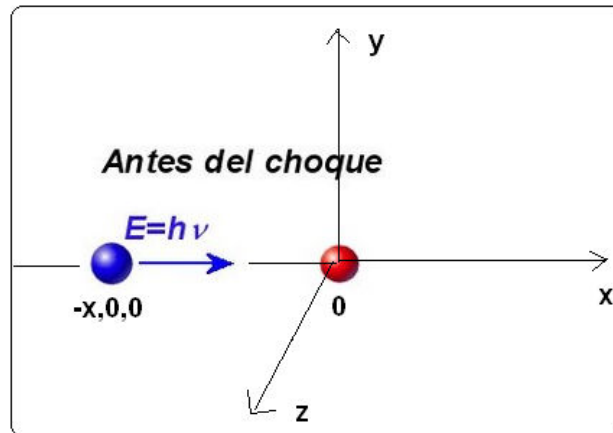


PROYECTO FINAL:

EFFECTO COMPTON COMPLETO

Efecto Compton: este efecto es consecuencia de una colisión entre el fotón gamma incidente y un electrón que se encuentre libre en el material. Este electrón porta una energía que depende del ángulo con que fue dispersado finalmente el fotón.



DESARROLLO:

Efecto Compton completo.

Tenemos dos entes, 1) electrón 2) fotón; Ahora tenemos un grupo de electrones y un grupo de fotones.

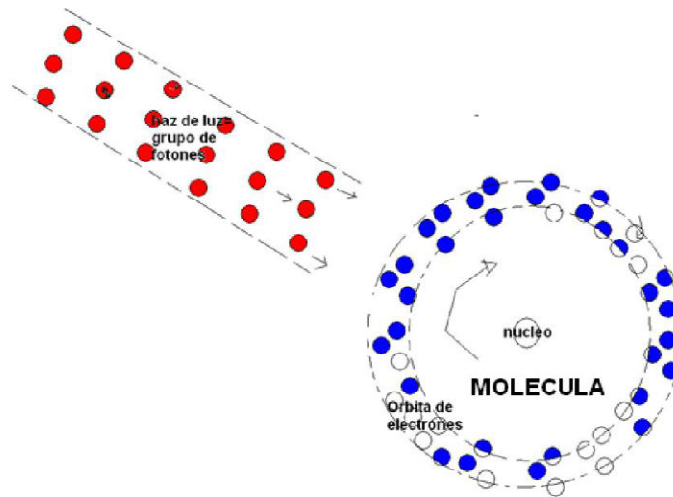
Y se requiere SIMILAR con las ecuaciones anteriores, pero ahora dado la posición "x" de cada fotón su desplazamiento dirigido en grupo como un "haz de luz" hacia la molécula. Mientras que la molécula está girando en el sentido de las manecillas del reloj y cada electrón giran sobre su propio eje también con respecto a las manecillas del reloj. Entonces si se confirma visualmente y matemática que chocará un fotón con un electrón, deberá aplicarse la ecuación del efecto Compton.

Debe poder verse por diversos cambios de vista de la cámara; deben moverse fotones a una velocidad con una dirección y los electrones deberán tener dos movimientos, uno alrededor del núcleo y sobre sí mismo.

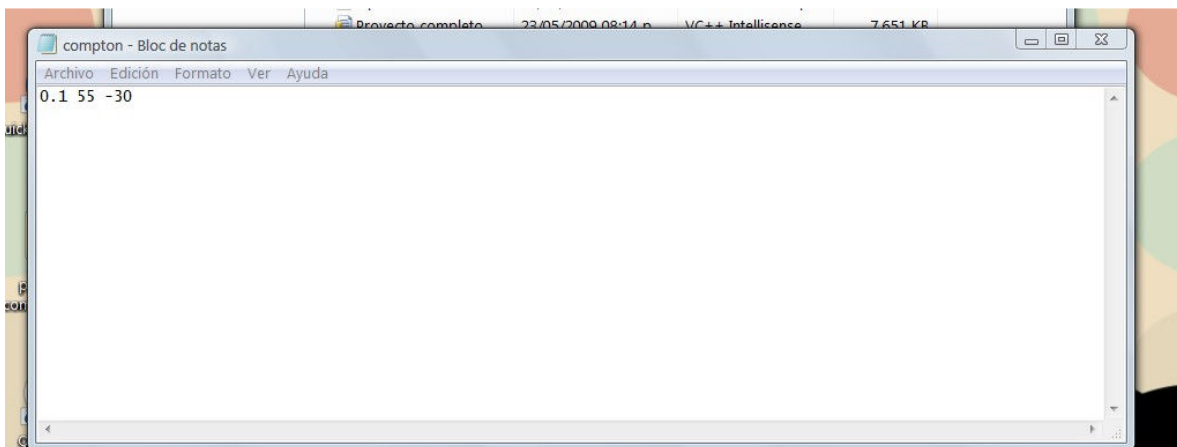
Para la entrega se les pide formato tipo Práctica, desglosar procesos, es decir, presentar un diagrama de bloques de cuantos procesos se requieren tener para que TODO funcione.

Se calificará Simulaciones parciales y totales:

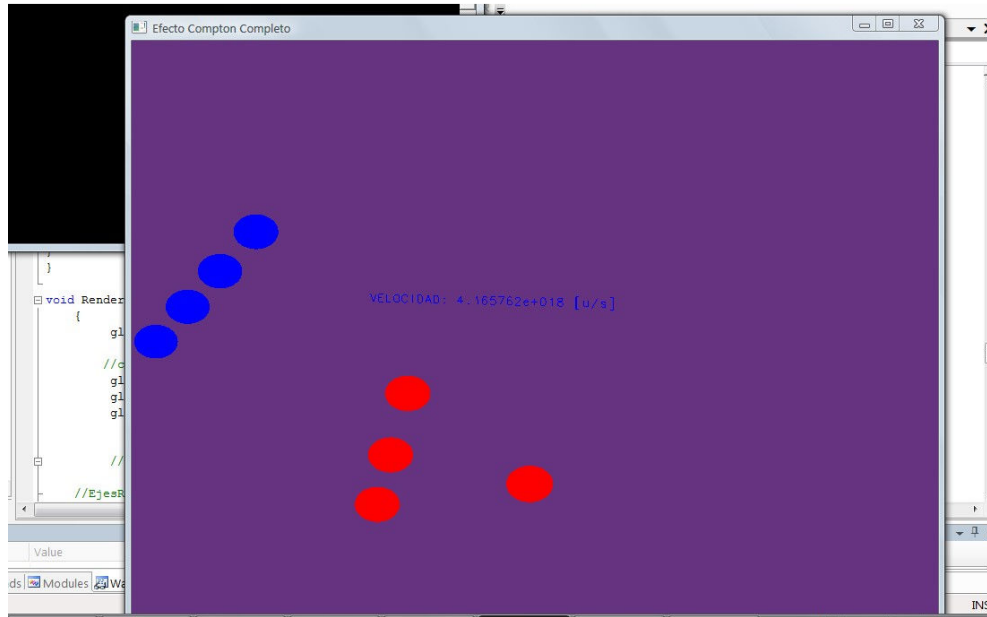
- 1) Movimiento de varios electrones con una velocidad y una posición CONOCIDA (deberá poderse imprimir)*
- 2) Movimiento de nube de electrones alrededor del núcleo. Imprimir Velocidad angular y posición)*



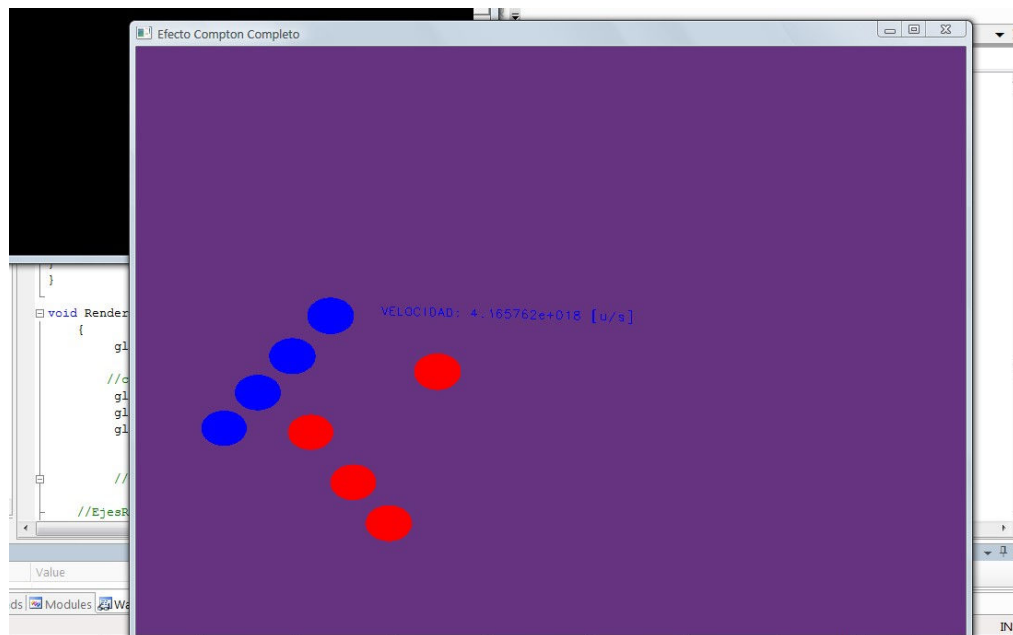
DEMOSTRACIÓN:



Utilizando este archivo .txt



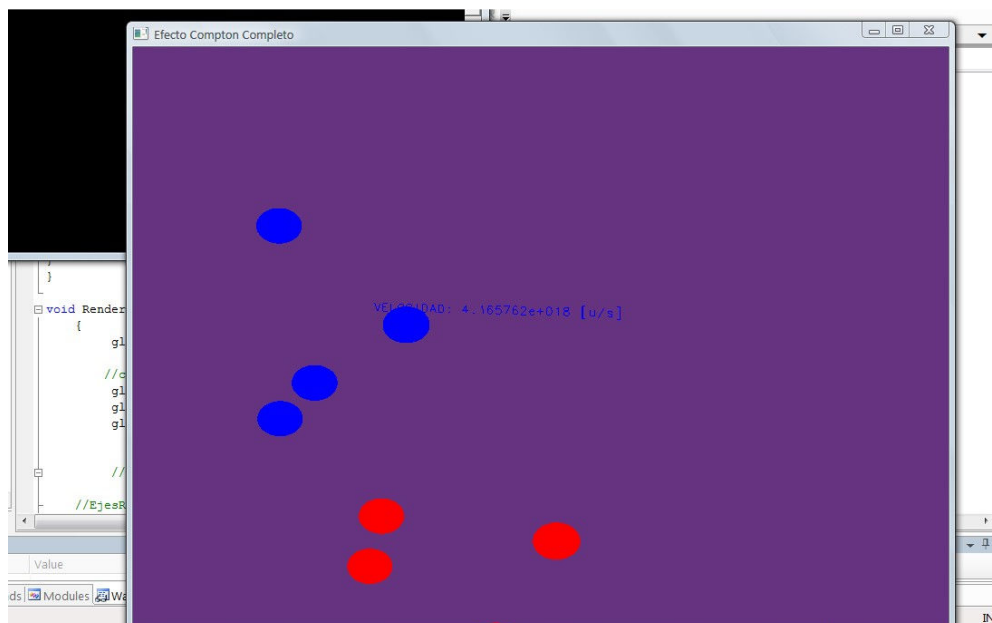
Los fotones y los electrones moviéndose



Acercados los protones a los electrones



Durante el choque



Después del choque

CÓDIGO DEL PROYECTO.

```
#include <conio.h>
#include <cstdlib>
#include <iostream>
#include <string>
```

LABORATORIO DE COMPUTACIÓN GRÁFICA
ALUMNA: OLIVA GARCÍA GABRIELA

```
#include <fstream>
#include "openglut.h"
#include <math.h>
#include <stdio.h>

GLUquadricObj *particula, *particula2, *particula3, *particula4,
*particula5,
*particula9, *particula10, *particula11, *particula12;

//Definición del modelo de una luz
GLfloat light_Ambient [4] = { 0.2, 0.2, 0.2, 1.0};
GLfloat light_Diffuse [4] = { 0.9, 0.9, 0.9, 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 };

double w;
double e;
double f;
double x=3.1416;
double xpos=-4;
double xpos2=-4;
double xpose;
double h=6.63E-34;
char lam[30];
char angul[5];
char angule[5];
char texto[300];
double la;
double angulo;
double angu;
double xnueva;
double anguloe;
double angue;
double t=1E-30;
double ang;
double y;
double y2;
double y3;
double hugo;
double m;
double xe;
double z;
double an;
double gabi;
double k;
double j;
double s;
double v;
```

```
float camara1=0;
float camara2=0;
float camara3=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()
{
glLineWidth(0.1);
glEnable(GL_LINE_STIPPLE);
glBegin (GL_LINES);

    glVertex3f ( 0.0, 0.0, 0.0);
    glVertex3f (20.0, 0.0, 0.0);

    glColor3f(0.0,0.0,1.0);
    glVertex3f ( 0.0, 0.0, 0.0);
    glVertex3f ( 0.0,20.0, 0.0);

    glColor3f(0.0,0.0,1.0);
    glVertex3f ( 0.0, 0.0, 0.0);
    glVertex3f ( 0.0, 0.0,20.0);
glEnd();
}

void esfera()
{
    glColor3f(0.0, 0.0, 1.0);
    glutSolidSphere(0.4,55,55);
}

void esfera2()
{
    glColor3f(0.0, 0.0, 1.0);
    glutSolidSphere(0.4,55,55);
}

void esfera3()
{
    glColor3f(0.0, 0.0, 1.0);
    glutSolidSphere(0.4,55,55);
}

void esfera4()
{
    glColor3f(1.0, 0.0, 0.0);
    glutSolidSphere(0.4,55,55);
}
```

```
void esfera5()
{
    glColor3f(0.0, 0.0, 1.0);
    glutSolidSphere(0.4,55,55);
}

void esfera9()
{
    glColor3f(1.0, 0.0, 0.0);
    glutSolidSphere(0.4,55,55);
}

void dibujarparticula()
{
    particula= gluNewQuadric();
    glColor3f(0.0, 0.0, 1.0);
    esfera();
    gluDeleteQuadric(particula);
}

void dibujarparticula2()
{
    particula2= gluNewQuadric();
    glColor3f(0.0, 0.0, 1.0);
    esfera2();
    gluDeleteQuadric(particula2);
}

void dibujarparticula3()
{
    particula3= gluNewQuadric();
    glColor3f(0.0, 0.0, 1.0);
    esfera3();
    gluDeleteQuadric(particula2);
}

void dibujarparticula4()
{
    particula4= gluNewQuadric();
    glColor3f(1.0, 0.0, 0.0);
    glRotatef(0,0.0,0.0,1.0);
    glTranslatef(2.2,-2.2,0);
    esfera4();
    gluDeleteQuadric(particula2);
}

void dibujarparticula5()
{
    particula5= gluNewQuadric();
    glColor3f(0.0, 0.0, 1.0);
    esfera5();
    gluDeleteQuadric(particula2);
}

void dibujarparticula9()
{
    particula9= gluNewQuadric();
    glColor3f(1.0, 0.0, 0.0);
    glTranslatef(1.5,0.9,0);
    esfera9();
}
```

```
gluDeleteQuadric(particula2);
}
void dibujarparticula10()
{
particula10= gluNewQuadric();
glColor3f(1.0, 0.0, 0.0);
glTranslatef(0.0,0.0,0);
esfera9();
gluDeleteQuadric(particula2);
}
void dibujarparticula11()
{
particula11= gluNewQuadric();
glColor3f(1.0, 0.0, 0.0);
glTranslatef(0.0,0.0,0);
esfera9();
gluDeleteQuadric(particula2);
}
void dibujarparticula12()
{
particula12= gluNewQuadric();
glColor3f(1.0, 0.0, 0.0);
glTranslatef(-0.4,1.6,0);
esfera9();
gluDeleteQuadric(particula2);
}

void idle(void)
{

glutPostRedisplay();
xpos=xpos+e;
gabi=xpos;
an=-45;
angulo=-45;

j=-45;

k=xpos;

xe=1.8;
hugo=3.5;
m=-0.9;

z=2.0;

    y++;
    y2++;
    y3++;

if (xpos>=-0.2)
{
xe=-1.8;
```

```
z=1.8;
xpose=xpos+e;
gabi=xpose;
an=-45;
angulo=-45;

j=-45;

k=xpose;

hugo=3.5;
m=-0.9;
    if (xpose=4.0)
    {

xpose=xpos+e;
xe=xpose;
gabi=0;
angulo=angu;
j=angu;
k=xpose;
y2=angue;
an=-45;

hugo=xpose;
y3=angue;

    }
if (xpos>=7)
{

    xpos=-8;
    gabi=xpos;
    y2=y;
    xe=1.8;
    z=2.0;
    an=-45;
    angulo=-45;
    j=-45;
    k=xpos;
    y3=y;
    hugo=3.5;
    m=-0.9;
}
}

} //fin de Idle

void Archivo()
{
std::ifstream texto("compton.txt");
if (texto.is_open())
{
    texto >> lam;
    texto >> angul;
    texto >> angule;
}
```

```
texto.close();

la=atof(lam); //transforma char a double
angu=atof(angul);
angue=atof(angule);
f=(1/(la*t));
w=2*x*f;
e=h*w;

} //fin de if

else std::cout<<"No se pudo abrir el archivo";
} //fin de funcion

void raton( GLsizei x, GLsizei y )
{
//if(x<=225)
camara1= (float)x*0.4 - 90;
camara2= (float)y*0.4 - 90;
if(x<=450)
{
camara3 = 40;
camara1= (float)x*0.4 - 90;
camara2= (float)y*0.4 - 90;
} else{
camara3 = -40;
camara1= -(float)x*0.4 + 270;
camara2= (float)y*0.4 - 90;
}
}

void RenderScene(void)
{
    glClear(GL_COLOR_BUFFER_BIT|GL_DEPTH_BUFFER_BIT);

    //camara
    glMatrixMode(GL_MODELVIEW);
    glLoadIdentity();
    gluLookAt( camara1, camara2, camara3,
               1, 1, -5, // si pones 5,5,-5 se ve cortado
               0, 1, 0);

    //fin camara

    //EjesReferencia();
    Archivo();

    char letrero[1000];
    sprintf_s(letrero, "VELOCIDAD: %e [u/s]", e*10e19);

    glPushMatrix();
    glTranslatef(-3.3f, 2.2f, -5.0f);
    glScalef(0.002f, 0.002f, 0.002f);
    glColor3f(0.0, 0.0, 1.0);
    glutStrokeString(GLUT_STROKE_MONO_ROMAN, (unsigned char*)letrero);
```

```
glPopMatrix();

////////////////////////////////////

    glMatrixMode(GL_MODELVIEW);
    glPushMatrix();
    glRotatef(j,0.0,0.0,1.0);           //primero se rota...
    glTranslatef(k,2.5,1);           //despues se translada...
    dibujarparticula();
    glPopMatrix();

////////////////////////////////////

    glPushMatrix();
    glRotatef(an,0.0,0.0,1.0);
    glTranslatef(gabi,1.2,1.0);
    dibujarparticula2();
    glPopMatrix();

    glPushMatrix();
    glRotatef(an,0.0,0.0,1.0);
    glTranslatef(gabi,0.0,1.0);
    dibujarparticula3();
    glPopMatrix();

////////////////////////////////////Centro

    glPushMatrix();
    glRotatef(0,0.0,0.0,1.0);
    glTranslatef(-0.7,0.0,0.0);
    dibujarparticula4();

    glPushMatrix();
    glRotatef(y,0.0,0.0,1.0);
    glTranslatef(-0.7,0.0,0.0);
    dibujarparticula9();
    glPopMatrix();

    glPushMatrix();
    glRotatef(y2,0.0,0.0,1.0);
    glTranslatef(xe,z,0.0);
    dibujarparticula10();
    glPopMatrix();

    glPushMatrix();
    glRotatef(y3,0.0,0.0,1.0);
    glTranslatef(hugo,m,0.0);
    dibujarparticula11();
    glPopMatrix();
    glPopMatrix();

////////////////////////////////////

    glPushMatrix();
    glRotatef(angulo,0.0,0.0,1.0);
    glTranslatef(xpos,-1.2,1.0);
    dibujarparticula5();
```

```
        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); //NOTE QUE
SE ASIGNO GLUT_DEPTH utilizar buffer de profundidad
    glutInitWindowSize(900,700);
    glutInitWindowPosition(300,100);
    IdeWindow=glutCreateWindow("Efecto Compton Completo");
    glutDisplayFunc( RenderScene );
    glutIdleFunc ( idle );
    glutMotionFunc ( raton );
    //glutKeyboardFunc (ControlTeclado);
    glClearColor(0.4f, 0.2f, 0.5f, 0.7f);

    //
//Habilitando el bufer de profundidad
    glEnable(GL_DEPTH_TEST);
//cte GL_DEPTH_TEST HABILITADO
//
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    gluPerspective(15,800/600,.1,1000); //explicacion abajo
    glMatrixMode(GL_MODELVIEW);
    glLoadIdentity();
    gluLookAt(30,0,30,
              0,0,0, // si pones 5,5,-5 se ve cortado
              0,1,0);
    //camara tiene un ojo, posicion de lente y giro de tripie.No esta
en pixeles
// eyex, eyey, eyez: especifica la poisicon de la lente de la camara u
ojo.
// centerx, centery, centerz: especifica la posion del punto al que se
mira
//upx, upy, upz: especifica el vector de orientacion de la camara

    //luces();

    glutMainLoop();

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

    return 0;
}
```

CONCLUSIÓN.

Con este proyecto se pudo corroborar todo el conocimiento adquirido durante el curso para poder realizar múltiples animaciones donde se involucren muchos elementos en escena. Además se vuelve a reafirmar el uso de pilas dentro de las animaciones y también la interacción con el usuario usando el mouse.