Fisica Practica 1

Christian Graf, Ismael Arroyo

POLITÉCNICA

11.04.2019

Implementar el problema del movimiento de un electrón en un en presencia de un campo eléctrico uniforme en el que penetra con un ángulo inicial α y una velocidad inicial v_0

El programa debe:

1. Calcular y mostrar por pantalla para cada pareja de valores de velocidad y ángulo iniciales:
   - Alcance máximo (x_máxima) de la partícula y tiempo transcurrido (t_máximo)
   - Altura máxima (y_máxima), respecto de la posición inicial; tiempo que tarda en alcanzarla (t_de_y_máx) y coordenada x correspondiente (x_de_y_máx).
2. Representar gráficamente la trayectoria de la partícula en un intervalo de tiempo adecuado.

Constantes:

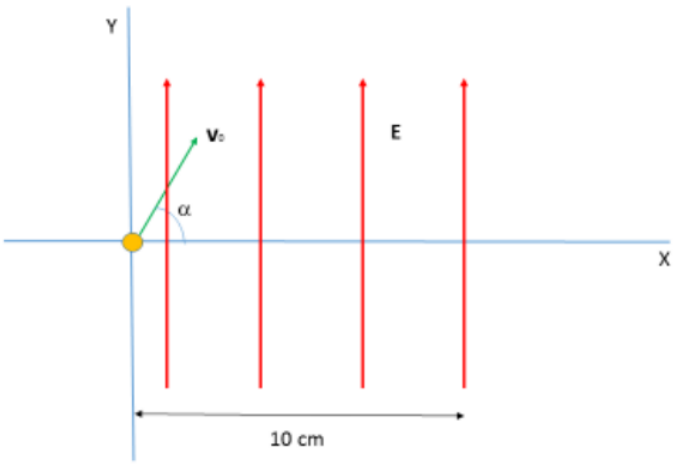
$$\vec{\mathbf{E}} = 3.50 * 10^3 N/C$$

$$m_e = 9.11 * 10^{-31} kg$$

$$e = 1.6 * 10^{-19} C$$

Los inputs:

$$\alpha, v_0$$



Solución:

El programa se divide principalmente en dos partes:

- La clase Electron.py
- La clase main.py

Electron.py

La clase Electron.py contiene la logica del programa. Aqui reciden las constantes del electron ademas de las variables que se dan como input. Hemos usado las siguientes funciones para hacer los calculos:

- Primero inicializamos los valores con **init**

```
def __init__(self,v0,alpha):
        self.x_array = []
        self.y_array = []
        self.angle = alpha
        self.v = v0
        self.v0 = v0
        self.e = -1.6*math.pow(10,-19)
        self.m = 9.11*math.pow(10,-31)
        self.Ex, self.Ey = (0, 3.5*1000)
        self.x,self.y = (0,0)
        self.v0x, self.v0y = self.calculatev0_x_y(v0,alpha)
        self.Fx, self.Fy = (self.e*self.Ex, self.e*self.Ey)
        self.ax, self.ay = (self.Fx/self.m, self.Fy/self.m)
        (self.maxx,self.maxy,self.maxyx,self.time) = self.calculateMax
            ()
```

- Con **calculatev0_x_y** calculamos la velocidad inicial en funcion del angulo

```
    def calculatev0_x_y(self,v0,alpha):
        return ( v0*math.cos(math.radians(alpha)),
                 v0*math.sin(math.radians(alpha)))
```

- Con **calculateVelocity** calculamos la velocidad en un momento del tiempo.

```
 def calculateVelocity(self,a,t,v0):
        return ( (a*t) + v0)
```

- Con **calculateTime** calculamos el tiempo que ha transcurrido

```
def calculateTime(self,v,v0,a):
        return ((v-v0)/a)
```

- Con **calculateLength** calculamos la distancia que se ha movido desde el origen.

```
  def calculateLength(self,v0,a,t):
        return ((0.5*a*math.pow(t,2))+(v0*t))
```

- Finalmente con **calculateMax** encontramos los valores maximos de x e y

```
def calculateMax(self):
        highest_x = -999999.000
        curr_x = 0.000
        highest_y = -999999.000
        highest_y_xcoord = 0
        curr_y = 0.000
        time = -1
        found = False
        for t in np.arange (0,1000,0.01):
            if (curr_x >= 0.1):
                self.Ey=0
                self.Ex=0
            curr_x = self.calculateLength(self.v0x,self.ax,t*math.pow
                (10,-9))
            self.x_array.append(curr_x)
            curr_y = self.calculateLength(self.v0y,self.ay,t*math.pow
                (10,-9))
            self.y_array.append(curr_y)
            if curr_x > highest_x:
                highest_x = curr_x
            else:
                print("found max x")
                found = True
                time = t*math.pow(10,-9)
                break
            if curr_y > highest_y:
                highest_y = curr_y
            elif found == False:
                found = True
                time = t*math.pow(10,-9)
```

```
                highest_y_xcoord=curr_x
                print('found max y')
        return (highest_x,highest_y,highest_y_xcoord,time)
```

Main.py

El main se encarga principalmente de llamar a Electron.py y pintar la grafica. Hemos usado la libreria *matplotlib* para pintarla.

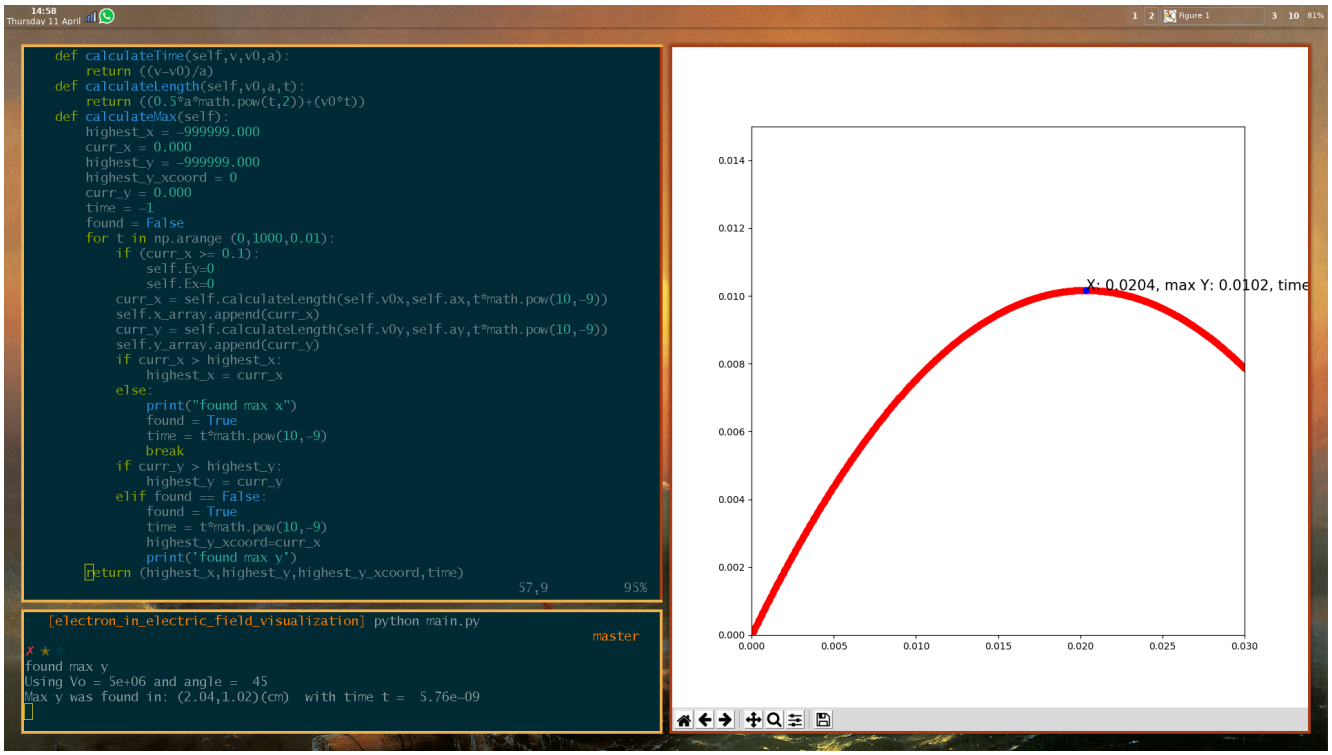
```
import math
import matplotlib.pyplot as plt
from Electron import *

Vo = 5*math.pow(10,6)
degrees = 45
electron = Electron(Vo,degrees)
print("Using Vo = {:.2} and angle = ".format(Vo),degrees)
print("Max y was found in: ({:.2f},{:.2f})(cm)".format(electron.maxyx*
    math.pow(10,2),electron.maxy*math.pow(10,2)))

plt.plot(electron.x_array, electron.y_array, 'ro')
plt.axis([0,0.03,0,0.015])
plt.plot(electron.maxyx,electron.maxy,'bo')
formatList = ["{:.4f}".format(electron.maxyx),"{:.4f}".format(electron.
    maxy),"{:.4f}".format(electron.time*math.pow(10,6))]
text = "X: {}, max Y: {}, time: {}*10^-6".format(*formatList)
plt.text(electron.maxyx,electron.maxy,text,fontsize = 15)
plt.show()
```

Ejemplo de ejecucion 1

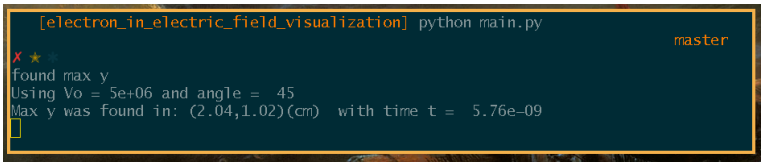
Como se puede ver aqui, con los valores de entrada dados de ejemplo:

$$\alpha = 45°, v_0 = 5 * 10^6 m/s$$

obetenemos el movimiento del electron en el tiempo:



El programa tambien nos da output por el terminal:



Resolución de x_max:

Para calcular x_{max} seguimos el mismo procedimiento que en y_{max} :

$$x = x_0 + v_0 t + \frac{1}{2} a t^2$$

$$v = v_0 + at$$

a su vez, tenemos en cuenta la segunda ley de Newton

$$F = ma \, , F = qE \rightarrow a = \frac{mE}{q}$$

Luego para calcular la aceleracion en el eje x tenemos:

$$\vec{\mathbf{a_x}} = \frac{m\vec{\mathbf{E_x}}}{q}, \vec{\mathbf{E_x}} = 0$$

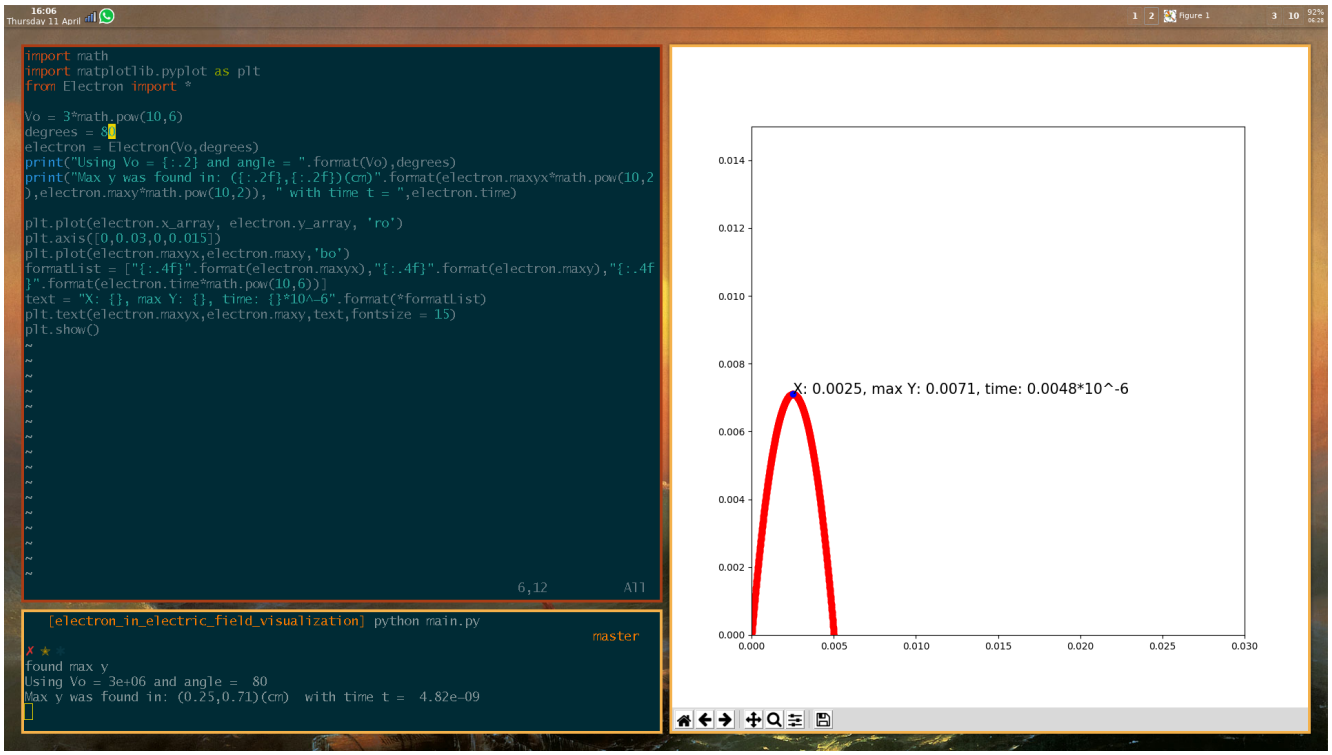
$$\rightarrow \vec{\mathbf{a_x}} = 0 \rightarrow v_x = v_0 + 0t \rightarrow v_x = v_0$$

Por lo tanto no existira nunca un x_{max} puesto que no hay aceleracion negativa en el eje x

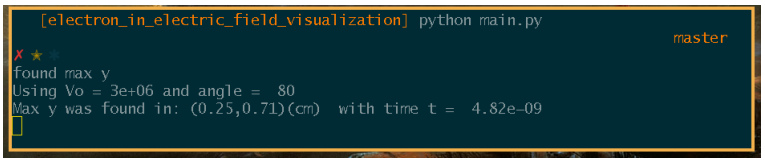
Ejemplo de ejecucion 2:

$$\alpha = 80°, v_0 = 3 * 10^6 m/s$$

obetenemos el movimiento del electron en el tiempo:



El programa tambien nos da output por el terminal:



nos encontramos con la misma situacion de antes con xmax