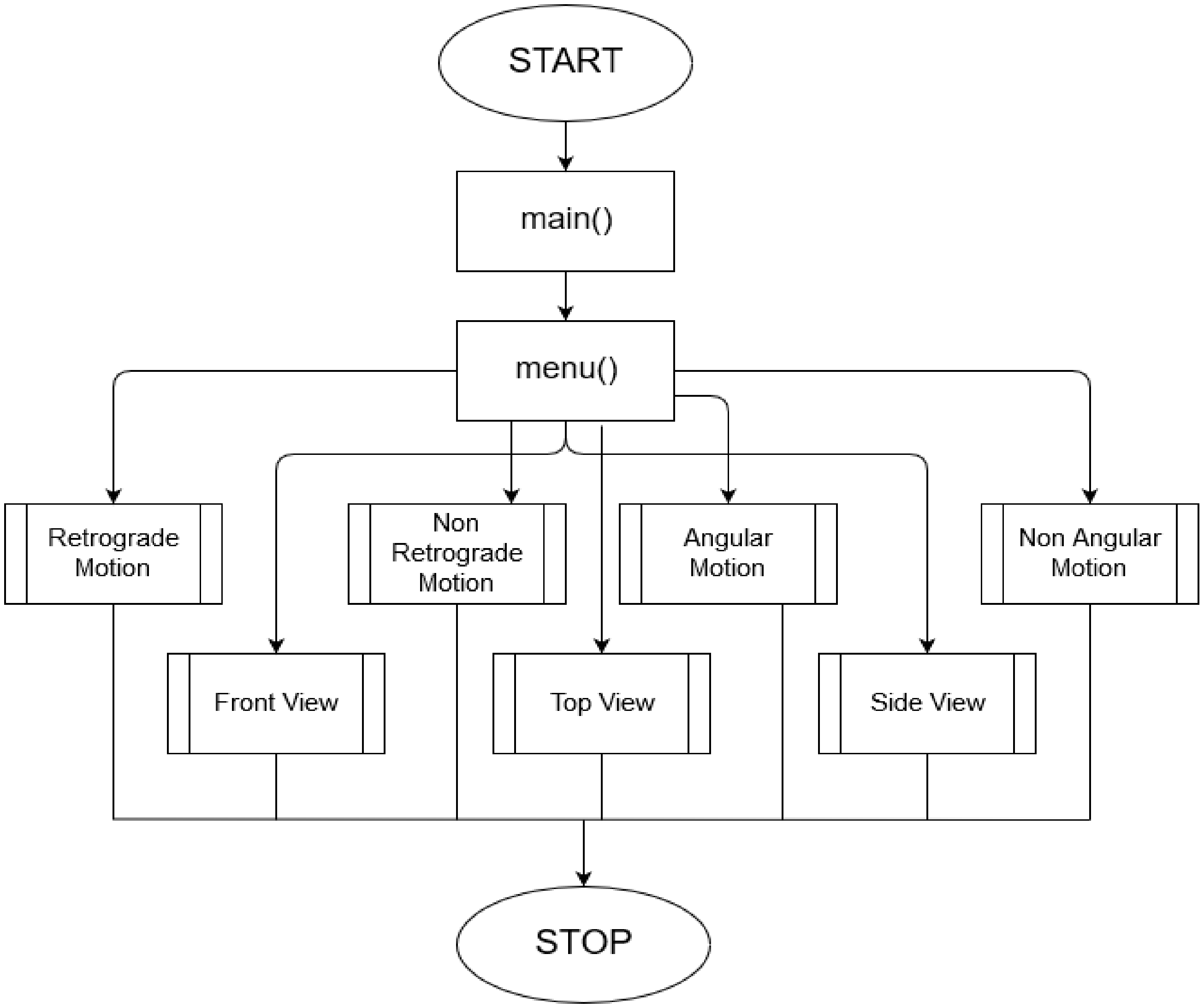

Fig 2.1 Flowchart

The Fig2.1 depicts the flowchart for the menu functions used in this project.

2.3 OpenGL API's Used With Description

- **Glut** : An introduction to the OpenGL Utility Toolkit
- **GlutInit( )** : Initialize the GLUT library and graphics system.
  - Declaration: void GlutInit (int argc, char **argv);
  - Description: To start the graphics system, you must first call GlutInit ( ), GlutInit will initialize the GLUT library and negotiate a session with the window system. During this process, GlutInit may cause the termination of the GLUT program with an error message to the user if GLUT cannot be properly initialized.
- **glutInitDisplayMode( )** : Sets the initial display mode.
  - Declaration: void glutInitDisplayMode(unsigned int mode);
  - Description: The initial display mode is used when creating top-level windows, sub windows, and overlays to determine the OpenGLdisplay mode for the to-be-created window or overlay.
- **glTranslate( ):** Alters the current matrix by a displacement of(x,y,z).
  - TYPE is either GLfloat or GLdouble.
- **glutCreateMenu**: Returns an identifier for a top level menu and register the callback function f that returns an integer value corresponding to the menu entry selected.
- **glutDisplayFunc**: Register the display function that is executed when the window needs to be redrawn.
- **glutPostRedisplay( )**: Request that the display callback be executed after the current callback returns.
- **glutMainLoop( )**:
  - Cause the program to enter an event processing loop. It should be the last statement in main( ).
- **glPushMatrix( ) and glPopMatrix( )**:

  - Pushes to and pops from the matrix stack corresponding to the current matrix mode.
- **glBegin( )**:
  - Initiates a new primitive of type mode and starts the collection of vertices. Values of mode include GL_POINTS,GL_LINES and GL_POLYGON.
- **glEnd( )**: Terminates a list of vertices .
- **glutReshapeFunc( )** :
  - Sets the reshape callback for the current window.
- **glutCreateWindow** ( ):
  - Creates a top-level window.
- **glutInitWindowSize** ( ): Requests future windows to open at a given width/height.
- **glOrtho( ):**
  - Declaration: Void glOrtho (GLdouble left, GLdouble right, GLdouble bottom, GLdouble top, GLdouble near, GLdouble far)
  - Description: It defines an orthographic viewing volume with all parameters measured from the center of the projection plane.
- **glViewport**( ): Set the viewport.
- **glLoadIdentity** ( ): Replace the current matrix with the identity matrix.
- **void menu(int item):** This function is used to perform menu options.
- **int main(int argc, char *argv[]):**
  - The program starts its execution from this function. And this function calls all the user defined and graphical functions.

2.4 Source Code

```
#include<windows.h>
#include<GL/glut.h>
#include <ctime>

GLfloat d=0,retrograde=0,angle=0,view=0,f=0;
// 4 Flag variables for angle and keyboard controls
static int menuRetrograde, menuAngle, menuView, menuArt;
//Variables used for menu
static int menu_id;
static int submenu_id;
static int value = 0;

GLfloat a[100]= {0.55, 0.54, 0.97, 27, 0.64, 0.14, 1.0, 0.48,
            0.73, 0.32, 0.24, 0.1, 0.37, 0.94, 0.1, 0.4, 0.74, 0.29, 0.91,
            0.63, 0.05, 0.1, 0.43, 0.02, 0.0, 0.67, 0.79, 0.43, 0.49, 0.58,
            0.41, 0.76, 0.53, 0.77, 0.44, 0.97, 0.93, 0.08, 0.88, 0.13, 0.95,
            0.44, 0.23, 0.86, 0.83, 0.37, 0.96, 0.4, 0.06, 0.67, 0.89, 0.76, 1.0,
            0.01, 0.1, 0.6, 0.66, 0.49, 0.93, 0.57, 0.13, 0.79, 0.8, 0.88, 0.91, 0.15,
            0.38, 0.59, 0.99, 0.13, 0.23, 0.52, 0.68, 0.99, 0.71, 0.53, 0.86, 0.26, 0.39,
            0.65, 0.23, 0.53, 0.66, 0.94, 0.86, 0.52, 0.86, 0.64, 0.79, 0.91, 0.78, 0.4,
            0.04, 0.24, 0.8, 0.87, 0.64
            };

void spin()
{
   d=d+0.15;
   if(d>360)
      d=0;
   if(f>360)
      f=0;
   glutPostRedisplay();
}
//To resize without changing shape
void reshape(int w, int h)
{
   GLdouble aspect = (GLdouble)w / (GLdouble)h;
   glMatrixMode(GL_PROJECTION);
   glLoadIdentity ();
   gluOrtho2D(-1.0 * aspect, 1.0 * aspect, -1.0, 1.0);
   glMatrixMode(GL_MODELVIEW);
   glViewport(0, 0, w, h);
}
//Randomly Draws points to create stars
void background()
```

```
{
   glPointSize(2);
   for(int i=0; i<100; i++)
   {
      if(a[i]>0.15&&a[i-1]>0.15&&a[i+1]>0.3)
      {
         glBegin(GL_POINTS);
         glVertex3f(a[i],a[i+1],-a[i-1]);
         glVertex3f(2*a[i],2*a[i+1],-2*a[i-1]);
         glVertex3f(a[i],2*a[i+1],-a[i-1]);
         glVertex3f(2*a[i],a[i+1],-a[i-1]);
         glVertex3f(a[i],-a[i+1],-a[i-1]);
         glVertex3f(2*a[i],-a[i+1],-a[i-1]);
         glVertex3f(a[i],-2*a[i+1],-a[i-1]);
         glVertex3f(-a[i],a[i+1],-a[i-1]);
         glVertex3f(-2*a[i],2*a[i+1],-2*a[i-1]);
         glVertex3f(-2*a[i],a[i+1],-a[i-1]);
         glVertex3f(-a[i],-a[i+1],-a[i-1]);
         glVertex3f(-2*a[i],-a[i+1],-a[i-1]);
         glVertex3f(-a[i],-2*a[i+1],-a[i-1]);
         glEnd();
      }
   }
   glFlush();
}

void drawArtSat()
{
   GLfloat Y[]={1,0.4,0};

   glPushAttrib(GL_ALL_ATTRIB_BITS);
   glPushMatrix();
   glScalef(1,0.05,1);
   glMaterialfv(GL_FRONT_AND_BACK,GL_DIFFUSE,Y);
   glutSolidSphere(0.5,30,30);
   glPopMatrix();
   glPopAttrib();

   glPushMatrix();
   glTranslatef(0,-0.5,0);
   glScalef(0.4,1,0.4);
   glutSolidCube(1);
   glPopMatrix();
   glPushMatrix();
   glTranslatef(-0.5,-0.45,0);
   glScalef(0.75,0.01,0.3);
   glutSolidCube(1);
   glPopMatrix();
   glPushMatrix();
   glTranslatef(0.5,-0.45,0);
   glScalef(0.75,0.01,0.3);
   glutSolidCube(1);
   glPopMatrix();

   glPushMatrix();
   glTranslatef(0,0.1,0);
   glScalef(0.01,0.5,0.01);
   glutSolidCube(1);
   glPopMatrix();
}

//Front view of the scene
void frontview()
{
   glClearColor(0,0,0,0);
   glClear(GL_COLOR_BUFFER_BIT|GL_DEPTH_BUFFER_BIT);

   //Light Source
   glLightModeli(GL_LIGHT_MODEL_LOCAL_VIEWER,GL_TRUE);
   glEnable(GL_LIGHTING);
   glEnable(GL_LIGHT0);
   GLfloat AmbientLight[]   = {0.1, 0.1, 0.1, 1.0};  //R,G,B,Alpha
   GLfloat DiffuseLight[]   = {0.5f, 0.5f, 0.5f, 0.1f}; //R,G,B,Alpha
   GLfloat LightPosition[]  = {-0.9,0.2, 0.9, 0.1}; // Set the light position
   glLightfv(GL_LIGHT0,GL_AMBIENT,AmbientLight);
   glLightfv(GL_LIGHT0,GL_DIFFUSE,DiffuseLight);
   glLightfv(GL_LIGHT0, GL_POSITION,LightPosition);

   //Colors
   GLfloat Black[] = {0.0, 0.0, 0.0, 1.0};
   GLfloat Cyan[] = {0.0, 1.0, 1.0, 1.0};
   GLfloat White[] = {1, 1, 1, 0.5};
   GLfloat Brown[]= {0.8, .4, 0.2, 0.2};

   glPushAttrib(GL_ALL_ATTRIB_BITS);
   glPushMatrix();
   //Setting Material Properties
   glMaterialfv(GL_FRONT_AND_BACK,GL_AMBIENT,Cyan);
   glMaterialfv(GL_FRONT_AND_BACK, GL_DIFFUSE,Cyan);
   glMaterialf(GL_FRONT_AND_BACK, GL_SHININESS, 100.0);
   glRotatef(d,0,1,0);
   glutSolidSphere(0.5,32,32);
   glPopMatrix();
   glPopAttrib();
   glPushAttrib(GL_ALL_ATTRIB_BITS);
   glPushMatrix();
   //Setting Material Properties
   glRotatef(f,0,1,0);
   glRotatef(125,1,0,0.5);
   glScalef(0.05,0.05,0.05);
   glTranslatef(4,-15,5);
   drawArtSat();
   glPopMatrix();
   glPopAttrib();
```

```
if(retrograde==0)
{
    glPushMatrix();

    glMaterialfv(GL_FRONT, GL_AMBIENT, Brown);
    glMaterialfv(GL_FRONT, GL_DIFFUSE, Brown);
    glMaterialf(GL_FRONT, GL_SHININESS, 0.0);
    //Angled Motion
    if(angle==0)
        glRotatef(d,0,1,0.1);
    else //Normal Motion
        glRotatef(d,0.2,1,0.3);
    glTranslatef(0.9,0,0.1);
    glScalef(0.1,0.1,0.1);
    glutSolidSphere(0.5,32,32);
    glPopMatrix();
}
if(retrograde==1)  //Retrograde Motion
{
    glPushMatrix();
    glMaterialfv(GL_FRONT, GL_AMBIENT, Brown);
    glMaterialfv(GL_FRONT, GL_DIFFUSE, Brown);
    glMaterialf(GL_FRONT, GL_SHININESS, 0.0);
    if(angle==0)
        glRotatef(-d,0,1,0);
    else
        glRotatef(-d,0.2,1,0.3);
    glTranslatef(0.9,0,0.1);



    glScalef(0.1,0.1,0.1);
    glutSolidSphere(0.5,32,32);
    glPopMatrix();
}
glPushAttrib(GL_ALL_ATTRIB_BITS);
```

```
    glPushMatrix();
    glMaterialfv(GL_FRONT_AND_BACK, GL_AMBIENT, White);
    glMaterialfv(GL_FRONT_AND_BACK, GL_SPECULAR,White);
    glMaterialf(GL_FRONT_AND_BACK, GL_SHININESS, 00.0);
    background();
    glPopMatrix();
    glPopAttrib();
    glutSwapBuffers();
}

void topview()
{
    glClearColor(0,0,0,0);
    glClear(GL_COLOR_BUFFER_BIT|GL_DEPTH_BUFFER_BIT);
    //Light Source
    glLightModeli(GL_LIGHT_MODEL_LOCAL_VIEWER,GL_TRUE);
    glEnable(GL_LIGHTING);
    glEnable(GL_LIGHT0);
    GLfloat AmbientLight[]   = {0.1, 0.1, 0.1, 1.0};  //R,G,B,Alpha
    GLfloat DiffuseLight[]   = {0.5f, 0.5f, 0.5f, 0.1f}; //R,G,B,Alpha
    GLfloat LightPosition[]  = {-0.9, 0.9,0.2, 0.1}; // Set the light position
    glLightfv(GL_LIGHT0,GL_AMBIENT,AmbientLight);
    glLightfv(GL_LIGHT0,GL_DIFFUSE,DiffuseLight);
    glLightfv(GL_LIGHT0, GL_POSITION,LightPosition);

    //Colors
    GLfloat Black[] = {0.0, 0.0, 0.0, 1.0};
    GLfloat Cyan[] = {0.0, 1.0, 1.0, 1.0};
    GLfloat White[] = {1, 1, 1, 0.5};
    GLfloat Brown[]= {0.8, .4, 0.2, 0.2};

    glPushAttrib(GL_ALL_ATTRIB_BITS);
    glPushMatrix();
    //Setting Material Properties
    glMaterialfv(GL_FRONT_AND_BACK,GL_AMBIENT,Cyan);
    glMaterialfv(GL_FRONT_AND_BACK, GL_DIFFUSE,Cyan);
    glMaterialf(GL_FRONT_AND_BACK, GL_SHININESS, 100.0);
    glRotatef(d,0,0,1);
    glutSolidSphere(0.5,32,32);

    glPopMatrix();
    glPopAttrib();

    glPushAttrib(GL_ALL_ATTRIB_BITS);
    glPushMatrix();
    glRotatef(f,0,0,1);
    glRotatef(125,1,0.2,0);
    glScalef(0.05,0.05,0.05);
    glTranslatef(4,5,-15);
    drawArtSat();
    glPopMatrix();
    glPopAttrib();

    if(retrograde==0)
    {
       glPushMatrix();
       glMaterialfv(GL_FRONT, GL_AMBIENT, Brown);
       glMaterialfv(GL_FRONT, GL_DIFFUSE, Brown);
       glMaterialf(GL_FRONT, GL_SHININESS, 0.0);
       //Angled Motion
       if(angle==0)
          glRotatef(d,0,0.1,1);
       else //Normal Motion
          glRotatef(d,0.2,0.3,1);
       glTranslatef(0.9,0.1,0);
       glScalef(0.1,0.1,0.1);
       glutSolidSphere(0.5,32,32);
       glPopMatrix();
    }
    if(retrograde==1)  //Retrograde Motion
```

```
    {
        glPushMatrix();
        glMaterialfv(GL_FRONT, GL_AMBIENT, Brown);
        glMaterialfv(GL_FRONT, GL_DIFFUSE, Brown);
        glMaterialf(GL_FRONT, GL_SHININESS, 0.0);
        if(angle==0)
            glRotatef(-d,0,0,1);
        else
            glRotatef(-d,0.2,0.3,1);
        glTranslatef(0.9,0.1,0);
        glScalef(0.1,0.1,0.1);
        glutSolidSphere(0.5,32,32);
        glPopMatrix();
    }

    glPushAttrib(GL_ALL_ATTRIB_BITS);
    glPushMatrix();
    glMaterialfv(GL_FRONT_AND_BACK, GL_AMBIENT, White);
    glMaterialfv(GL_FRONT_AND_BACK, GL_SPECULAR,White);
    glMaterialf(GL_FRONT_AND_BACK, GL_SHININESS, 00.0);
    background();
    glPopMatrix();
    glPopAttrib();
    glutSwapBuffers();
}

void sideview()
{
    glClearColor(0,0,0,0);
    glClear(GL_COLOR_BUFFER_BIT|GL_DEPTH_BUFFER_BIT);
    //Light Source
    glLightModeli(GL_LIGHT_MODEL_LOCAL_VIEWER,GL_TRUE);
    glEnable(GL_LIGHTING);
    glEnable(GL_LIGHT0);
    GLfloat AmbientLight[]   = {0.1, 0.1, 0.1, 1.0};  //R,G,B,Alpha
    GLfloat DiffuseLight[]    = {0.5f, 0.5f, 0.5f, 0.1f}; //R,G,B,Alpha
    GLfloat LightPosition[]   = {0.9,-0.9,0.2, 0.1}; // Set the light position
    glLightfv(GL_LIGHT0,GL_AMBIENT,AmbientLight);
    glLightfv(GL_LIGHT0,GL_DIFFUSE,DiffuseLight);
    glLightfv(GL_LIGHT0, GL_POSITION,LightPosition);
    //Colors
    GLfloat Black[] = {0.0, 0.0, 0.0, 1.0};
    GLfloat Cyan[] = {0.0, 1.0, 1.0, 1.0};
    GLfloat White[] = {1, 1, 1, 0.5};
    GLfloat Brown[]= {0.8, .4, 0.2, 0.2};
    glPushAttrib(GL_ALL_ATTRIB_BITS);
    glPushMatrix();
    glMaterialfv(GL_FRONT_AND_BACK,GL_AMBIENT,Cyan);
    glMaterialfv(GL_FRONT_AND_BACK, GL_DIFFUSE,Cyan);
    glMaterialf(GL_FRONT_AND_BACK, GL_SHININESS, 100.0);
    glRotatef(d,1,0,0);
    glutSolidSphere(0.5,32,32);
    glPopMatrix();
    glPopAttrib();

    glPushAttrib(GL_ALL_ATTRIB_BITS);
    glPushMatrix();
    glRotatef(f,1,0,0);
    glRotatef(125,0.2,1,0);
    glScalef(0.05,0.05,0.05);
    glTranslatef(5,4,-15);
    drawArtSat();
    glPopMatrix();
    glPopAttrib();

    if(retrograde==0)
    {
        glPushMatrix();

        glMaterialfv(GL_FRONT, GL_AMBIENT, Brown);
```

```
    glMaterialfv(GL_FRONT, GL_DIFFUSE, Brown);
    glMaterialf(GL_FRONT, GL_SHININESS, 0.0);
    //Angled Motion
    if(angle==0)
        glRotatef(d,1,0,0.1);
    else //Normal Motion
        glRotatef(d,1,0.2,0.3);
    glTranslatef(0.1,0.9,0);
    glScalef(0.1,0.1,0.1);
    glutSolidSphere(0.5,32,32);
    glPopMatrix();
}
if(retrograde==1)  //Retrograde Motion
{
    glPushMatrix();
    glMaterialfv(GL_FRONT, GL_AMBIENT, Brown);
    glMaterialfv(GL_FRONT, GL_DIFFUSE, Brown);
    glMaterialf(GL_FRONT, GL_SHININESS, 0.0);
    if(angle==0)
        glRotatef(-d,1,0,0);
    else
        glRotatef(-d,1,0.2,0.3);
    glTranslatef(0.1,0.9,0);
    glScalef(0.1,0.1,0.1);
    glutSolidSphere(0.5,32,32);
    glPopMatrix();
}

glPushAttrib(GL_ALL_ATTRIB_BITS);
glPushMatrix();
glMaterialfv(GL_FRONT_AND_BACK, GL_AMBIENT, White);
glMaterialfv(GL_FRONT_AND_BACK, GL_SPECULAR,White);
glMaterialf(GL_FRONT_AND_BACK, GL_SHININESS, 00.0);
background();
glPopMatrix();
```

```
    glPopAttrib();
   glutSwapBuffers();
 }
void Key(unsigned char ch, int x, int y)
{
    if(ch=='r') //For Retrograde
        retrograde=1;
    if(ch=='n')  //For Normal Motion
        retrograde=0;
    if(ch=='a') //For Angled Motion
        angle=1;
    if(ch=='s') //For Non Angular Motion
        angle=0;
    if(ch=='1') //For front view
        view=0;
    if(ch=='2') //For top View
        view=1;
    if(ch=='3') //For side view
        view=2;
    //Controlling Artificial Satellite
    if(ch=='+')
        f+=5;
    if(ch=='-')
        f-=5;
    glutPostRedisplay();
}
void display()
{
    if(value == 1)
        retrograde=1;
    else if(value == 2)
        retrograde=0;
    else if(value == 3)
        view=0;
    else if(value == 4)
```

```
        view=1;
    else if(value == 5)
        view=2;
    else if(value == 6)
        angle=1;
    else if(value == 7)
        angle=0;
    else if(value == 8)
        {
            f+=5;
            value=0;
        }
    else
         if(value == 9)
        {
            f-=5;
            value=0;
        }

    if(view==0)
        frontview();
    else if(view==1)
        topview();
    else
        sideview();
    glutPostRedisplay();
}

void menu(int num)
{
  if(num == 0)
    exit(0);
  else
    value = num;
  glutPostRedisplay();
```

```
}
void createMenu(void){
   submenu_id = glutCreateMenu(menu);
   glutAddMenuEntry("Front View (Keyboard Shortcut->1)", 3);
   glutAddMenuEntry("Top View (Keyboard Shortcut->2)", 4);
   glutAddMenuEntry("Side View (Keyboard Shortcut->3)", 5);
   menu_id = glutCreateMenu(menu);
   glutAddMenuEntry("Retrograde Motion (Keyboard Shortcut->r)", 1);
   glutAddMenuEntry("Non Retrograde Motion (Keyboard Shortcut->n)", 2);
   glutAddSubMenu("Views", submenu_id);
   glutAddMenuEntry("Angular Motion (Keyboard Shortcut->a)", 6);
   glutAddMenuEntry("Non Angular Motion (Keyboard Shortcut->s)", 7);
   glutAddMenuEntry("Quit", 0);
   glutAttachMenu(GLUT_RIGHT_BUTTON);
   menu_id = glutCreateMenu(menu);
   glutAddMenuEntry("Move Artificial Satellite Forward (Keyboard Shortcut->+)", 8);
   glutAddMenuEntry("Move Artificial Satellite Backward (Keyboard Shortcut->-)", 9);
   glutAttachMenu(GLUT_LEFT_BUTTON);
}
int main(int C,char *V[])
{
   glutInit(&C,V);
   glutInitWindowSize(1366,768);
   glutInitDisplayMode(GLUT_RGB|GLUT_DOUBLE|GLUT_DEPTH);
   glutCreateWindow("Simulation Of Satellite Motions Around a Planet");
   glutDisplayFunc(display);
   glutIdleFunc(spin);
   glutReshapeFunc(reshape);
   glutKeyboardFunc(Key);
   createMenu();
   glEnable(GL_DEPTH_TEST);
   glutMainLoop();
   return 0;
}
```

CHAPTER – 3

RESULT ANALYSIS

3.1 Snapshots

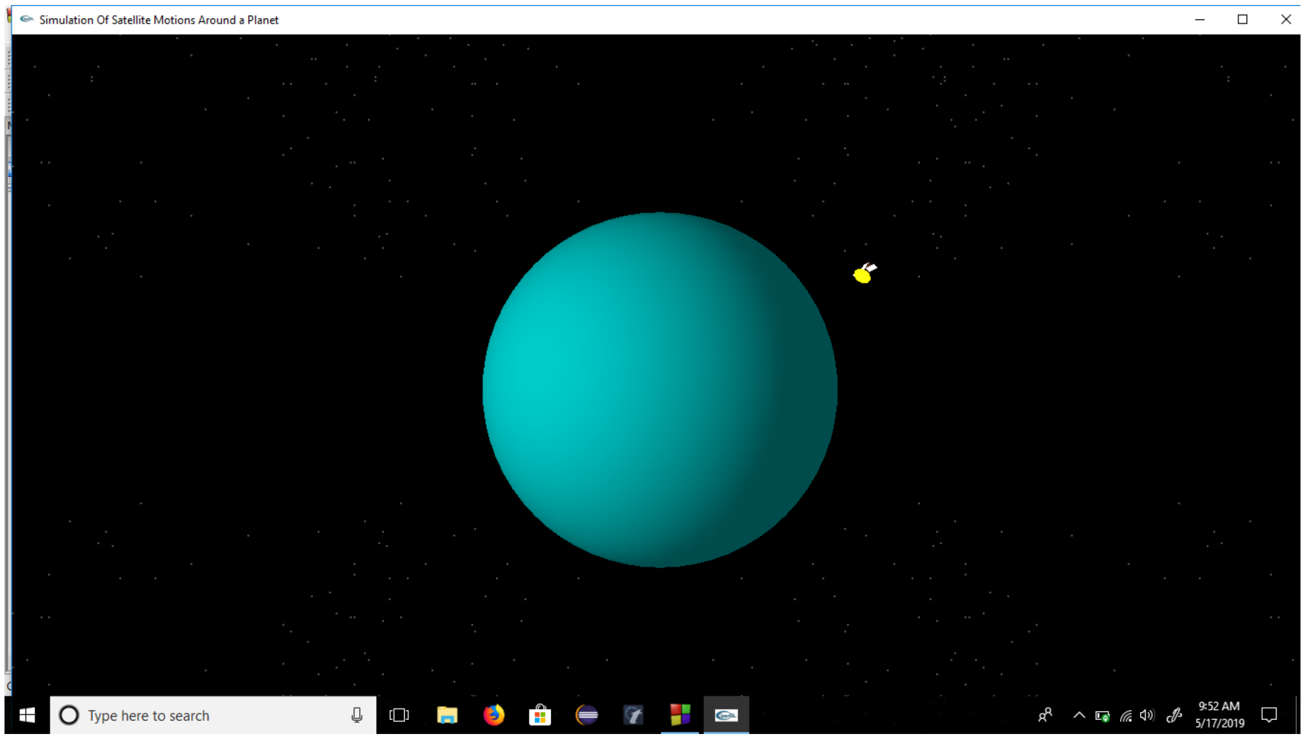

Fig 3.1 Front View

Fig 3.1 depicts the Front View of the planet and motion of the natural and artificial satellites.

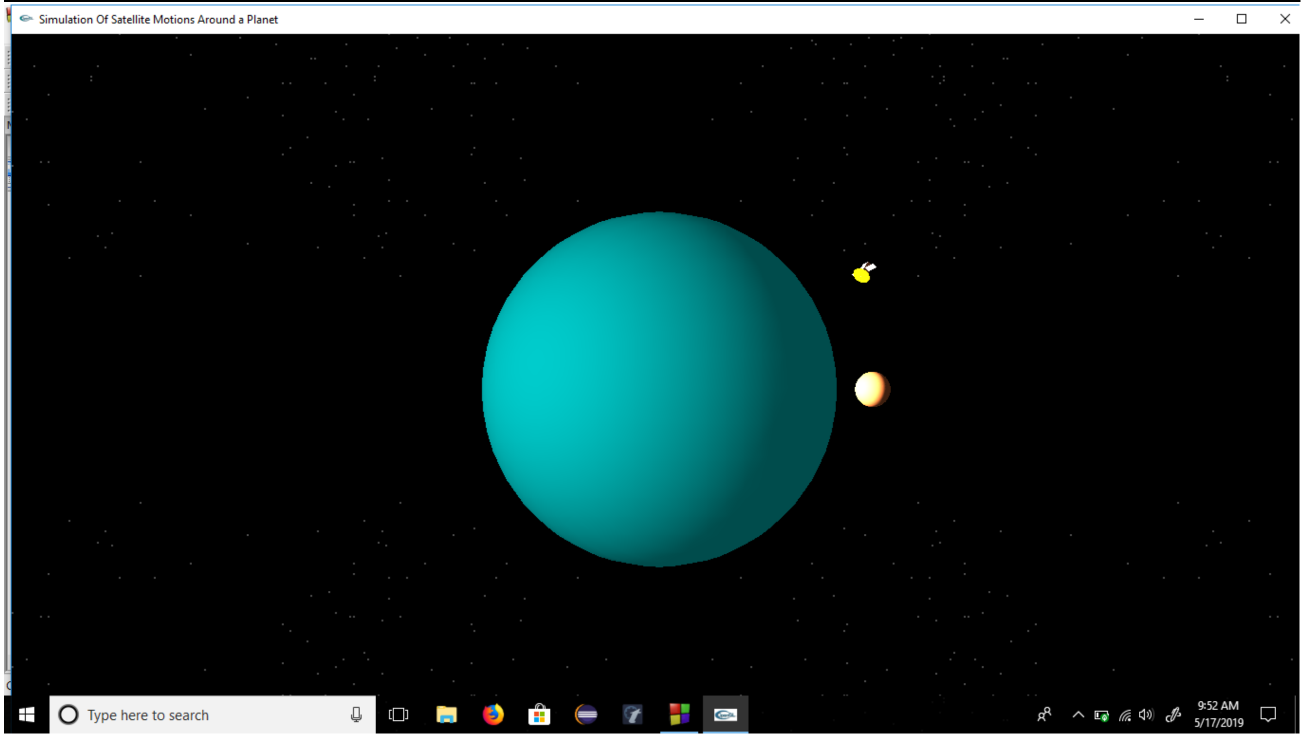

Fig 3.2 Retrograde Motion

Fig 3.2 depicts the Retrograde Motion of natural satellite.

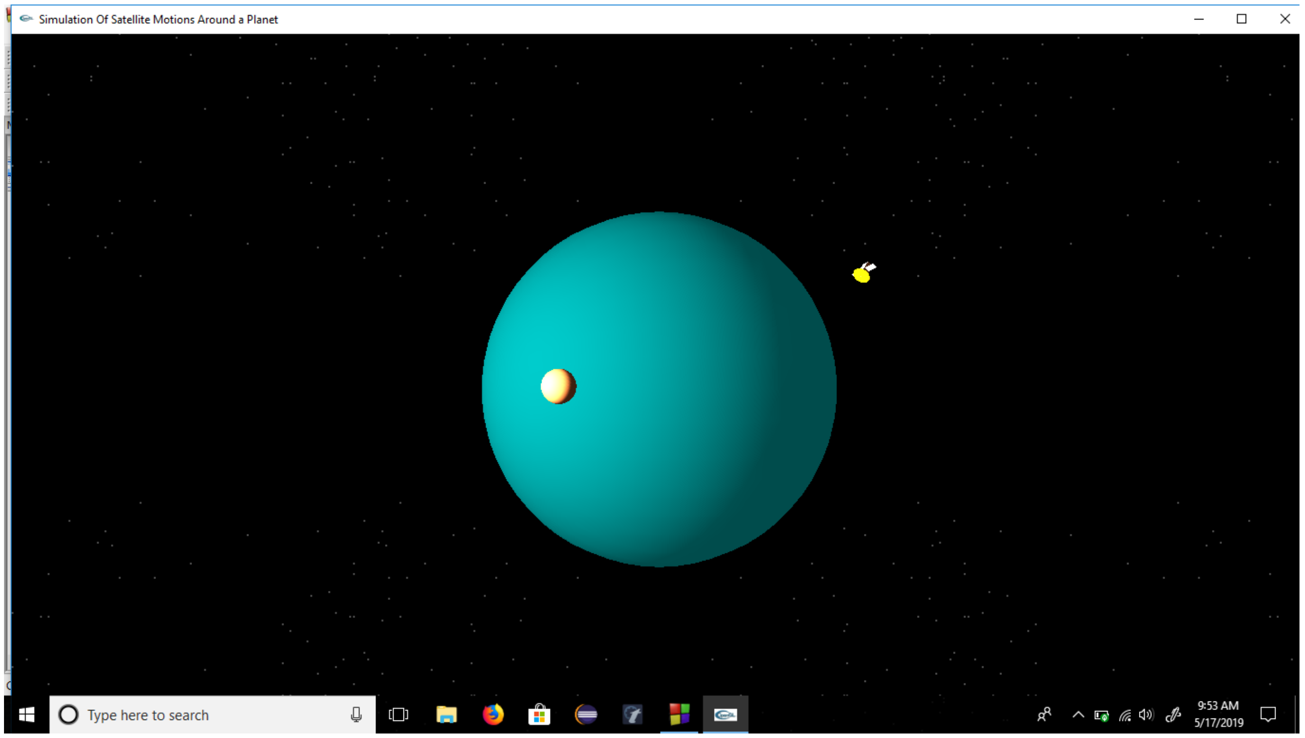

Fig 3.3 Non Retrograde Motion

Fig 3.3 depicts the Non Retrograde Motion of natural satellite.

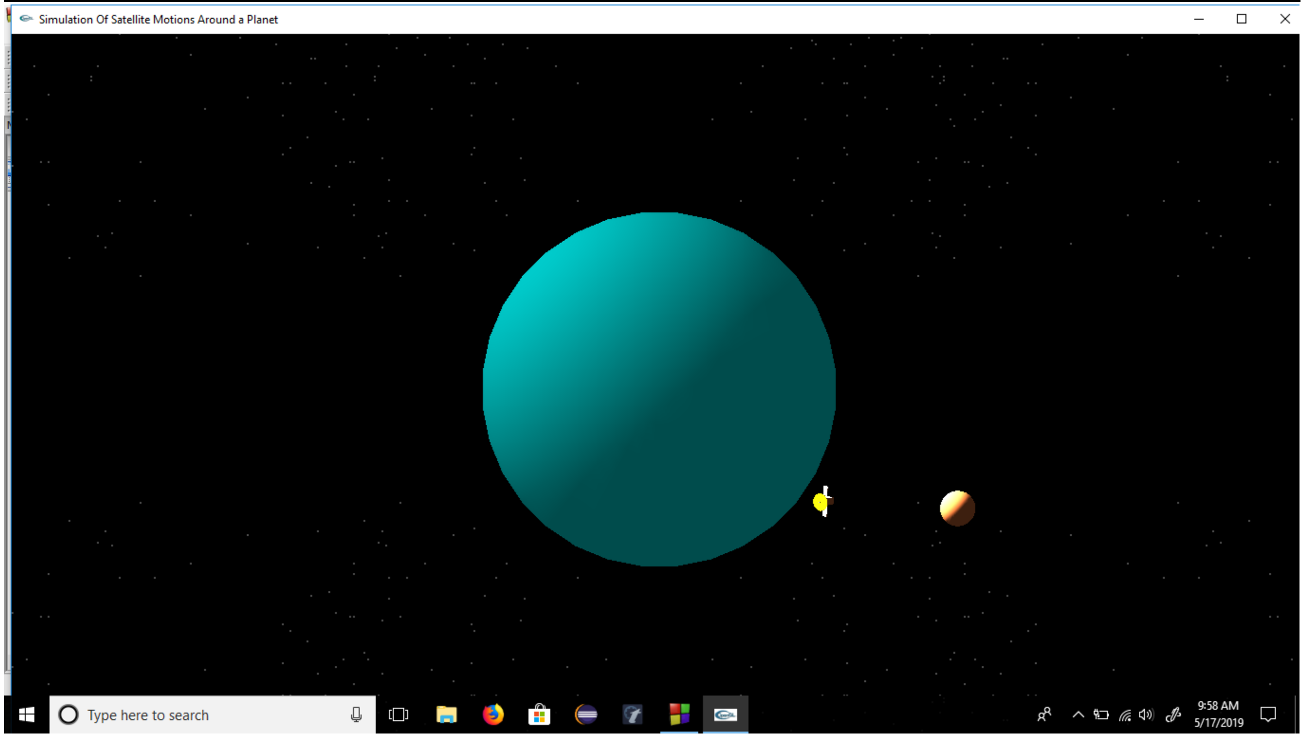

Fig 3.4 Top View

Fig 3.4 depicts the Top View of the planet and motion of the natural and artificial satellites.

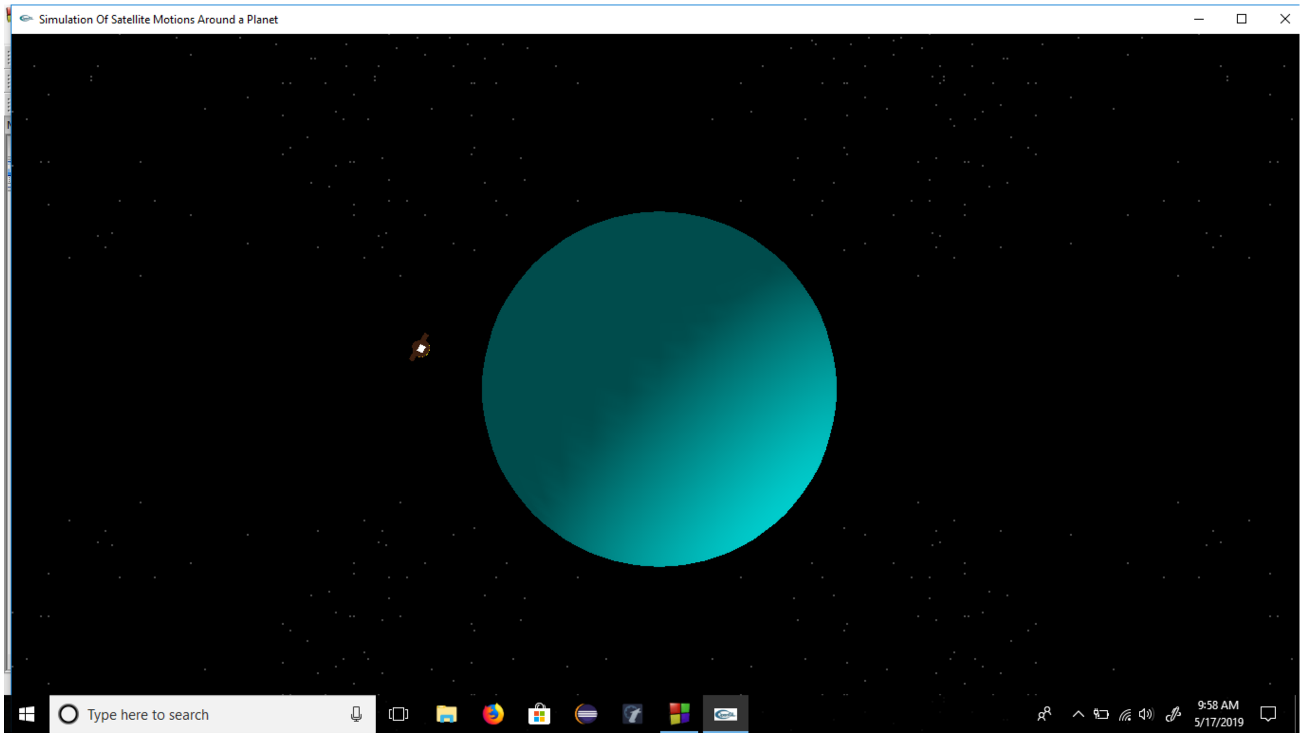

Fig 3.5 Side View

Fig 3.5 depicts the Side View of the planet and motion of the natural and artificial satellites.

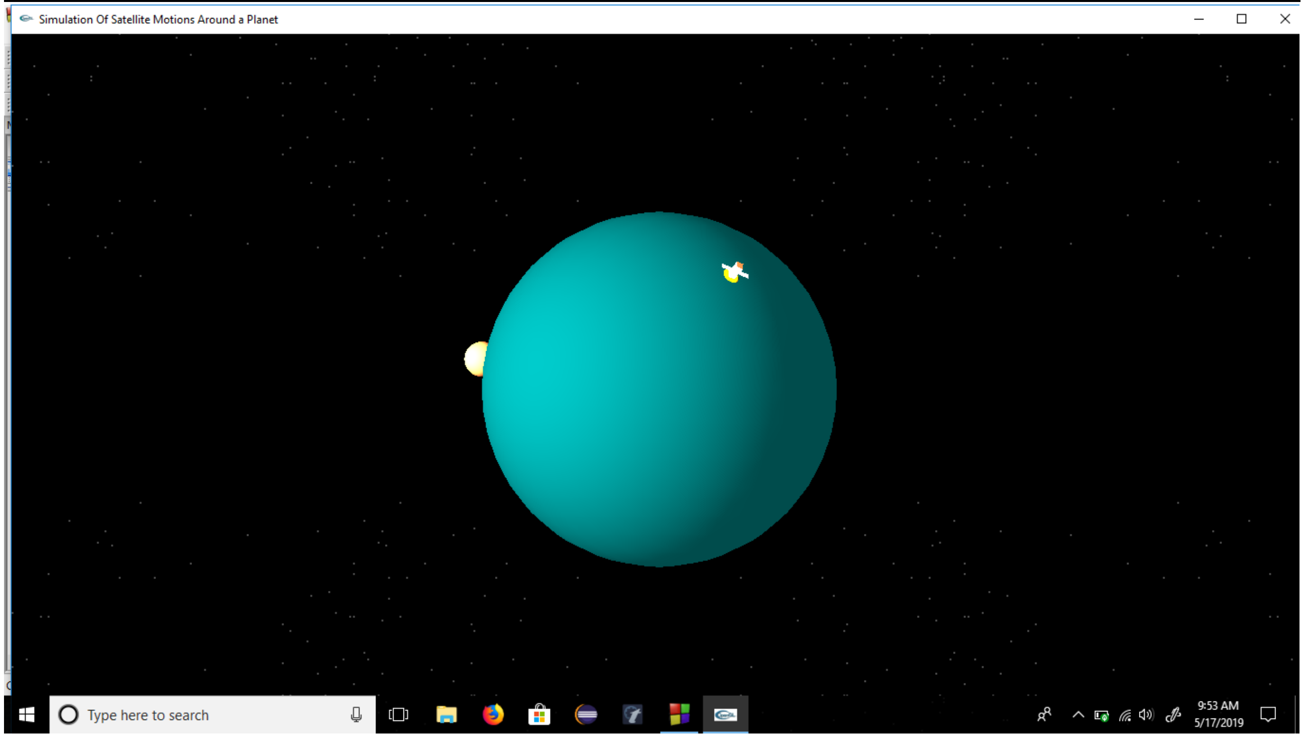

Fig 3.6 Angular Motion

Fig 3.6 depicts the Angular Motion of natural satellite.

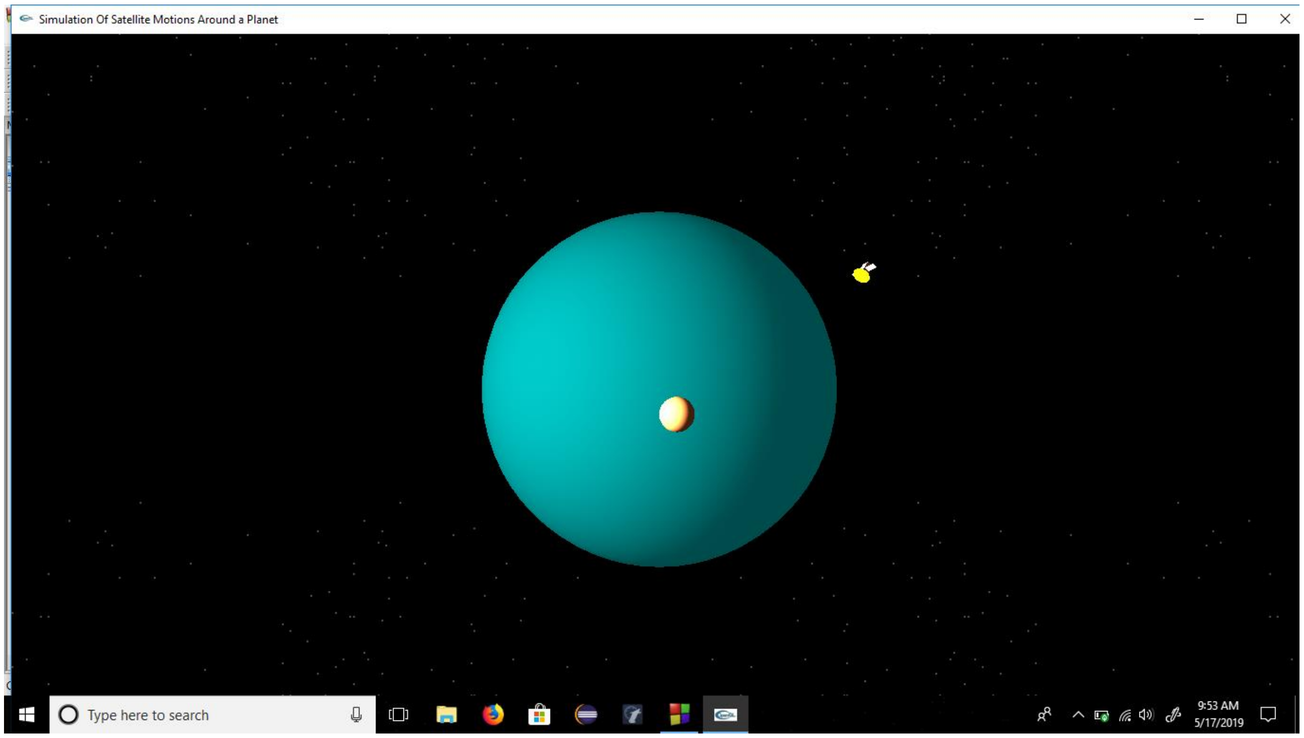

Fig 3.7 Non Angular Motion

Fig 3.7 depicts the Non Angular Motion of natural satellite.

3.2 Discussion

Computer Graphics is one of the most exciting and rapidly growing computer fields. It is also an extremely effective medium for communication between user and computers. It displays the information in the form of graphical objects such as pictures, charts, graphs instead of simple text. Humans can understand information content of a displayed diagram or perspective view much faster than it can understand the table of numbers. Computer graphics started with a display of data on hard copy plotters and cathode ray tube screens soon after the introduction of computers itself.

Graphics provides one of the most natural means for communicating with the computer, since highly developed 2D and 3D pattern recognition ability allow us to perceive and process data rapidly and efficiently.

This project demonstrates the interaction of user with the system by clicking on either the left mouse button or the right mouse button and obtaining the outcome corresponding to the user's inputs. The outcome of this project is that it visualizes to the users the different types of motions of a satellite around a planet. It presents the users with the option of viewing it from different angles and controlling the type of motion of the satellite, hence giving a clear picture about it.

CHAPTER – 4

CONCLUSION AND FUTURE WORK

4.1 Conclusion

This project helps to understand the concepts behind OpenGL and its programming. This project gives a brief insight as to how programs, involving graphics, are written using OpenGL.

This project demonstrates how the OpenGL can be used to implement graphics which can be applied in various fields such as advertising industries to endorse company's products animation studios to make animated films, cartoons, gaming industries by displaying the use of high-end graphics in designing games, in engineering, architecture, medical fields by creating real-world models for better understanding, clarity and bringing out fresh, new ideas to enhance them.

The conclusion from this project is that it describes the motions of a satellite around a planet clearly so that the users can understand them without any doubts.

4.2 Future Enhancement

This project attempts to show the motion of just one natural satellite and one artificial satellite. This can be extended to more than one. The whole model can be extended to also include the entire solar system.

This project uses limited mouse and keyboard functions for user interaction and as a future enhancement this project can make use of more keyboard functions to select the number of satellites and select different planets of the solar system.

CHAPTER – 5

REFERENCES

[1] Interactive Computer Graphics Pearson Education 5th Edition**:** Edward Angel

[2] OpenGL Documentation

[3] https://learnopengl.com

[4] https://www.youtube.com

[5] Computer Graphics with OpenGL Version, 4thEdition: Donald Hearn, Pauline Baker