



UNIVERSIDAD NACIONAL AUTONOMA DE MEXICO  
FACULTAD DE INGENIERIA

LABORATORIO DE COMPUTACION GRAFICA  
EFECTO COMPTON

MONARES ZABALETA CARLOS ALBERTO

*GRUPO: 4*

## OBJETIVOS:

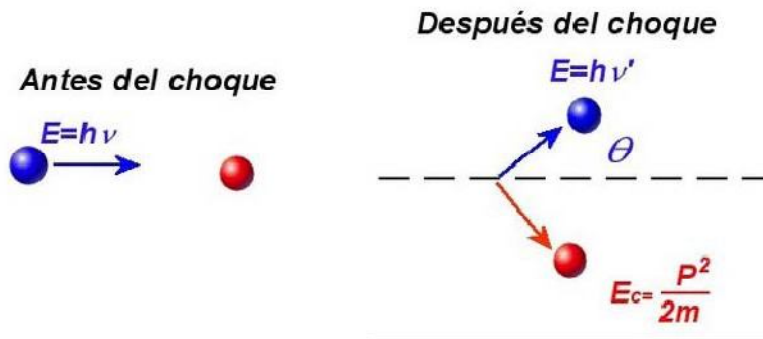
- **Objetivo1:** Acoplar ecuaciones con visualización con opengl
- **Objetivo2:** Que el alumno relacione la parte teoría con la parte visual, habiendo adquirido las bases requeridas para correr y modificar programas "opengl".

## INTRODUCCION

### ¿Qué es Efecto Compton?

Es la desviación en movimiento del golpe de un fotón contra un electrón, donde ambos salen disparados en diferentes direcciones.

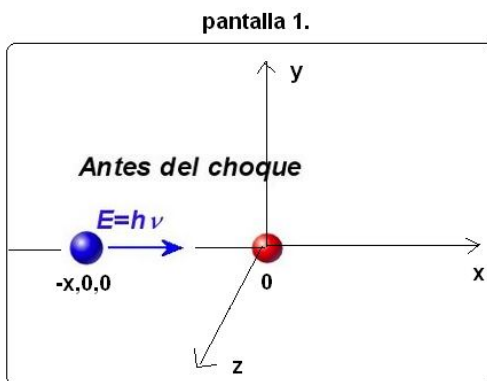
### ¿Cómo funciona?



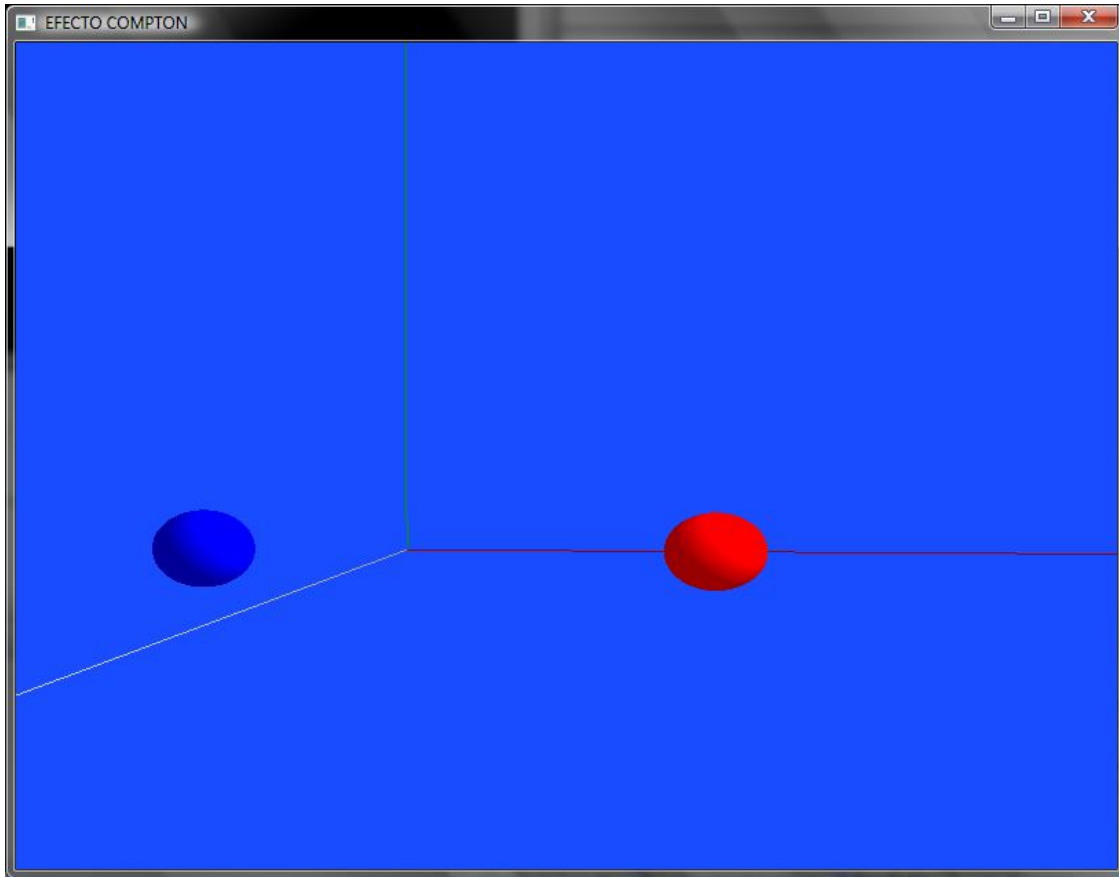
## DESARROLLO

### Parte 1

En la parte inicial, se tienen las dos pelotitas que representan al electrón y al fotón:



Captura de pantalla:



## Parte 2

Las esferas comenzaran a moverse de acuerdo con las ecuaciones siguientes:

a) Ecuación del fotón inicial:

$$E = h\omega, \quad \omega = 2 * \pi * f \quad f = \frac{1}{\lambda}$$

Donde:

$$h = 6.63 \times 10^{-34}$$

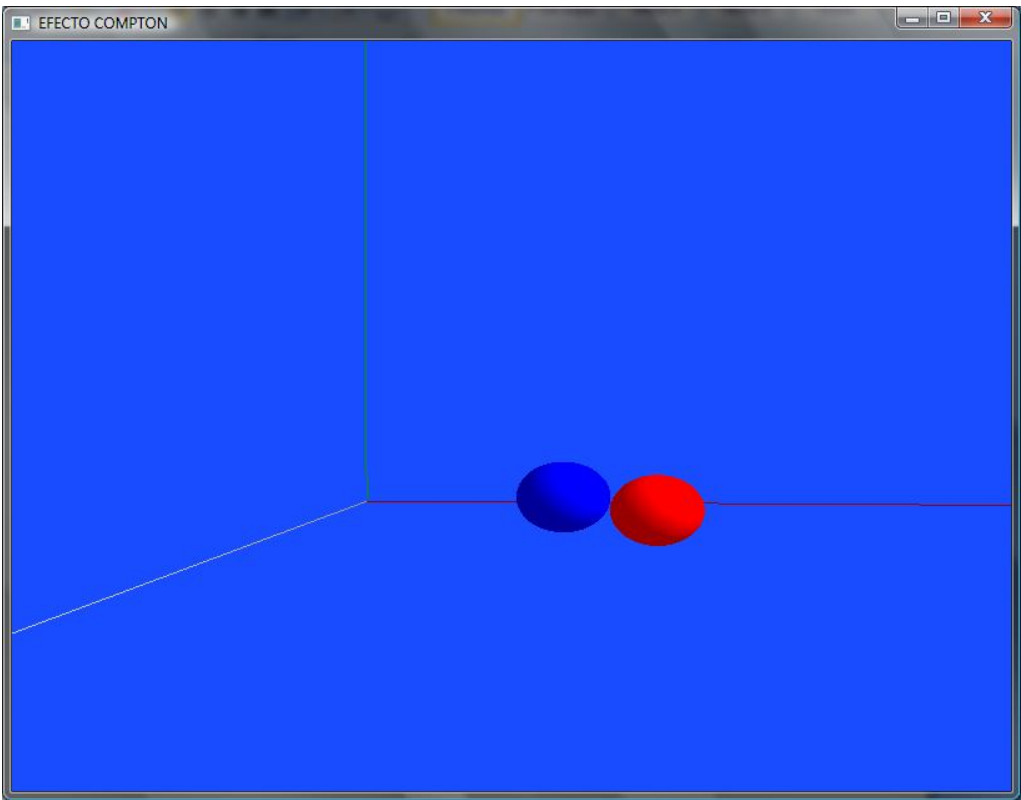
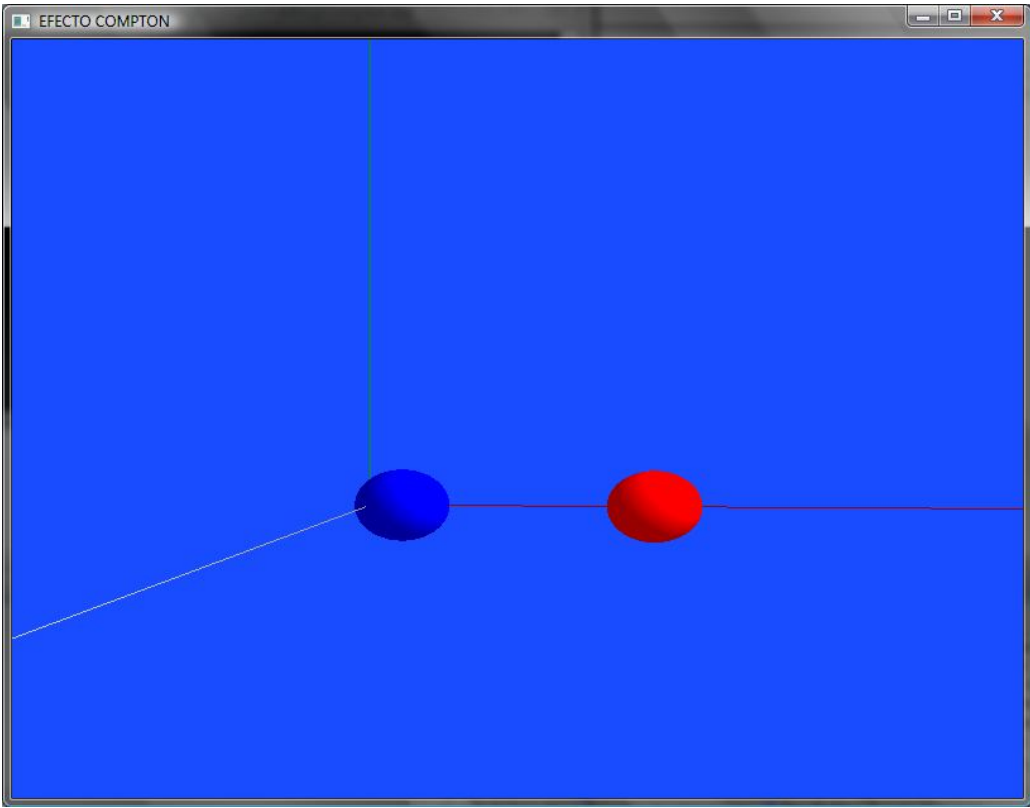
La energía obtenida se representara visualmente con la velocidad de la partícula, si tiene una longitud grande, tenemos una frecuencia pequeña, tendremos una E1, pequeña también, veremos que la pelota va lento para golpear al electrón, si es al contrario ira muy rápido.

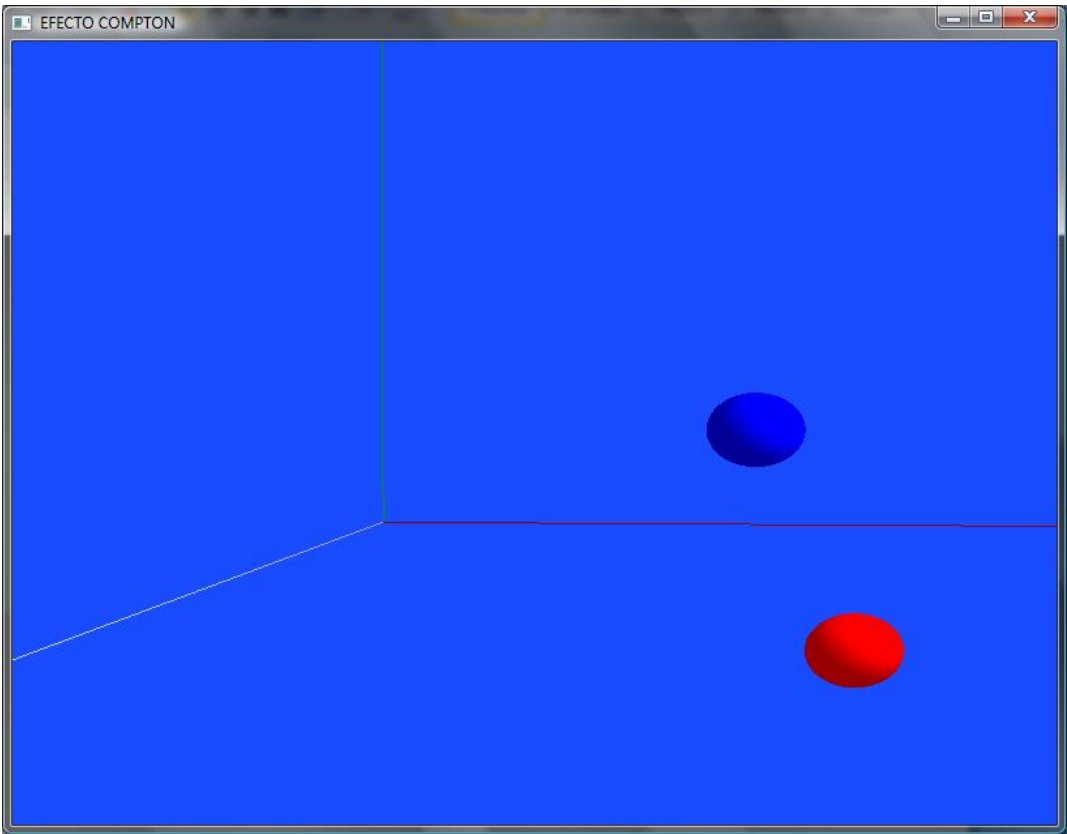
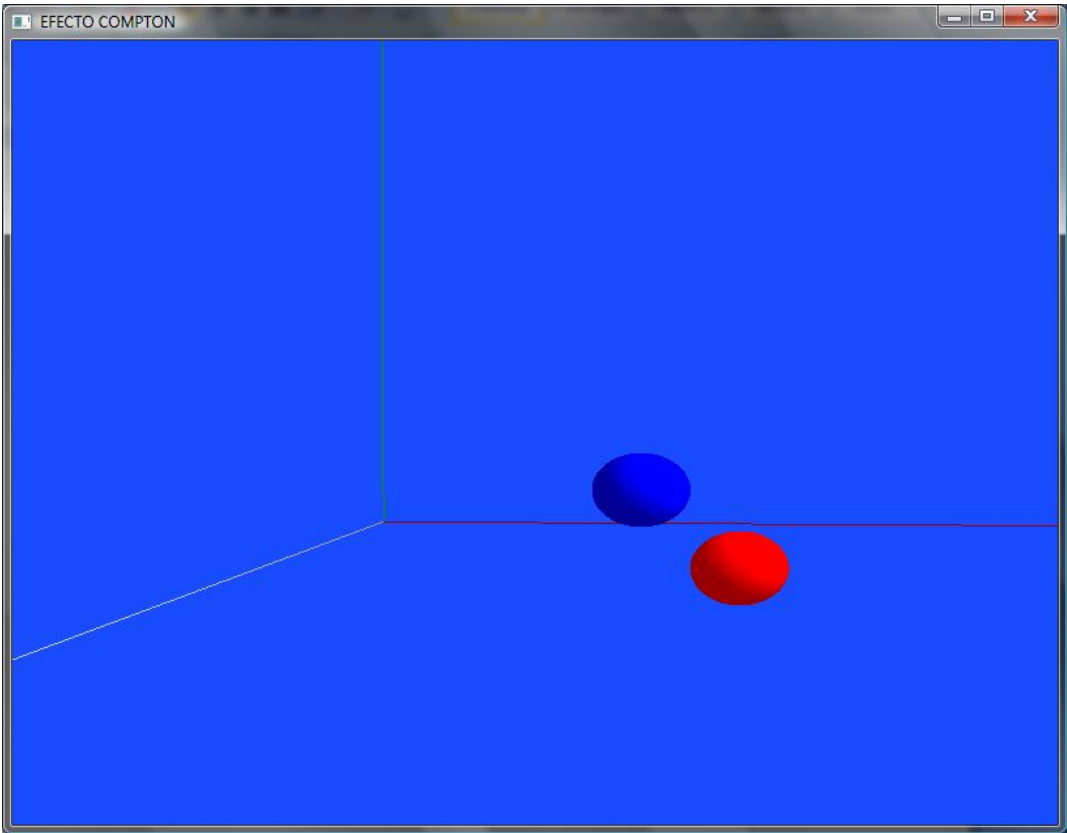
**NOTA 1:** Para comenzar se muestra una imagen del archivo de texto empleado, nótese que el programa está preparado para reconocer que el primer valor representa la longitud de onda y, por tanto, considera que dicho valor es del orden de  $10^{-10}$  m, el siguiente valor corresponde al ángulo del fotón y el último al ángulo del electrón:

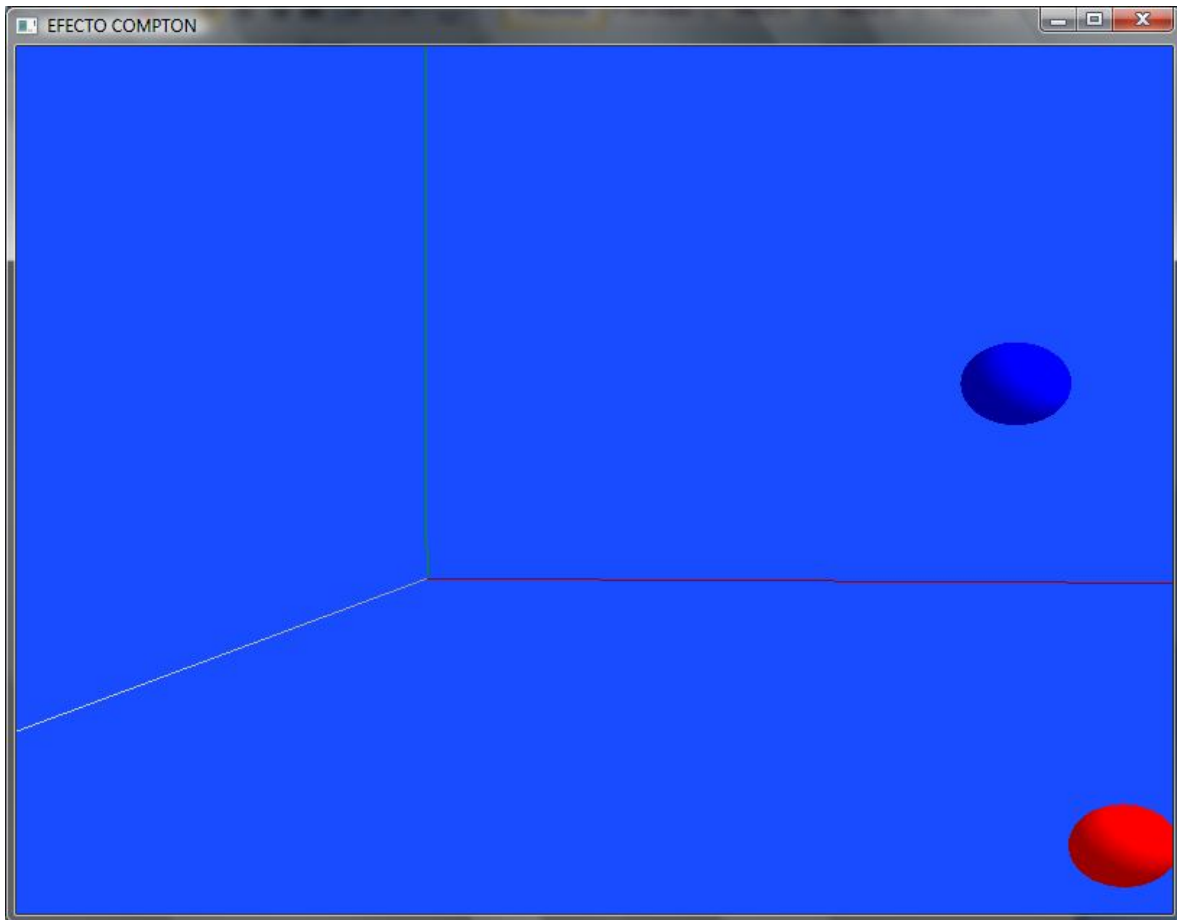


**NOTA 2:** El valor obtenido de la energía del fotón fue multiplicado por una constante para que la velocidad del mismo pudiera ser apreciada, ya que de lo contrario, la velocidad real a la que deberían moverse las esferas sería tan alta que sería imposible observarlas.

A continuación, se muestran algunas capturas de pantalla del programa en funcionamiento:







### CODIGO FUENTE:

```
//MONARES ZABALETA CARLOS ALBERTO
//LABORATORIO DE COMPUTACION GRAFICA
#include "openglut.h"
#include <math.h>
#include <fstream>
#include <iostream>
#include <cstdlib>

GLUquadricObj *quadobj;
//Definición del modelo de una luz
GLfloat light_Ambient [4] = { 0.4, 0.4, 0.4, 1.0};
GLfloat light_Diffuse [4] = { 0.7, 0.7, 0.7, 1.0};
GLfloat light_Position [4] = {20.0, 15.0, 10.0, 1.0};

//Definición de las características ópticas del material: coeficientes de reflexión
GLfloat material [4] = {1.0, 0.2, 0.2, 1.0 };
GLfloat RedMaterial [4] = {1.0, 0.0, 0.0, 1.0 };
GLfloat GreenMaterial [4] = {0.0, 1.0, 0.0, 1.0 };
GLfloat BlueMaterial [4] = {0.0, 0.0, 1.0, 1.0 };
GLfloat WhiteMaterial [4] = {1.0, 1.0, 1.0, 1.0 };
```

```

float angle;
float v;
char vertices[30];
float puntos[30];
int lon;
float teta;
float fi;

void luces(void)
{
//Cargando las ecuaciones de luz
    glEnable (GL_LIGHTING);
    glEnable (GL_LIGHT0);

    glLightfv(GL_LIGHT0, GL_AMBIENT, light_Ambient );
    glLightfv(GL_LIGHT0, GL_DIFFUSE, light_Diffuse );
    glLightfv(GL_LIGHT0, GL_POSITION, light_Position );
}

void EjesReferencia()
{
    glNewList(1, GL_COMPILE);

    glBegin (GL_LINES);
        glMaterialfv(GL_FRONT, GL_AMBIENT_AND_DIFFUSE, RedMaterial );
//eje X
        glVertex3f ( 0.0, 0.0, 0.0);
        glVertex3f ( 20.0, 0.0, 0.0);

        glMaterialfv(GL_FRONT, GL_AMBIENT_AND_DIFFUSE, GreenMaterial );
//Eje Y
        glVertex3f ( 0.0, 0.0, 0.0);
        glVertex3f ( 0.0, 20.0, 0.0);

        glMaterialfv(GL_FRONT, GL_AMBIENT_AND_DIFFUSE, WhiteMaterial );
//Eje Z
        glVertex3f ( 0.0, 0.0, 0.0);
        glVertex3f ( 0.0, 0.0, 20.0);
    glEnd();

    glEndList();
}

void LeeArchivo(void){
    lon=0 ;
    for(int h=0; h<30; h++)
        {
            vertices[h] = ' ';
            puntos[h] = 0.0;
        }
    std::ifstream myfile("muestra2.txt");
    while(!myfile.eof())
        {
            myfile >> vertices;
        }
}

```

```

        puntos[lon] = atof(vertices);
        lon++;
    }
    myfile.close();
}

void Electron(){
    glNewList(2, GL_COMPILE);
    glMaterialfv (GL_FRONT, GL_AMBIENT_AND_DIFFUSE, RedMaterial);
    glPolygonMode (GL_FRONT_AND_BACK, GL_FILL);
    glShadeModel (GLU_SMOOTH);
    quadobj = gluNewQuadric();
    gluQuadricDrawStyle( quadobj, GLU_FILL); //estilo lleno, alambrado,
de puntos o silueta.
    gluQuadricNormals( quadobj, GLU_SMOOTH); // tipo de normal una por
cara (flat)
    gluSphere(quadobj, 0.5, 300, 300);
    gluDeleteQuadric(quadobj);
    glEndList();
}

void Foton(){
    glNewList(3, GL_COMPILE);
    glMaterialfv (GL_FRONT, GL_AMBIENT_AND_DIFFUSE, BlueMaterial);
    glPolygonMode (GL_FRONT_AND_BACK, GL_FILL);
    glShadeModel (GLU_SMOOTH);
    quadobj = gluNewQuadric();
    gluQuadricDrawStyle( quadobj, GLU_FILL); //estilo lleno, alambrado,
de puntos o silueta.
    gluQuadricNormals( quadobj, GLU_SMOOTH); // tipo de normal una por
cara (flat)
    gluSphere(quadobj, 0.5, 300, 300);
    gluDeleteQuadric(quadobj);
    glEndList();
}

void ecuaciones(void){
//Para la energia se tiene  $E = hw$ 
//Para  $w = 2 * \pi * f$ 
//Para  $f = 1/\lambda$ 
//l se obtiene del usuario de un archivo de texto
//h es una constante,  $h = 6.63 \times 10^{-34}$ 
//Simplificando ecuaciones tenemos  $E = 2*\pi*h/l$ 
//Y sustituyendo valores tenemos  $E = 4.1657 \times 10^{-33} / \lambda$ 
float lambda;
float e;
lambda = puntos[0];
e = (4.1657 * pow(10.0, -33)) / (lambda * pow(10.0, -10));
//Como el resultado sigue siendo muy pequeno, se multiplica el valor
//de la energia por  $10^{19}$  para obtener la velocidad
v = e * pow(10.0, 20);
fi = (puntos[2]) * (3.1416 / 180);
teta = (puntos[1]) * (3.1416 / 180);
}

void idle(void)
{

```

```

angle = angle+(float)v;
glutPostRedisplay();
if (angle>=8) angle=-2;
}

void RenderScene(void)
{
    /*Borra frame buffer*/
    glClear(GL_COLOR_BUFFER_BIT|GL_DEPTH_BUFFER_BIT);

    /*Pasa listas por tuberia Opendgl*/
    glCallList(1);          //Ejes de referencia fijos

    glMatrixMode(GL_MODELVIEW);
    glPushMatrix();
    if(angle<=2.0)
    {
        glTranslatef(angle,0,0);
    }else{glTranslatef(angle,angle*sin(teta)-2.0*sin(teta),0);}
    glCallList(3);
    glPopMatrix();

    glPushMatrix();
    if(angle<=2.0)
    {
        glTranslatef(3,0,0);
    }else{glTranslatef(angle+1.0,-angle*sin(fi)+2.0*sin(fi),0);}
    glCallList(2);
    glPopMatrix();

    /*Renderea en bufer de color posterior (oculto)*/
    glFlush();

    /*Envia el bufer de color posterior al frente*/
    glutSwapBuffers();
}

int main(void)
{
    int IdeWindow;

    glutInitDisplayMode(GLUT_DOUBLE | GLUT_RGB | GLUT_DEPTH);
    glutInitWindowSize(800,600);
    glutInitWindowPosition(300,100);
    IdeWindow=glutCreateWindow("EFECTO COMPTON");
    glutDisplayFunc(RenderScene);
    glutIdleFunc( idle );

    glClearColor(0.1f, 0.3f, 1.5f, 1.0f);

    //Habilitando el bufer de profundidad
    glEnable(GL_DEPTH_TEST);

    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    gluPerspective(15,800/600,.1,1000); //explicacion abajo

```

```

    glMatrixMode(GL_MODELVIEW);
    glLoadIdentity();
    gluLookAt(6,3,40, 1,1,-5, 0,1,0);

    /*Prepara la luz*/
    luces();

    /*Prepara ejes de referencia*/
    EjesReferencia();

    LeeArchivo();
    Electron();
    Foton();
    ecuaciones();

    angle=-2;

    glutMainLoop();

    //Destruir la ventana y el contexto GL
    glutDestroyWindow(IdeWindow);

    return 0;
}

```

## CONCLUSIONES

Este proyecto realmente es un compendio de todo lo aprendido hasta ahora, ya que combina elementos de todas las practicas anteriores, en cuanto a la velocidad del electron se pudo constatar que esta varia de acuerdo a como se varia la longitud de onda.