Course project

روان ناصر إدريس

19015686

Code :

I save the initial values (x, y, z, r, color) of every planet in an array to call it again

```
// Initialize global arrayOfPlanets.
//Mercury
Planet Mercury( x: 3.8, y: 0.0, z: 0,  r: 0.3, scale: 1.0, colorR: 1.0, colorG: 0.9, colorB: 0.0);  arrayOfPlanets[0] = Mercury;
//Venus
Planet Venus( x: 5.0, y: 0.0, z: 0, r: 0.5, scale: 1.0 , colorR: 0.9, colorG: 0.1, colorB: 0.0);    arrayOfPlanets[1] = Venus;
//Earth
Planet Earth( x: 7.0, y: 0.0, z: 0, r: 1.0, scale: 1.0 , colorR: 0.0, colorG: 0.1, colorB: 0.7);    arrayOfPlanets[2] = Earth;
//Mars
Planet Mars( x: 9.8, y: 0.0, z: 0, r: 0.7, scale: 1.0, colorR: 0.6, colorG: 0.2, colorB: 0.3);      arrayOfPlanets[3] = Mars;
//Jupiter
Planet Jupiter( x: 12.4, y: 0.0, z: 0,  r: 1.2, scale: 1.0, colorR: 0.6, colorG: 0.7, colorB: 0.3); arrayOfPlanets[4] = Jupiter;
//Saturn
Planet Saturn( x: 15.5, y: 0.0, z: 0,  r: 1.0, scale: 1.0, colorR: 0.9, colorG: 0.0, colorB: 0.0);  arrayOfPlanets[5] = Saturn;
//Uranus
Planet Uranus( x: 17.9, y: 0.0, z: 0, r: 0.7, scale: 0.7, colorR: 0.2, colorG: 0.5, colorB: 0.7);    arrayOfPlanets[6] = Uranus;
//Neptune
Planet Neptune( x: 20, y: 0.0, z: 0,  r: 0.6, scale: 0.9, colorR: 0.2, colorG: 0.8, colorB: 0.5);    arrayOfPlanets[7] = Neptune;
```

Function to draw orbits for planets

```
void display_orbit()
{
    glColor3f( red: 0.5, green: 0.5, blue: 0.5);

    int i=0;
    for(i=0;i<8;i++){
        glPushMatrix();
        glRotatef( angle: 90, x: 1.0, y: 0.0, z: 0.0);
        glScalef( x: scale_array[i], y: scale_array[i], z: scale_array[i]);
        glBegin( mode: GL_POINTS);

        double rotation_deg2=0.0;
        int i=0;
        for(i=0;i<300;i++)
        {
            glVertex2d( x: cos( X: rotation_deg2), y: sin( X: rotation_deg2));
            rotation_deg2+=rotation_deg1;
        }

        glEnd();
        glPopMatrix();
    }
}
```

Function draw to draw every element in the array and put moon when it draw the earth planet

```
if (m) // If asteroid exists.
{
    glPushMatrix();
    glRotatef( angle: ang, x: 0.0, y: 1.0, z: 0);
    glTranslatef( x: centerX,  y: centerY,  z: centerZ);
    glColor3d( red: c1, green: c2, blue: c3);
    glScalef( x: s, y: s, z: s);
    glutSolidSphere(radius, slices: 50, stacks: 50);

    glRotatef( angle: Moon_rotate, x: 0, y: 1, z: 0.5);
    glTranslatef( x: centerX-5.5,  y: centerY,  z: centerZ);
    glColor3d( red: 1, green: 1, blue: 1);
    glScalef( x: 0.35, y: 0.35, z: 0.35);
    glutSolidSphere( radius: 0.8, slices: 50, stacks: 50);
    glPopMatrix();
}
else
{
    glPushMatrix();
    glRotatef( angle: ang, x: 0.0, y: 1.0, z: 0);
    glTranslatef( x: centerX,  y: centerY,  z: centerZ);
    glColor3d( red: c1, green: c2, blue: c3);
    glScalef( x: s, y: s, z: s);
    glutSolidSphere(radius, slices: 50, stacks: 50);
    glPopMatrix();
}}
```

Function to set the position of fixed camera on space craft

```
// Fixed camera.
gluLookAt( eyex: 0.0,  eyey: 22.0,  eyez: 10.0,  centerx: 0.0,  centery: 0.0,  centerz: 0.0,  upx: 0.0,  upy: 1.0,  upz: 0.0);
```

Function to set the position of dynamic camera on space craft

```
gluLookAt( eyex: xVal - 10 * sin( X: (M_PI / 180.0) * angle),
           eyey: 0.0,
           eyez: zVal -4 * cos( X: (M_PI / 180.0) * angle),
           centerx: xVal - 11 * sin( X: (M_PI / 180.0) * angle),
           centery: 0.0,
           centerz: zVal - 11 * cos( X: (M_PI / 180.0) * angle),
           upx: 0.0,
           upy: 1.0,
           upz: 0.0);
```

Draw the sun and handle its light and emission

```
glPushMatrix();
glEnable( cap: GL_DEPTH_TEST);
glEnable( cap: GL_COLOR_MATERIAL);
glPushMatrix();
glColor3f( red: 0.7, green: 0.5, blue: 0.0);
glTranslatef( x: 0.0, y: 0.0, z: 0.0);
glLightfv( light: GL_LIGHT7, pname: GL_POSITION, params: temp_color1);
glMaterialfv( face: GL_FRONT_AND_BACK, pname: GL_EMISSION, params: Yellow_color);
glutSolidSphere( radius: 3, slices: 50, stacks: 50);
glMaterialfv( face: GL_FRONT_AND_BACK, pname: GL_EMISSION, params: Black_color);
glPopMatrix();
```

Function to handel the lighting of planets

```
void Light_initialize()*
{
    glEnable( cap: GL_LIGHTING);
    glEnable( cap: GL_LIGHT7);
    glLightfv( light: GL_LIGHT7, pname: GL_AMBIENT, params: Ambient_color);
    glLightfv( light: GL_LIGHT7, pname: GL_DIFFUSE, params: Diffuse_color);
    glLightfv( light: GL_LIGHT7, pname: GL_SPECULAR, params: Specular_color);
}
```

Function to update the angel of each planet to rotate 360 degree around the sun

```
void change_axis(int xc){
    if (isAnimate){
        Moon_rotate+=2;
        if(Moon_rotate>360){
            Moon_rotate-=360;
        }

        Earth_rotate+=0.7;
        if(Earth_rotate>360){
            Earth_rotate-=360;
        }

        Mercury_rotate+=2;
        if(Mercury_rotate>360){
            Mercury_rotate-=360;
        }

        Venus_rotate+=0.9;
        if(Venus_rotate>360){
            Venus_rotate-=360;
        }
```

Function to handle the key input if user click space the system stop if It was running or start if it was stop

if user click ‘-‘ the speed of rotation will decrease if user click ‘+’ the speed of rotation will increase

```
/* Keyboard input processing routine.
void keyInput(unsigned char key, int x, int y)
{
    switch (key)
    {
        case 27:
            exit( Code: 0);
            break;
        case ' ':
            if (isAnimate) isAnimate = 0;
            else
            {
                isAnimate = 1;
                change_axis( xc: 1);
            }
            break;
        case '-': animationPeriod += 5;
            break;
        case '+': if (animationPeriod > 5) animationPeriod -= 5;
            break;
        default:
            break;
    }
    glutPostRedisplay();
}
```

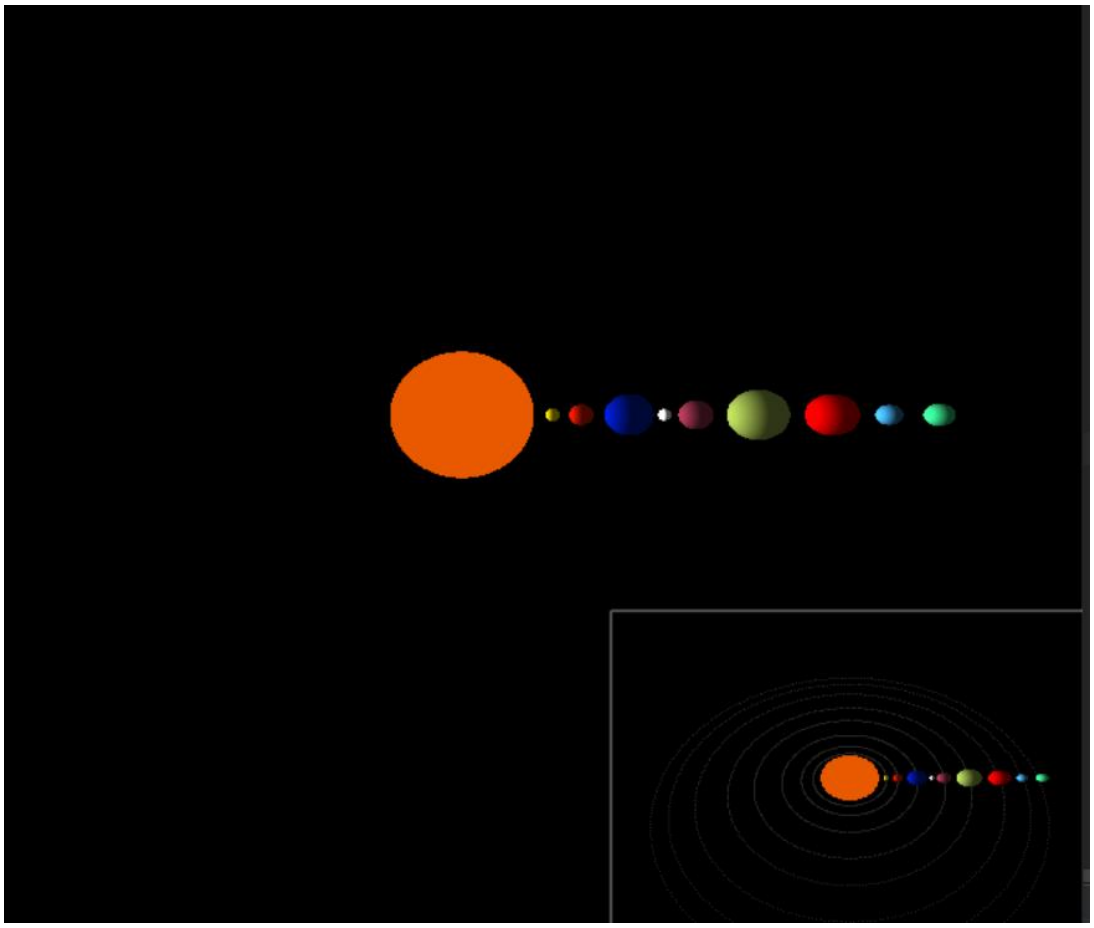
```
// Routine to output interaction instructions to the C++ window.
void printInteraction(void)
{
    std::cout << "Interaction:" << std::endl;
    std::cout << "Press the left/right arrow keys to turn the craft." << std::endl
              << "Press the up/down arrow keys to move the craft." << std::endl
              << "Press the +/- keys to increase/decrease the speed of the system." << std::endl
              << "Press the space key to run and stop the system." << std::endl;
}
```

Function to change the position of camera every time the user click on the arrows in keyboard (up, down, right, left)

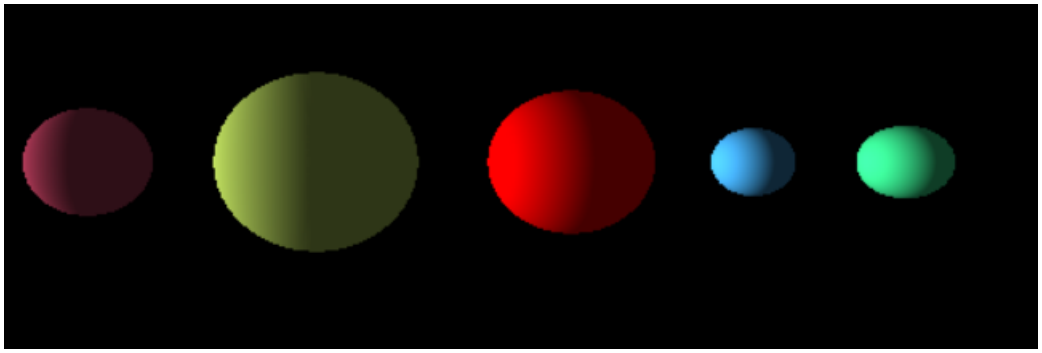
```
// Callback routine for non-ASCII key entry.
void specialKeyInput(int key, int x, int y)
{
    float tempxVal = xVal, tempzVal = zVal, tempAngle = angle;

    // Compute next position.
    if (key == GLUT_KEY_LEFT) tempAngle = angle + 5.0;
    if (key == GLUT_KEY_RIGHT) tempAngle = angle - 5.0;
    if (key == GLUT_KEY_UP)
    {
        tempxVal = xVal - sin( X: angle * M_PI / 180.0);
        tempzVal = zVal - cos( X: angle * M_PI / 180.0);
    }
    if (key == GLUT_KEY_DOWN)
    {
        tempxVal = xVal + sin( X: angle * M_PI / 180.0);
        tempzVal = zVal + cos( X: angle * M_PI / 180.0);
    }

    // Angle correction.
    if (tempAngle > 360.0) tempAngle -= 360.0;
    if (tempAngle < 0.0) tempAngle += 360.0;

    // Move spacecraft to next position only if there will not be collision with an asteroid.
    if (!asteroidCraftCollision( x: tempxVal,  z: tempzVal,  a: tempAngle))
    {
```

Results :

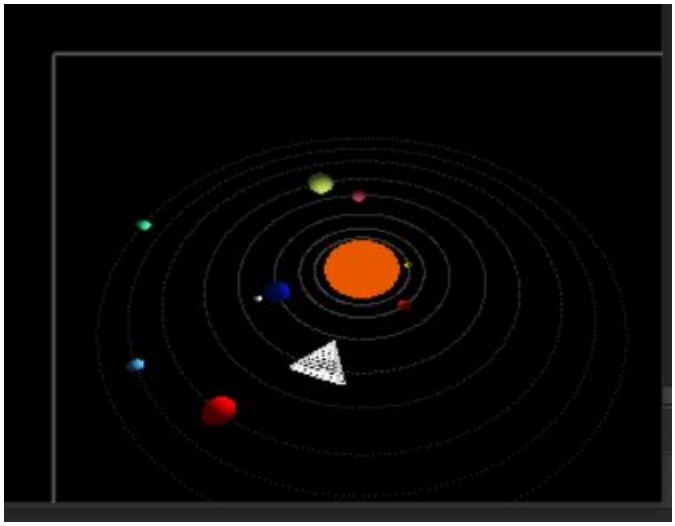
FIRST VIEW



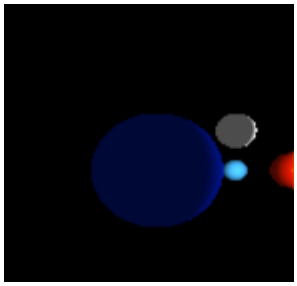
Lighting



Plan view (space craft):



The moon :



Moving the camera between planets: