Mecánica Celeste

Trabajo en grupo

Programación del cálculo, para cada planeta, de:

- **Anomalía excéntrica**
- **Energía**
- **Momento angular**
- **Posición**

1. Definición de variables

En primer lugar definimos cada variable que vamos a utilizar. Las definiremos como variables globales:

- *pos_final[3]* : vector de posición (que será calculado por el programa)

- *pos_final_derivadas[3]*: vector de derivada de posición

- *momento[3]*: vector de momento angular (calculado por el programa)

- *tiempo:* introducido por el usuario

- *energía:* será calculada por el programa

- *phi:* función vista en teoría por ʒ (utilizada para Newton-Raphson)

- *u:* anomalía excéntrica (calculada por el programa)

- *a:* semieje mayor (introducido en el programa para cada planeta)

- *periodo* (introducido en el programa para cada planeta)

- *si:* dado en teoría

- *matriz_rotacion[2][2]*

- *w*: ángulo utilizado en la matriz de rotación

- *letra, carácter, letra2*: serán utilizadas para la activación de las teclas en la visualización del sistema solar y la interacción del usuario con el programa.

```
Modelo SistemaSolar;
char letra, caracter, letra2;

double pos_final [3],pos_final_derivadas [3],momento [3], pos_rotacion[2], matriz_rotacion [2][2];
double tiempo,energia,phi,u,a,periodo,si,w;
```

2. Cálculo de funciones

- Función que calculará la **anomalía excéntrica**. Para ello programamos el método de Newton Raphson visto en clase de teoría, y denotamos por *phi* a la función ʒ.

```
/*Método para calcular u que usará el método de Newton Raphson a partir de una phi*/
void  calcularExcentricidad(double epsilon){
  si = (2*M_PI*tiempo)/periodo ;
  double u_antigua = M_PI;
  phi = (epsilon*(sin(u_antigua)-u_antigua*cos(u_antigua))+si)/(1.0-epsilon*cos(u_antigua));
  //Metodo de Newton-Raphson
  while (abs(u_antigua-phi) >= 0.00001){
    u_antigua = phi;
    phi = (epsilon*(sin(phi)-phi*cos(phi))+si)/(1.0-epsilon*cos(phi));
  }
  u = phi;
}

void Posicion(double epsilon){
```

- **Posición** que tiene cada planeta en un instante dado. Definimos una función en la cual debemos introducir el parámetro epsilon.

Para programar la posición bastará programar la fórmula correspondiente en cada coordenada pos_final[i] = i-esima coordenada del vector posición con i=1,2,3. Tenemos en cuenta también que hay una tercera coordenada en este vector (necesaria para calcular el momento angular) y esta tercera coordenada se igualará a cero. También hemos hecho el cálculo de la derivada de la posición, ya que se utilizará para el cálculo de la energía.

```
void Posicion(double epsilon){
  //Actualizamos las variables para usarlas en energia sin tener que pasar parámetros
  pos_final[0] = a*(cos(u)-epsilon);
  pos_final[1] = a*sqrt(1-(epsilon*epsilon))*sin(u) ;//sqrt(1-epsilon²) * sin(u) ;
  //declaramos una variable ayuda para hacer la derivada por cambio
  double ayuda = (2*M_PI*sqrt(pow(a,1/3))/periodo)/(sqrt(pow(a,1/3))*(1-epsilon*cos(u)));
  pos_final_derivadas[0] = (-a*sin(u)*ayuda);
  pos_final_derivadas[1] = sqrt(1.0-(epsilon*epsilon))*cos(u)*ayuda*a;
  //Completamos la 3 coordenada de ambos vectores para hacer el producto vectorial
  pos_final[2] = 0;
  pos_final_derivadas[2] = 0;
}
```

- Como ya tenemos todos los elementos necesarios para calcular la **energía**, simplemente aplicamos la fórmula E(t)= |x'(t)|/2 – mu/|x(t)|

```
/*Método para calcular la energia de un planeta en un tiempo. Varibles a y periodo actualizadas cuando calcula la posición*/
void calcularEnergia(){
  energia = ((pos_final_derivadas[0]*pos_final_derivadas[0]+pos_final_derivadas[1]*pos_final_derivadas[1])/2.0 )-(4*M_PI*M_PI*a*a*a/(periodo*periodo))
  /(sqrt(pos_final[0]*pos_final[0]+pos_final[1]*pos_final[1]));
}
```

- Para calcular el **momento angular** hacemos el producto vectorial de x(t) con x'(t) y tenemos los siguientes tres vectores:

```
/*Método que nos calcula el momento angular es decir el producto vectorial de posición y velocidad, siempre en tres coordenadas*/
void calcularMomentoAngular(){
  momento[0] = pos_final[1]*pos_final_derivadas[2]-pos_final[2]*pos_final_derivadas[1];
  momento[1] = pos_final[2]*pos_final_derivadas[0]-pos_final[0]*pos_final_derivadas[2];
  momento[2] = pos_final[0]*pos_final_derivadas[1]-pos_final[1]*pos_final_derivadas[0];
}
```

- Función que calcula, dada una anomalía excéntrica, el **tiempo** que tarda el planeta deseado.

```
void calcularTiempo( double epsilon){
    tiempo = (periodo * (u - epsilon * sin(u)))/(2*M_PI);
}
```

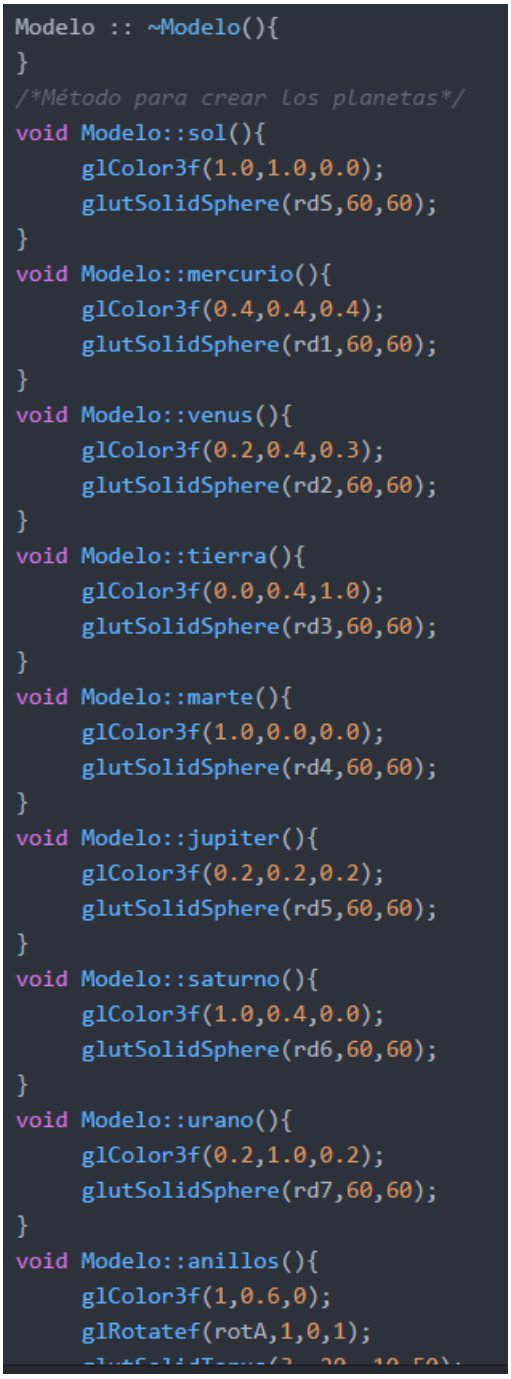
- Función que calcula la **matriz de rotación,** dado un ángulo *w*.

```
void PosicionMatriz (){
  matriz_rotacion[0][0] = cos(w);
  matriz_rotacion[0][1] = -sin(w);
  matriz_rotacion[1][0] = sin(w);
  matriz_rotacion[1][1] = cos(w);


  pos_rotacion[0] = matriz_rotacion[0][0] * pos_final[0] + matriz_rotacion[0][1] * pos_final[1] ;
  pos_rotacion[1] = matriz_rotacion[1][0] * pos_final[0] + matriz_rotacion[1][1] * pos_final[1] ;
}
```

3. Visualización del Sistema Solar

En primer lugar dibujamos los planetas, utilizando OpenGL. Dibujaremos las esferas, cada una de un color diferentes para poder distinguir los planetas de manera clara y visual. Por otro lado,



```
Modelo :: ~Modelo(){
}
/*Método para crear los planetas*/
void Modelo::sol(){
     glColor3f(1.0,1.0,0.0);
     glutSolidSphere(rdS,60,60);
}
void Modelo::mercurio(){
     glColor3f(0.4,0.4,0.4);
     glutSolidSphere(rd1,60,60);
}
void Modelo::venus(){
     glColor3f(0.2,0.4,0.3);
     glutSolidSphere(rd2,60,60);
}
void Modelo::tierra(){
     glColor3f(0.0,0.4,1.0);
     glutSolidSphere(rd3,60,60);
}
void Modelo::marte(){
     glColor3f(1.0,0.0,0.0);
     glutSolidSphere(rd4,60,60);
}
void Modelo::jupiter(){
     glColor3f(0.2,0.2,0.2);
     glutSolidSphere(rd5,60,60);
}
void Modelo::saturno(){
     glColor3f(1.0,0.4,0.0);
     glutSolidSphere(rd6,60,60);
}
void Modelo::urano(){
     glColor3f(0.2,1.0,0.2);
     glutSolidSphere(rd7,60,60);
}
void Modelo::anillos(){
     glColor3f(1,0.6,0);
     glRotatef(rotA,1,0,1);
```

A continuación, una vez construidos los planetas, procedemos al dibujado tanto de un planeta sólo, como del Sistema Solar completo (incluyendo Plutón), mediante traslaciones y rotaciones, utilizando funciones propias de OpenGl. Además, dibujaremos unas estrellas de fondo:

```
void Modelo::dibuja_Mercurio(){
  glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
  gluLookAt(1.0,1.0,1.0,0.0,0.0,0.0,0.0,1.0,0.0);

  estrellas();
  sol();

  //rotacion y traslacion para mercurio
  glPushMatrix();
  glRotatef(rot1,0,1,0);
  glTranslatef(-40,0,0);
  mercurio();
  glPopMatrix();
  glutSwapBuffers();
```

```
void Modelo::estrellas(){
    int i,j,k; srand(time(NULL));
    int h=0;
         glBegin(GL_POINTS);
         while(h<100){
         i=(rand()%300);
         j=(rand()%300);
         k=(rand()%300);
           glColor3f(1,1,1);
           glVertex3f(i,j,k);
           glVertex3f(i,-j,k);
           glColor3f(0,1,0);
           glVertex3f(-i,-j,k);
           glColor3f(0,0,1);
           glVertex3f(-i,j,-k);
           h++;
       }
    glEnd();

}
```

```
void Modelo::dibuja_Sistema(){
  glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
  gluLookAt(1.0,1.0,1.0,0.0,0.0,0.0,0.0,1.0,0.0);

  estrellas();
  sol();

  //rotacion y traslacion para mercurio
  glPushMatrix();
  glRotatef(rot1,0,1,0);
  glTranslatef(-40,0,0);
  mercurio();
  glPopMatrix();

  //rotacion y traslacion para venus
  glPushMatrix();
  glRotatef(rot2,0,1,0);
  glTranslatef(60,0,0);
  venus();
  glPopMatrix();

  //rotacion y tralacion para tierra
  glPushMatrix();
  glRotatef(rot3,0,1,0);
  glTranslatef(0,0,-80);
  tierra();
  glPopMatrix();

  //rotacion y traslacion para marte
  glPushMatrix();
  glRotatef(rot4,0,1,0);
  glTranslatef(100,0,0);
  marte();
  glPopMatrix();
```

Por último, haremos que los planetas giren en torno al Sol. Para ello, haremos 4 funciones (velocidad1, velocidad2, velocidad3, velocidad4) que harán que los planetas giren más rápido o más lento.

```
void Modelo::velocidad1(){
  rotA = rotA+4.0*0.03;
  rot1 = rot1+4.0*0.1;
  rot2 = rot2+4.0*0.015;
  rot3 = rot3+4.0*0.013;
  rot4 = rot4+4.0*0.05;
  rot5 = rot5+4.0*0.04;
  rot6 = rot6+4.0*0.03;
  rot7 = rot7+4.0*0.02;
  rot8 = rot8+4.0*0.01;
  rot9 = rot9+4.0*0.023;
}
```

Además, introduciremos una función que llame a los métodos de esta clase Modelo, para efectivamente dibujar los planetas que el usuario desee.

```
void DibujarObjetos(){

    if(toupper(letra) == 'M')
      SistemaSolar.dibuja_Mercurio() ;
    else if(toupper(letra) == 'V')
      SistemaSolar.dibuja_Venus();
    else if(toupper(letra) == 'T')
      SistemaSolar.dibuja_Tierra();
    else if(toupper(letra) == 'A')
      SistemaSolar.dibuja_Marte();
    else if(toupper(letra) == 'J')
      SistemaSolar.dibuja_Jupiter();
    else if(toupper(letra) == 'S')
      SistemaSolar.dibuja_Saturno();
    else if(toupper(letra) == 'U')
      SistemaSolar.dibuja_Urano();
    else if(toupper(letra) == 'N')
      SistemaSolar.dibuja_Neptuno();
    else if(toupper(letra) == 'P')
      SistemaSolar.dibuja_Pluton();
    else if(toupper(letra) == 'I')
      SistemaSolar.dibuja_Sistema();
}
```

4. Programa principal (main)

Después de haber realizado todos los cálculos en el programa, en último lugar vamos a pedirle al usuario qué planeta desea obtener sus resultados y, como podemos observar, en el programa introducimos un *while* que es el que hace que continuamente te esté pidiendo planetas hasta que el usuario decida finalizar el programa.

```
int main( int argc, char *argv[] ){
  while (true){
    sleep (1);

    cout << "\n\tIntroduce un planeta de la lista pulsando la letra correspondiente \n" << "\n\tMercurio --> M\n" << "\tVenus --> V\n" << "\tTierra --> T\n" << "\tMarte --> A\n" << "\tJupiter --> J\n"
    cout << "\n\tSu eleccion es: ";
    cin >> letra;

   //Comprobamos que se ha introducido una letra de la lista
    while (!(toupper(letra) == 'M' || toupper(letra) == 'V' || toupper(letra) == 'T' || toupper(letra) == 'A' || toupper(letra) == 'J' || toupper(letra) == 'J' || toupper(letra) == 'S' || toupper(letr
      cout << "Letra erronea. Vuelve a introducir una letra de la lista   ";
      cin >> letra;
    }

    cout << "\n\tIntroduce \n\tA --> si quieres calcular la anomalia excentrica a partir del tiempo \n\tT --> si quieres calcular el tiempo a partir de la anomalia excentrica \n";
    cout << "\n\tSu eleccion es: ";
    cin >> letra2;

    while (!(toupper(letra2) == 'A' ||  toupper(letra2) == 'T')){
      cout << "Letra erronea. Vuelve a introducir una letra de la lista   ";
      cin >> letra2;
    }
```

Ahora le pedimos el tiempo en días del planeta y según el planeta pedido por el usuario tenemos unos ciertos valores del semieje mayor, la excentricidad y el periodo de cada planeta.

```
if (toupper(letra2)=='A'){
   cout << "\tIntroduce el tiempo en días: ";
   cin >> tiempo;

  if(toupper(letra) == 'M'){
    periodo = 87.97;
    a = 0.387;
    w = 75.9 ;
    cout << "\n\t ___Para Mercurio___ : \n";
    calcularExcentricidad(0.206);
    Posicion(0.206);
  }
  else if(toupper(letra) == 'V'){
    periodo = 224.7;
    a = 0.723;
    w = 130.15;
    cout << "\n\t ___Para Venus___ : \n";
    calcularExcentricidad(0.007);
    Posicion(0.007);
  }
  else if (toupper(letra) == 'T'){
    periodo = 365.26;
    a = 1.00;
    w = 101.22;
    cout << "\n\t ___Para La Tierra___ : \n";
    calcularExcentricidad(0.017);
    Posicion(0.017);
  }
  else if(toupper(letra) == 'A'){
    periodo = 686.98;
    a = 1.524;
    w = 334.22;
    cout << "\n\t ___Para Marte___ : \n";
    calcularExcentricidad(0.093);
    Posicion(0.093);
  }
```

Del mismo modo, podemos pedirle que introduzca la anomalía excéntrica, y nos calculará el tiempo, la posición, etc:

```
} else if (toupper(letra2)=='T'){

    cout << "\tIntroduce la anomalia excentrica: ";
    cin >> u;

    if(toupper(letra) == 'M'){
      periodo = 87.97;
      a = 0.387;
      w = 75.9;
      cout << "\n\t ___Para Mercurio___ : \n";
      calcularTiempo(0.206);
      Posicion(0.206);
    }
    else if(toupper(letra) == 'V'){
      periodo = 224.7;
      a = 0.723;
      w = 130.5;
      cout << "\n\t ___Para Venus___ : \n";
      calcularTiempo(0.007);
      Posicion(0.007);
    }
    else if (toupper(letra) == 'T'){
      periodo = 365.26;
      a = 1.00;
      w = 101.22;
      cout << "\n\t ___Para La Tierra___ : \n";
      calcularTiempo(0.017);
      Posicion(0.017);
    }
    else if(toupper(letra) == 'A'){
      periodo = 686.98;
      a = 1.524;
      w = 334.22;
      cout << "\n\t ___Para Marte___ : \n";
      calcularTiempo(0.093);
      Posicion(0.093);
    }
    else if (toupper(letra) == 'J'){
      periodo = 4332.6;
```

Los resultados se mostrarán mediante salidas en el terminal que indicarán lo que vale cada función calculada anteriormente:

```
calcularMomentoAngular();
calcularEnergia();
PosicionMatriz();

cout << "\n\tEl tiempo es : " << tiempo;
cout << "\n\tLa posicion es: x("<<tiempo << ") = ["<< pos_final[0] <<" , "<< pos_final[1] << " ]" ;
cout << "\n\tLa energia es : " << energia ;
cout << "\n\tEl momento angular es c("<<tiempo<<"):[ " << momento[0]<<" , " << momento[1]<<" , "<< momento[2]<<" ]";
cout << "\n\tLa posicion con rotación es : x("<<tiempo << ") = ["<< pos_rotacion[0] <<" , "<< pos_rotacion[1] << " ]" ;
```

```
calcularMomentoAngular();
calcularEnergia();
PosicionMatriz();

cout << "\n\tLa posicion es: x("<<tiempo << ") = ["<< pos_final[0] <<" , "<< pos_final[1] << " ]" ;
cout << "\n\tLa energia es : " << energia ;
cout << "\n\tEl momento angular es c("<<tiempo<<"):[ " << momento[0]<<" , " << momento[1]<<" , "<< momento[2]<<" ]";
cout << "\n\tLa anomalia excentrica es : " << u;
cout << "\n\tLa posicion con rotación es : x("<<tiempo << ") = ["<< pos_rotacion[0] <<" , "<< pos_rotacion[1] << " ]" ;
```

5. Teclas utilizadas para la interacción con el usuario

m/M: seleccionar el planeta Mercurio

v/V: seleccionar el planeta Venus

t/T: seleccionar el planeta Tierra

a/A: seleccionar el planeta Marte

j/J: seleccionar el planeta Júpiter

s/S: seleccionar el planeta Saturno

u/U: seleccionar el planeta Urano

n/N: seleccionar el planeta Neptuno

p/P: seleccionar el planeta Plutón

1,2,3,4: para cambiar la velocidad de giro de los planetas.

T/t: para indicar si vamos a introducir un tiempo

a/A: para indicar que introducimos una anomalía excéntrica.

Q/q: para salir del programa una vez mostrado el planeta(s)

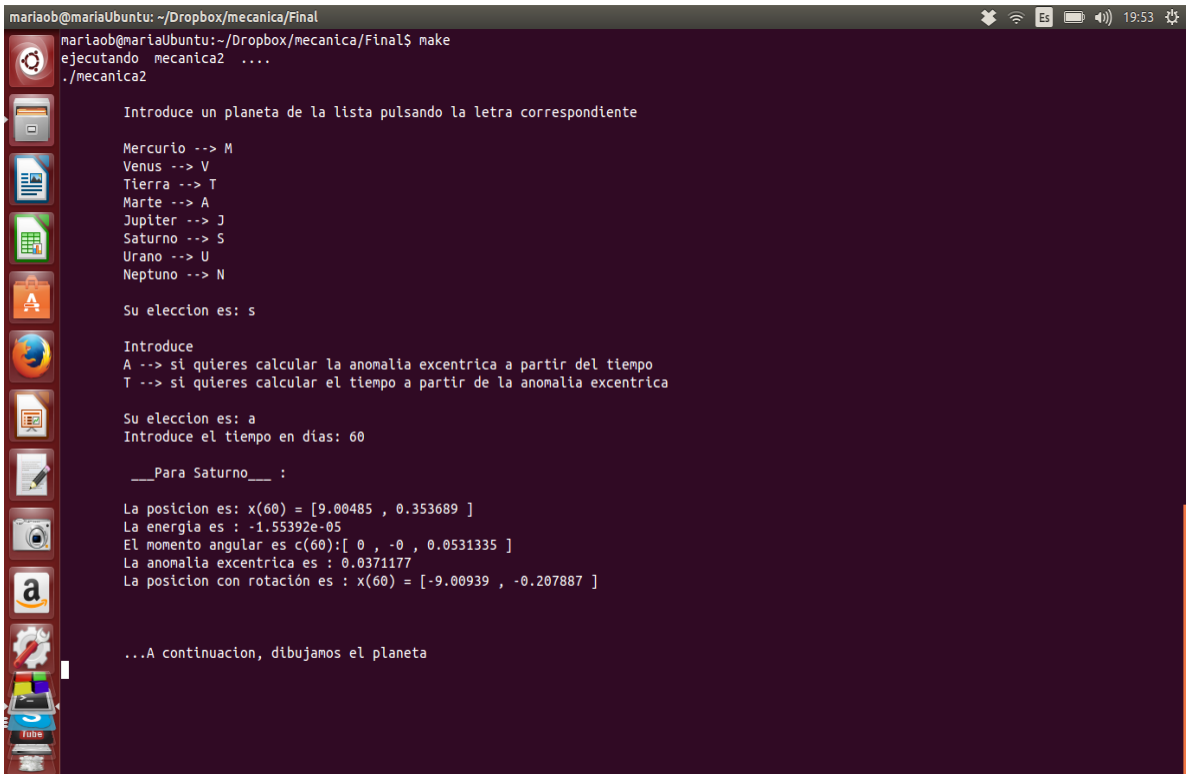
c/C: para cerrar la ventana de dibujado y seguir ejecutando el programa

+/-: alejar o acercar el dibujo

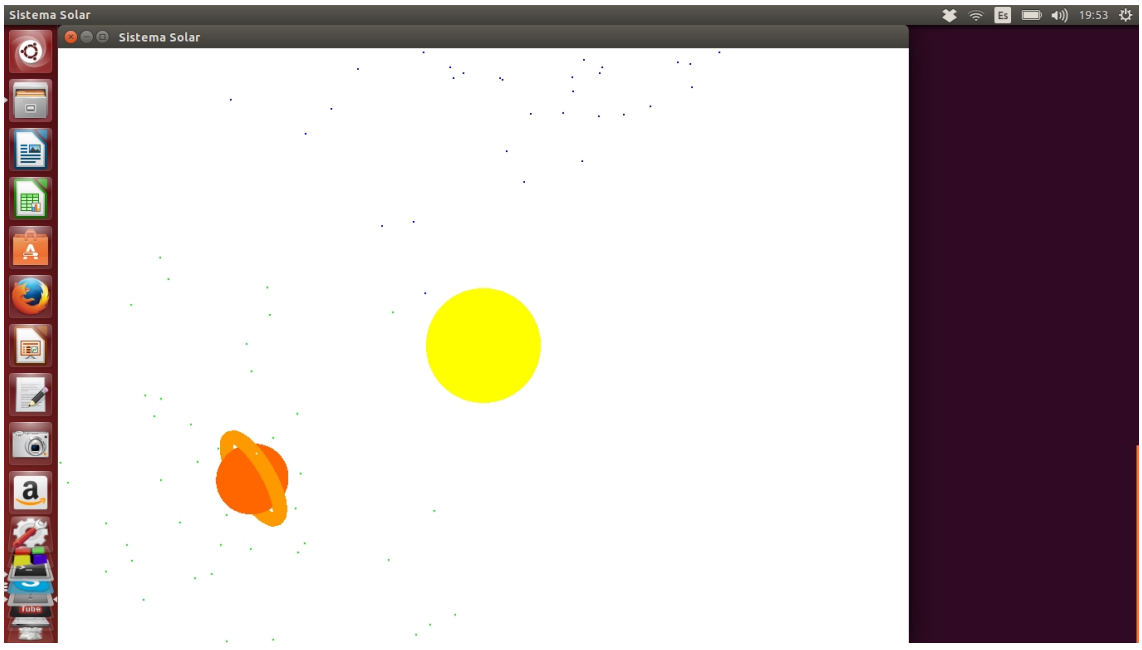
flechas: para cambiar los ejes

6. Ejemplo de uso

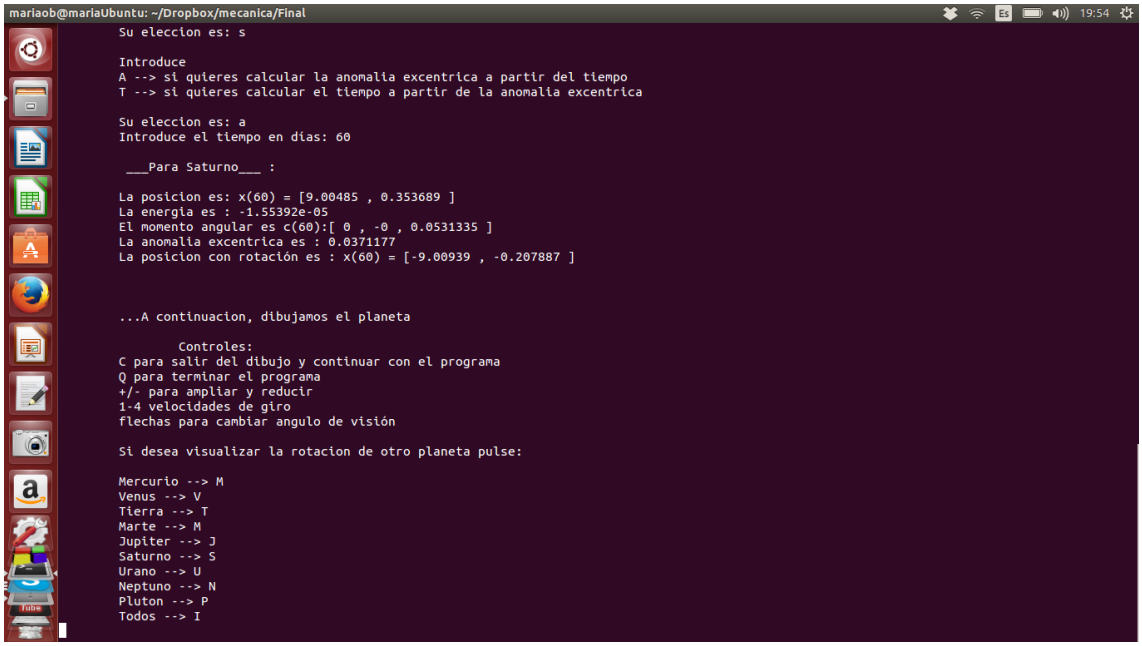
Tras ejecutar “make” en el terminal, se lanzará el programa y podremos ver como funciona:

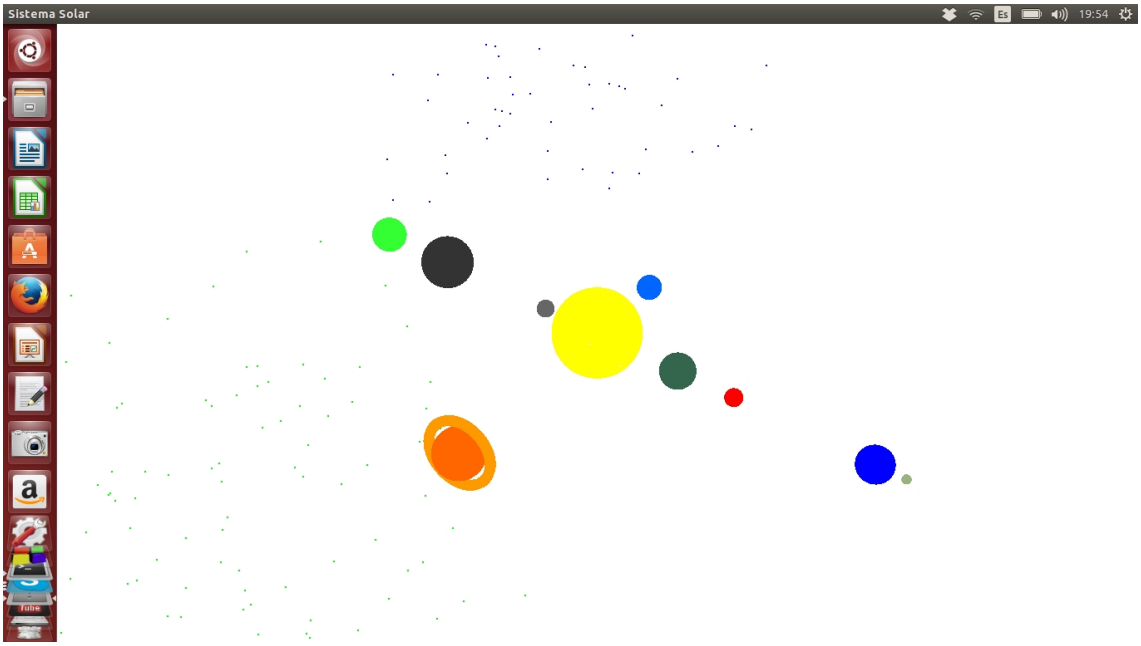


Tras mostrarnos los cálculos (en este caso hemos introducido una anomalía, pero como ya hemos comentado, podemos introducir el tiempo), como bien se indica en el terminal, se dibujará el planeta que hemos seleccionado, y podremos ver su movimiento pulsando 1,2,3 ó 4 para más o menos velocidad.



En el terminal se nos mostrarán las diferentes opciones que tenemos una vez abierta la ventana de dibujado: cambiar el planeta a visualizar; mostrar todo el sistema solar; alejar o acercar el dibujo; girar el planeta; cerrar la ventana; o terminar la ejecución del programa.





En el caso de cerrar la ventana (tecla c/C), el programa seguirá ejecutándose desde el terminal, para un nuevo planeta que introduzca el usuario.

TRABAJO REALIZADO POR:

- Lorena Álvarez Plana
- María Teresa Calvo Muñoz
- María Oliver Balsalobre
- Miguel Sánchez Maldonado