



UNIVERSIDAD NACIONAL AUTONOMA DE MEXICO
FACULTAD DE INGENIERIA

LABORATORIO DE COMPUTACION GRAFICA
EFECTO COMPTON COMPLETO

MONARES ZABALETA CARLOS ALBERTO

GRUPO: 4

OBJETIVOS:

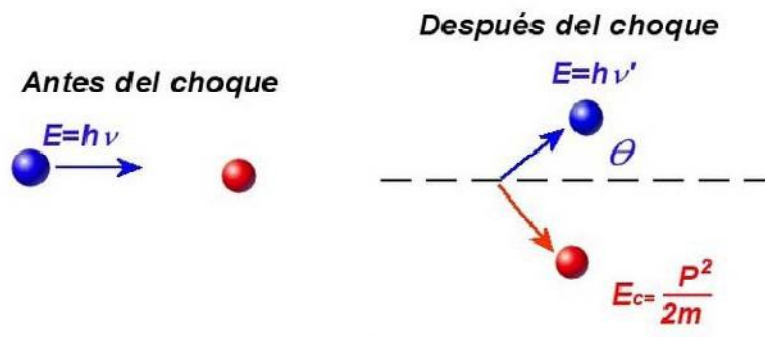
- Simular el efecto Compton completo empleando las ecuaciones necesarias
- Simular el movimiento de los electrones girando alrededor del núcleo de una molécula

INTRODUCCION

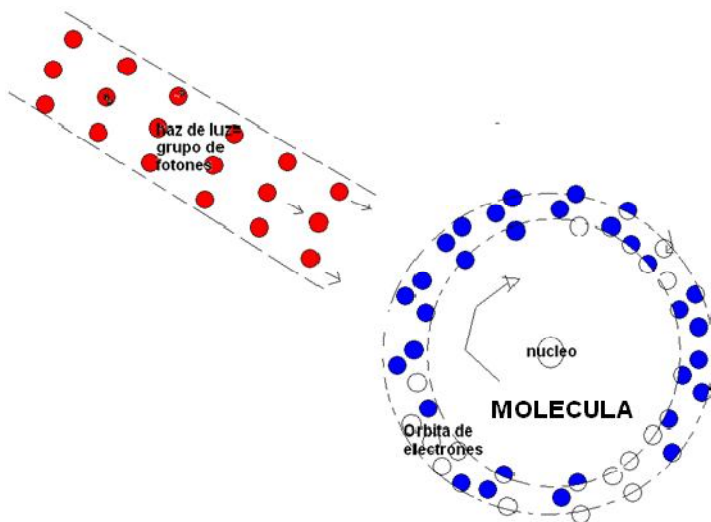
¿Qué es Efecto Compton?

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

¿Cómo funciona?

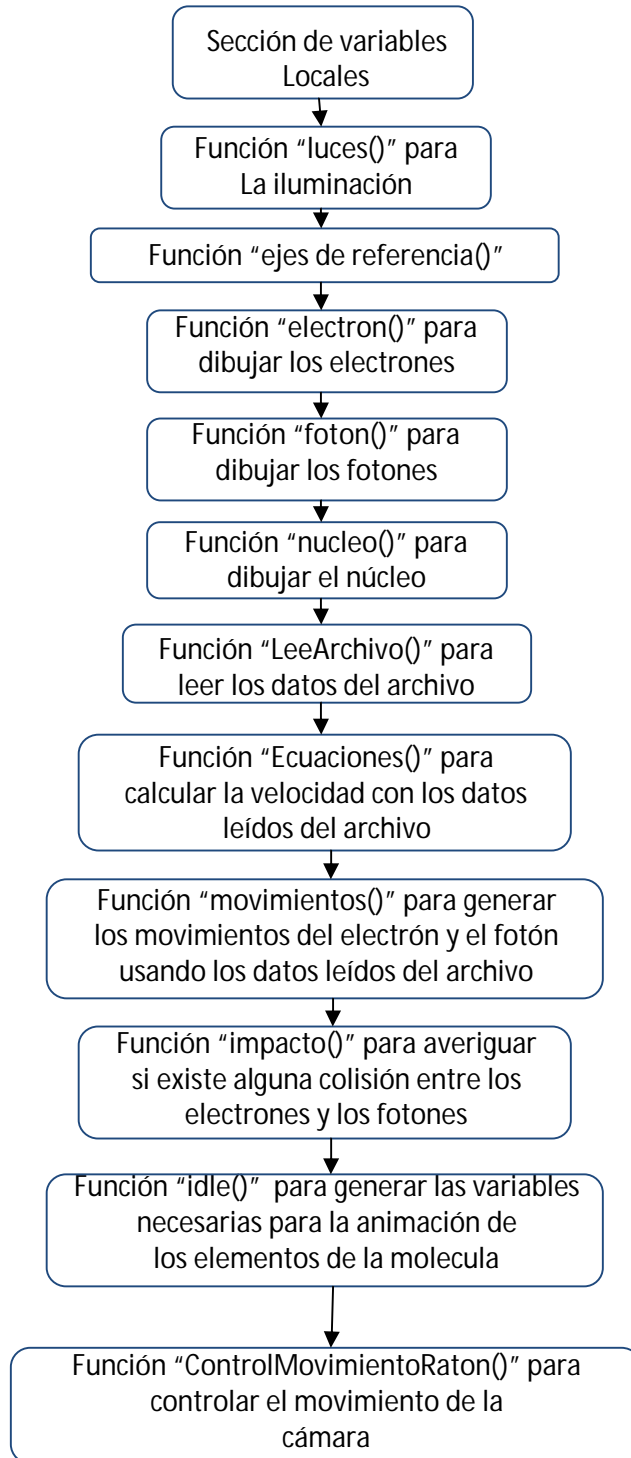


Visto de manera más general, se tiene un haz de luz que incide sobre los electrones de una molécula:

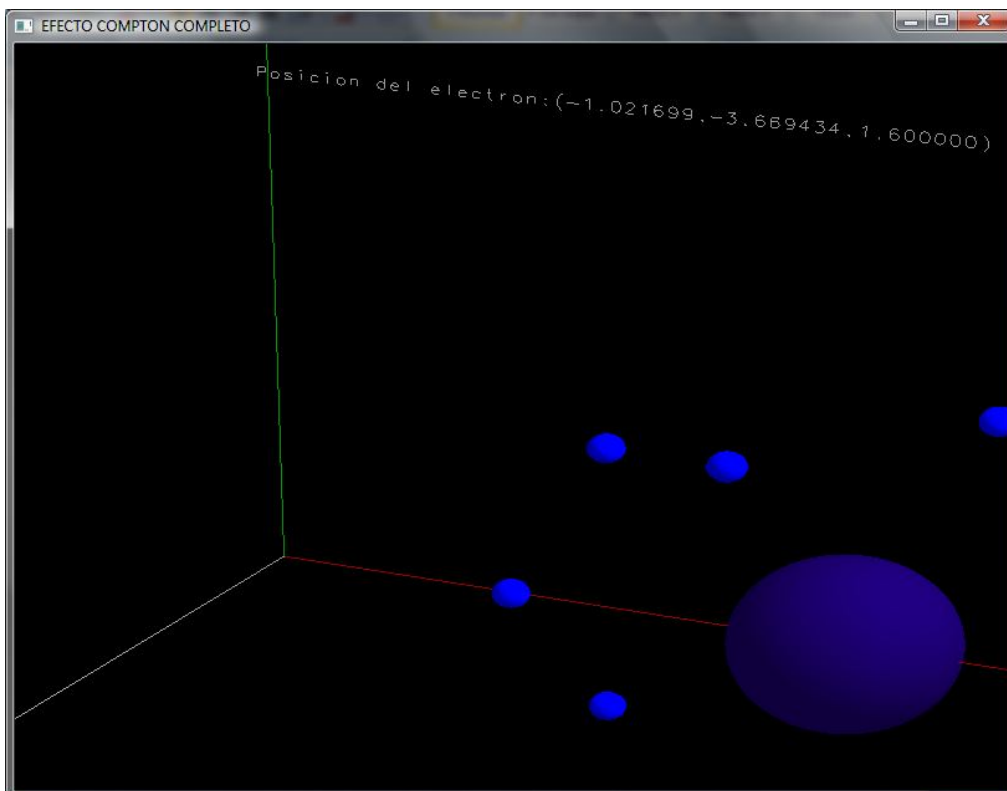
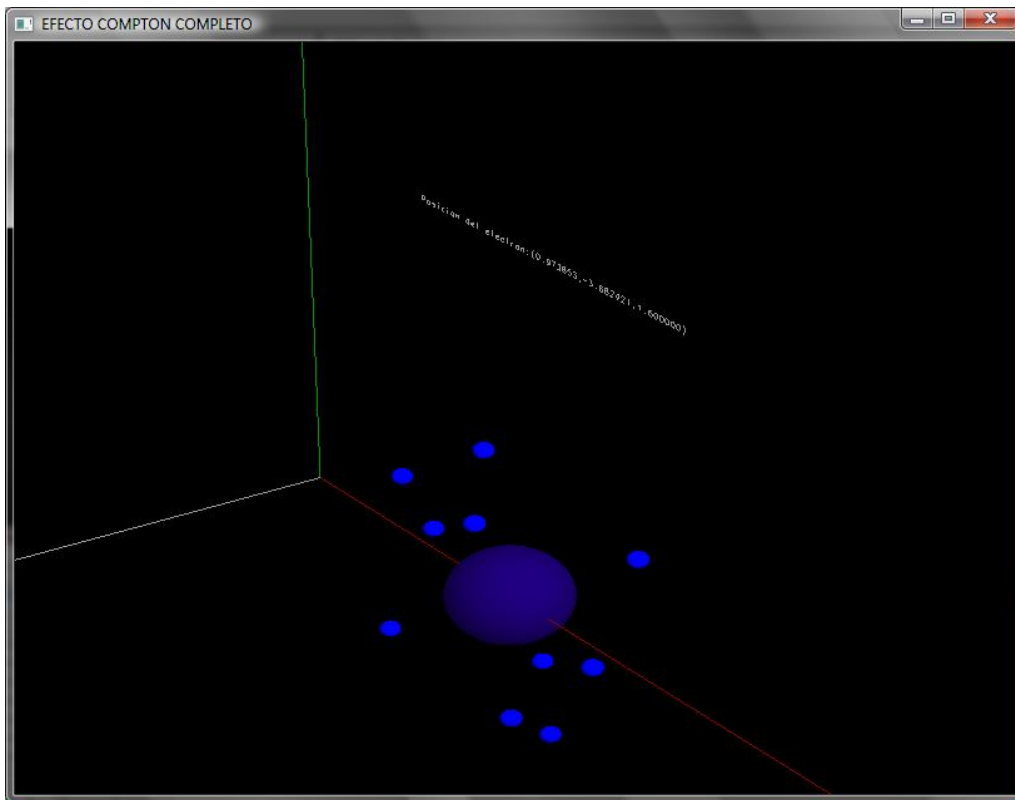


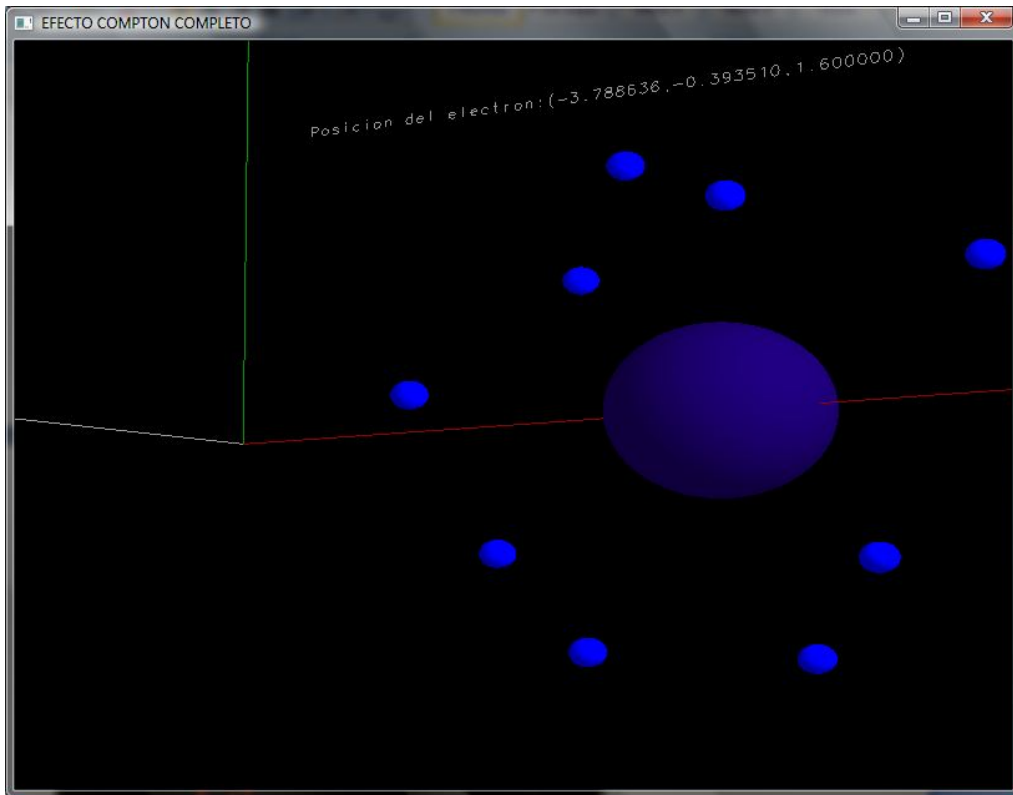
DESARROLLO

Se tiene el siguiente diagrama de bloques para explicar el funcionamiento del programa:

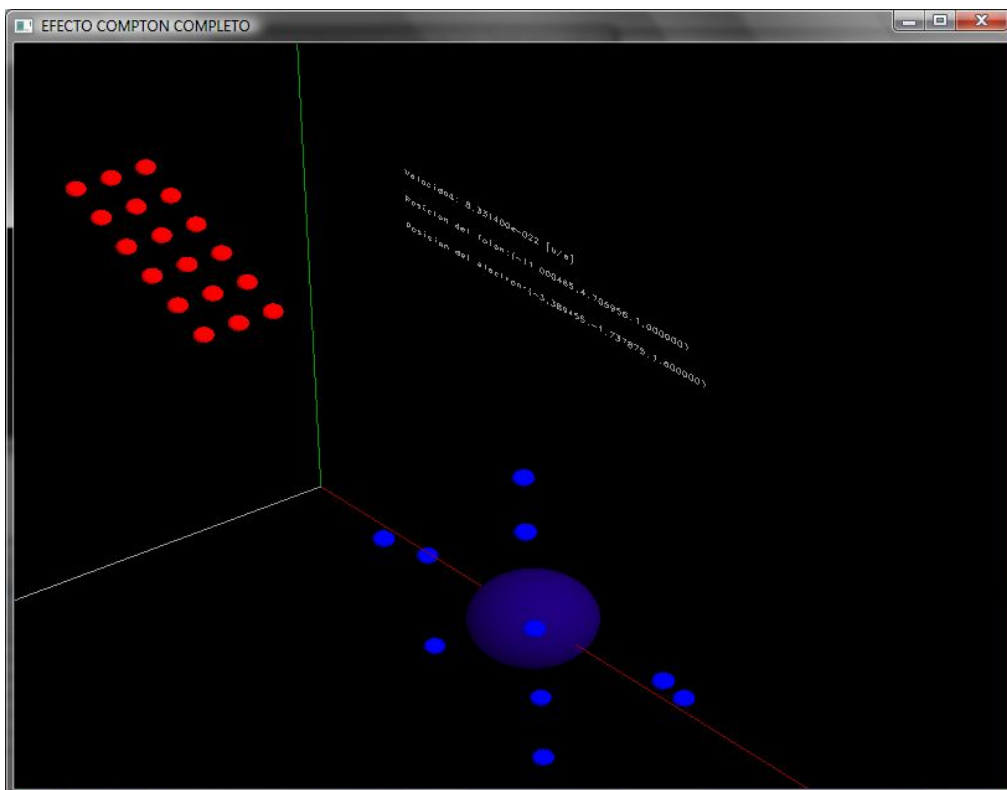


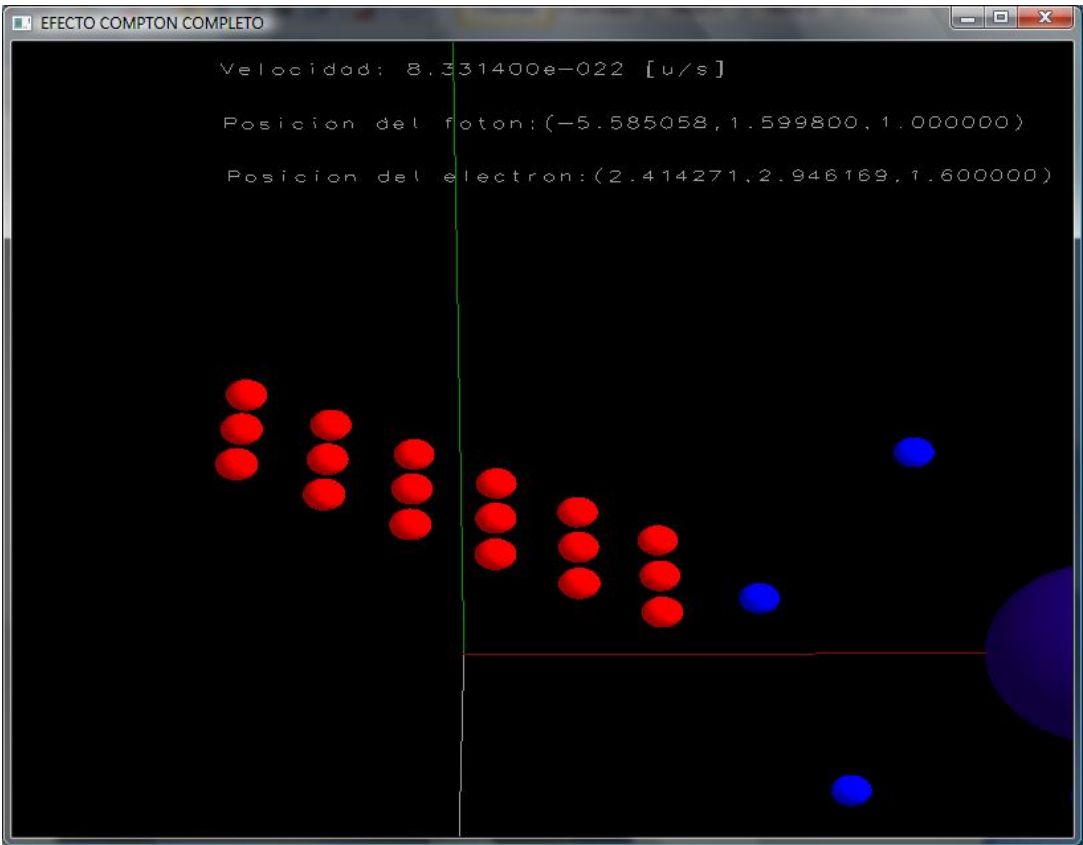
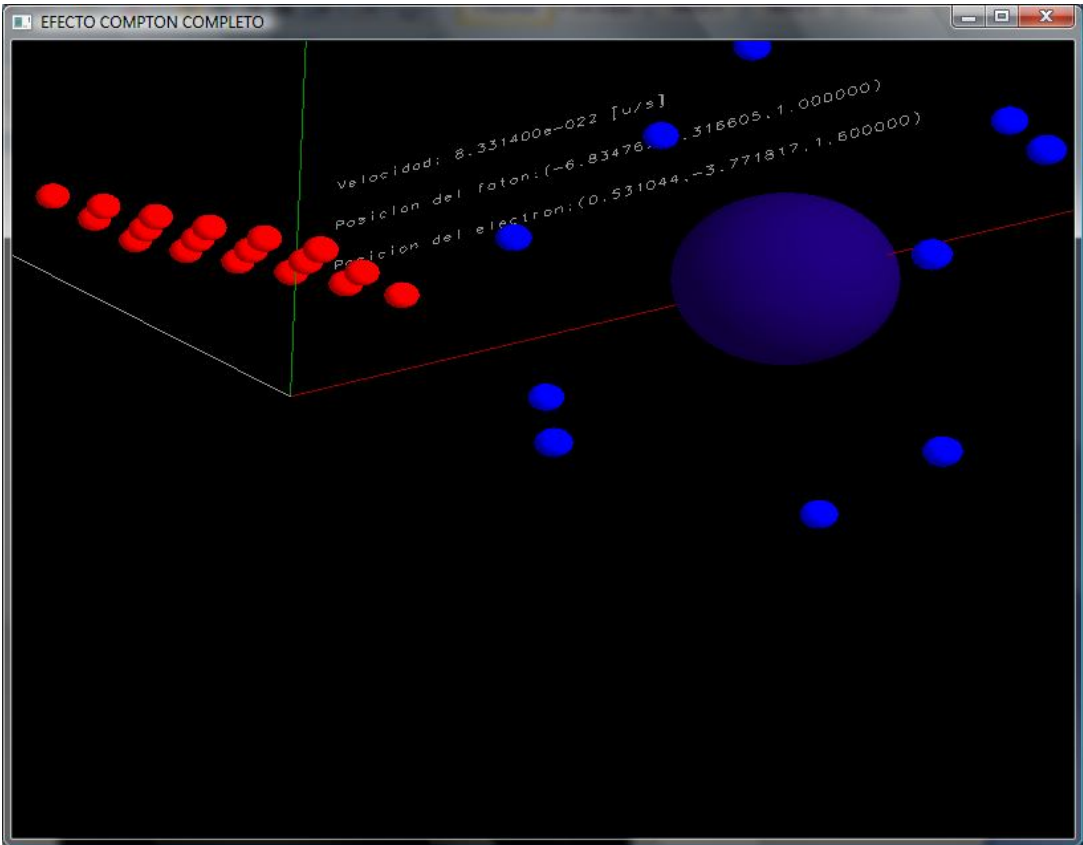
Se tienen las mismas ecuaciones de la práctica anterior, primeramente se tiene una molécula con varios electrones girando a su alrededor:





Posteriormente tenemos también el haz de luz que va a incidir sobre la molécula, se imprime la posición de uno de los electrones y uno de los fotones así como su velocidad:





CODIGO FUENTE

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

#define ESPAI 0.25
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 };
GLfloat OtherMaterial [4] = {0.1, 0.0, 0.4, 1.0 };

struct puntos{
    float x;
    float y;
    float z;
    bool impact;
}mnfoton[20],mnelectron[20];

bool opty=false;
float angle;
float angle1;
float angle2=0;
float v;
float alfa;
float beta;
char vertices[30];
float puntos[30];
char texto[300];
char pfoton[300];
char pelectron[300];
int lon;
int verif;
float teta;
float fi;
float cam1;
float cam2;
float cam3;
float a=0;
float b=0;
float c=0;
float xx=0;
```

```

float yy=0;
float zz=0;

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, 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, 0.25, 300, 300);
    gluDeleteQuadric(quadobj);
    glEndList();
}

void Foton(){
    glNewList(3, 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, 0.25, 300, 300);
    gluDeleteQuadric(quadobj);
    glEndList();
}

void Nucleo(){
    glNewList(4, GL_COMPILE);
    glMaterialfv (GL_FRONT, GL_AMBIENT_AND_DIFFUSE, OtherMaterial);
    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, 1.5, 300, 300);
    gluDeleteQuadric(quadobj);
    glEndList();
}

void ecuaciones(void){
//Para la energia se tiene  $E = hw$ 
//Para  $w = 2 * \pi * f$ 
//Para  $f = 1/l$ 
//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  $E = 4.1657 \times 10^{-33}/ l$ 
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 limpia()
{
for(int yhu=0;yhu<10;yhu++)
{
mnelectron[yhu].x = 0.0;
mnelectron[yhu].y = 0.0;
mnelectron[yhu].z = 0.0;
mnelectron[yhu].impact = false;
}

for(int fds=0;fds<18;fds++)
{
mnfoton[fds].x = 0.0;
mnfoton[fds].y = 0.0;
mnfoton[fds].z = 0.0;
mnfoton[fds].impact = false;
}
}

void movimientos(void){
float inicio[14] = {180,72,288,144,0,216,36,324,108,252,56,245,100,0};
//agrega un desplazamiento inicial a los electrones
float ini;
int orig,cuenta,anh;
cuenta=0;
for(int fgh=-1; fgh<=1; fgh++)//numero de fotones a lo largo de z
{
for(int lde=0; lde<=5; lde++) //numero de fotones a lo largo de x
{
//foton
a = -15+lde*angle;
b = -angle*sin(35*(3.1416/180))-lde*sin(35*(3.1416/180)) + 7.0;
c = fgh;
if(!mnfoton[cuenta].impact)
{
mnfoton[cuenta].y= b;
}else{
mnfoton[cuenta].y += 0.5*sin(teta);
}
mnfoton[cuenta].x= a;
mnfoton[cuenta].z= c;
cuenta++;
}
}
}

orig = 0;
ini=0;
anh=0;
cuenta=0;

```

```

for(int fg=-5; fg<5; fg++) //para colocar varios electrones
{
    ini = inicio[anh]*3.1416/180; //Porque sin y cos trabajan con radianes
no con grados
    //electron
    //Se emplean las siguientes ecuaciones para generar el movimiento en
cada uno de los electrones:
    if(fg<=0)
    {
        xx = (3.5-sin((-18*(fg+5))*3.1416/180))*cos(alfa+ini);
        yy = (3.5-sin((-18*(fg+5))*3.1416/180))*sin(alfa+ini);
        zz = 0.4*fg;
    }else{
        xx = (3.5-sin((18*(-5+fg))*3.1416/180))*cos(alfa+ini);
        yy = (3.5-sin((18*(-5+fg))*3.1416/180))*sin(alfa+ini);
        zz = 0.4*fg;
    }

    if(!mnelectron[cuenta].impact)
    {
        mnelectron[cuenta].x = xx;
        mnelectron[cuenta].y = yy;
    }else{
        mnelectron[cuenta].y = mnelectron[cuenta].y -
0.5*sin(fi);
        mnelectron[cuenta].x -= 0.5;
    }
    mnelectron[cuenta].z = zz;
    cuenta++;
    anh++;
}
}

void impacto(void)
{
    int cuenta;
    cuenta=0;
    for(int yhu=0;yhu<10;yhu++)
    {
        for(int fds=0;fds<18;fds++)
        {
            if(mnelectron[yhu].x<=mnfoton[fds].x)
            {
                if(mnelectron[yhu].y<=mnfoton[fds].y)
                {
                    mnfoton[fds].y += angle*sin(teta);
                    mnelectron[yhu].y = mnelectron[yhu].y-angle*sin(fi);
                    mnelectron[yhu].x -= angle;
                    mnfoton[fds].impact = true;
                    mnelectron[yhu].impact = true;
                }
            }
        }
    }
}

```

```

void idle(void)
{
verif = 0;
angle = angle+(float)v;
angle1 = angle1+0.2; //desde aqui se puede variar facilmente la velocidad
de los electrones
angle2 = angle2+0.2;

glutPostRedisplay();
if (angle>=11){angle=-3; limpia();}
if(alfa<=360)
{
    alfa = (alfa)*(3.1416/180) + angle1;
}else{
    alfa = 0;
    angle1 = 0;
}
if(opty)
    angle2=0;
}

void ControlMovimientoRaton( GLsizei x, GLsizei y )
{
//if(x<=225)
cam1= (float)x*0.4 - 90;
cam2= (float)y*0.4 - 90;
if(x<=450)
{
    cam3 = 40;
    cam1= (float)x*0.4 - 90;
    cam2= (float)y*0.4 - 90;
}else{
    cam3 = -40;
    cam1= -(float)x*0.4 + 270;
    cam2= (float)y*0.4 - 90;
}
}

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

    glMatrixMode(GL_MODELVIEW);
    glLoadIdentity();
    gluLookAt(cam1,cam2,cam3,
              1, 1, -5, // si pones 5,5,-5 se ve cortado
              0, 1,  0);

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

    glMatrixMode(GL_MODELVIEW);
    movimientos();
    impacto();
}

```

```

//Para colocar el nucleo de la molecula
glPushMatrix();
glTranslatef(8,0,0);

//Para el grupo de fotones:
cuenta=0;
for(int hgf=0;hgf<=18;hgf++)
{
    glPushMatrix();

glTranslatef(mnfoton[cuenta].x,mnfoton[cuenta].y,mnfoton[cuenta].z);
    glCallList(3); //llama la lista en donde se definio el foton
    glPopMatrix();
    cuenta++;
}
glRotatef(180,0,1,0);
cuenta=0;
for(int fgh=0;fgh<10;fgh++)
{
    glPushMatrix();

glTranslatef(mnelectron[cuenta].x,mnelectron[cuenta].y,mnelectron[cuenta]
.z);
    glCallList(2); //llama a la lista en donde se encuentra
definido el electron
    glPopMatrix();
    cuenta++;
}
glCallList(4);
glPopMatrix();

//muestra texto
sprintf_s(texto,"Velocidad: %e [u/s]",v/pow(10.0,20));

glPushMatrix();
glTranslatef(-3.0f,8.2f,-5.0f);
glScalef(0.002f,0.002f,0.002f);
glMaterialfv(GL_FRONT, GL_AMBIENT_AND_DIFFUSE, WhiteMaterial );
glutStrokeString(GLUT_STROKE_MONO_ROMAN,(unsigned char*)texto);
glPopMatrix();

sprintf_s(pfoton,"Posicion del foton:(%f,%f,%f)",a,b,c);

glPushMatrix();
glTranslatef(-3.0f,7.2f,-5.0);
glScalef(0.002f,0.002f,0.002f);
glMaterialfv(GL_FRONT, GL_AMBIENT_AND_DIFFUSE, WhiteMaterial );
glutStrokeString(GLUT_STROKE_MONO_ROMAN,(unsigned char*)pfoton);
glPopMatrix();

sprintf_s(pelectron,"Posicion del electron:(%f,%f,%f)",xx,yy,zz);

glPushMatrix();
glTranslatef(-3.0f,6.2f,-5.0);
glScalef(0.002f,0.002f,0.002f);
glMaterialfv(GL_FRONT, GL_AMBIENT_AND_DIFFUSE, WhiteMaterial );
glutStrokeString(GLUT_STROKE_MONO_ROMAN,(unsigned char*)pelectron);

```

```

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 COMPLETO MZCA");
    glutDisplayFunc(RenderScene);
    glutIdleFunc( idle );
    glutMotionFunc ( ControlMovimientoRaton );
    glutSetCursor(GLUT_CURSOR_CROSSHAIR);
    glClearColor(0.0f, 0.0f, 0.0f, 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();

    Nucleo();
    Electron();
    Foton();
    ecuaciones();

    angle=-2;
    angle1=0;
    cam1=0;
    cam2=0;
    cam3=0;
    alfa=0;

    glutMainLoop();

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

```

```
    return 0;  
}
```

CONCLUSIONES

En este proyecto podemos ver el efecto de la última práctica pero esta vez aplicado a muchos elementos, con esto la dificultad aumenta sobre todo al hacer las colisiones ya que debemos conocer el punto en el que el electrón y el fotón se encuentran, además, en este proyecto se reúnen todos los conocimientos previos como animación, modelado jerárquico, listas, cámara, iluminación, etc.