

IMPORTAR/EXPORTAR ARCHIVOS y Proyecto No. 1

Realiza un programa con opengl que lea y escriba archivos de tipo TXT, y que con los datos se grafiquen(sugerencia: punto a punto)

(Para visual Studio: o el software que estés utilizando)

Proyecto No. 1:

“Efecto COMPTON 1”

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”.

Introducción:

¿Que 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 ?

¿Qué debe verse?

Dos pelotitas una azul(Foton) y una roja(electron).

Parte 1. En la pantalla inicial, deben colocarse ambas pelotas en una posición inicial.

Parte 2. Comenzaran a moverse; el fotón deberá chocar con el electrón y ambos saldrán disparados en diferentes direcciones. Para que se realice esto requerimos:

Partimos de obtener del usuario (mediante un archivo de texto), la longitud de onda (λ) 10^{-10} m, ya dentro de nuestro proyecto calculamos obtenemos la f (frecuencia), ω (ω), y con esto calculamos la energía (E), porque $h = cte$

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 IMPORTANTE: los valores de la longitud de onda a ingresar deben estar del orden de 10^{-10} m.

b) Se debe encontrar el punto cuando la posición de la esfera roja ($x1$) llega a la posición de la esfera azul($x2$), tomando en cuenta radios ; si $x1=x2$, deberán cambiar de posición y dirección ambas esferas.

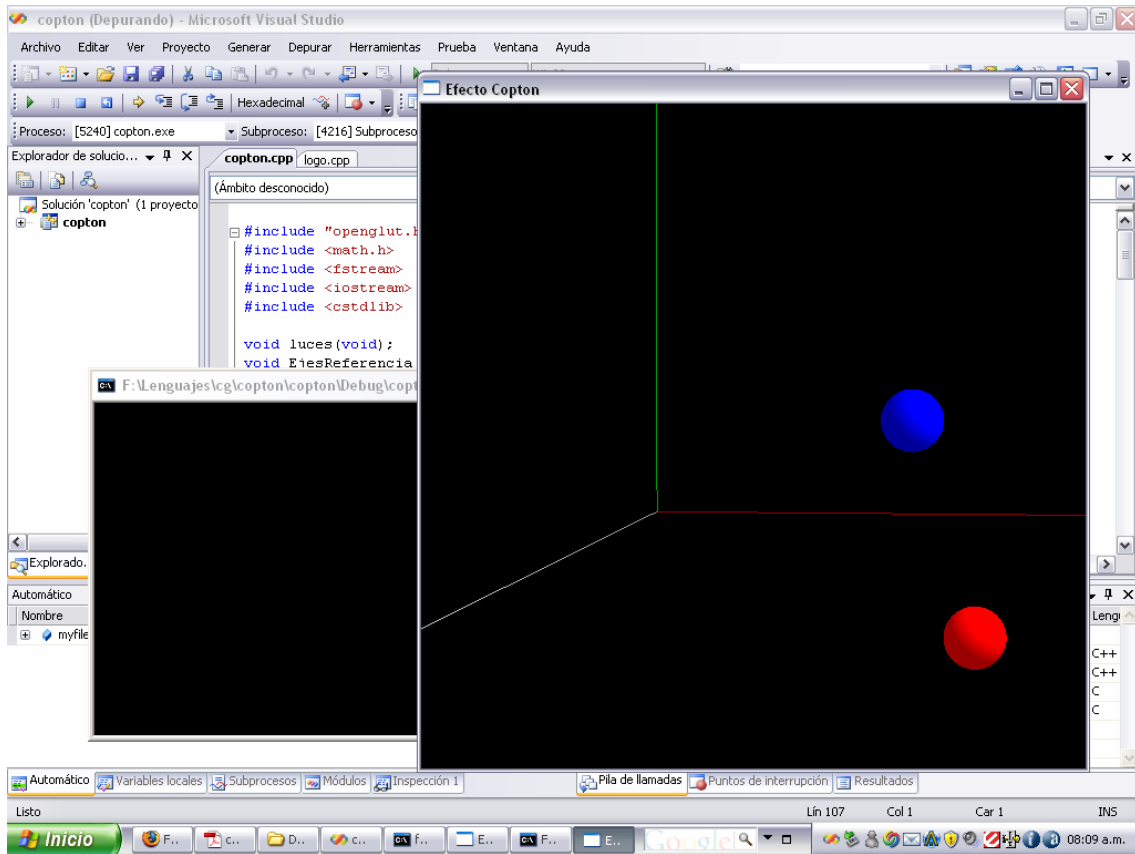
c) Después de chocar, para este proyecto 1, deberán partir en la misma dirección del fotón inicial, pero cada una con un ángulo fijo(dado por el usuario en la tabla del archivo de texto)

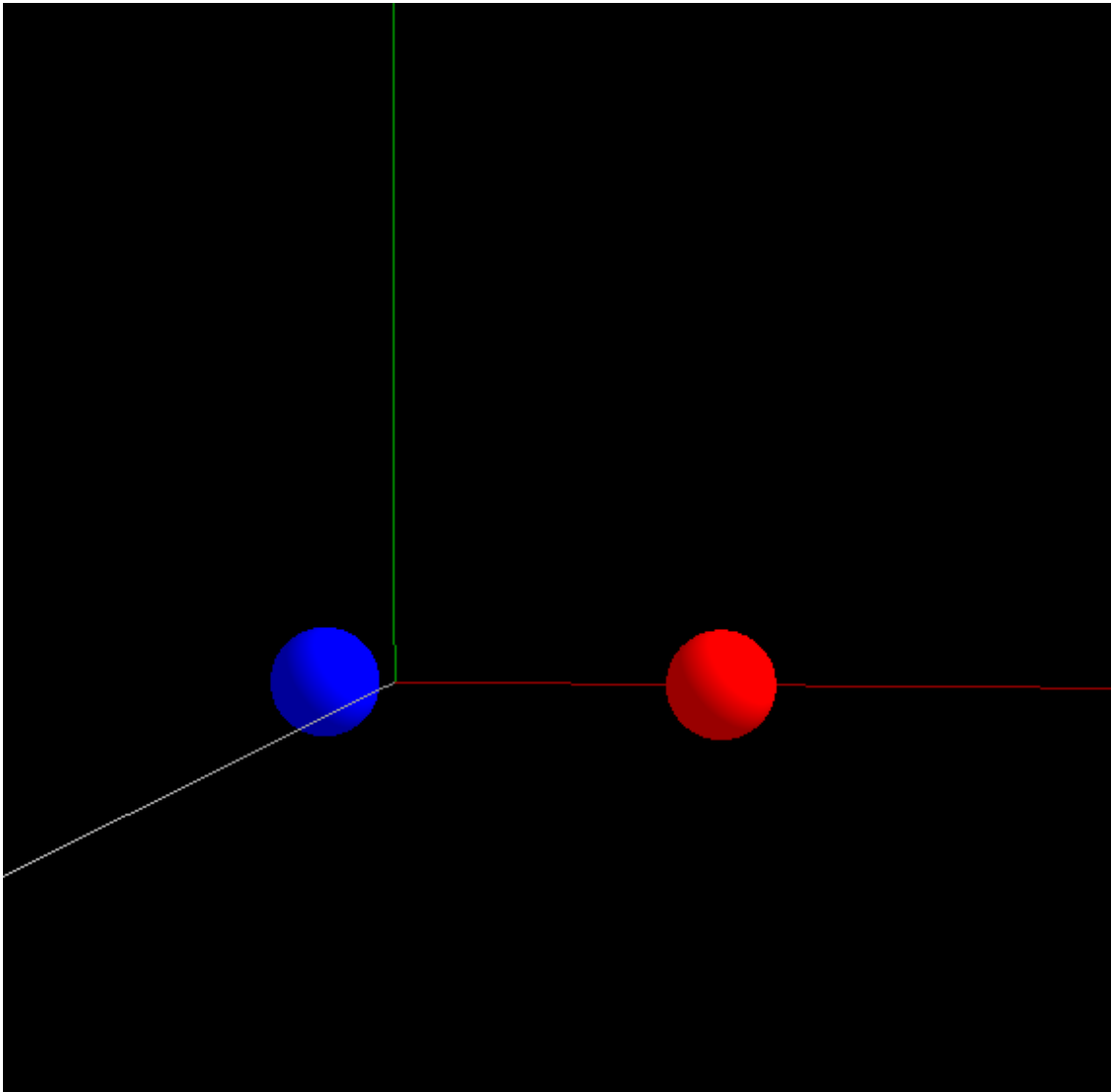
d) En nuestro archivo de txt deberemos poner una tabla

λ - longitud de onda inicial 10^{-10} m.)

θ - ángulo después de golpe. Fotón (grados)

θ - Angulo después del golpe .electrón(grados)





El código del programa fue el siguiente

```
#include "openglut.h"
#include <math.h>
#include <fstream>
#include <iostream>
#include <cstdlib>

void luces(void);
void EjesReferencia();
void LeeArchivo(void);
void Electron();
void Foton();
void ecuaciones(void);
void idle(void);
void RenderScene(void);
```

```

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;

int main(void)
{
int IdeWindow;
glutInitDisplayMode(GLUT_DOUBLE | GLUT_RGB | GLUT_DEPTH);
glutInitWindowSize(600,600);
glutInitWindowPosition(0,0);
IdeWindow=glutCreateWindow("Efecto Copton");
glutDisplayFunc(RenderScene);
glutIdleFunc( idle );
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();
Electron();
Foton();
ecuaciones();
angle=-2;
glutMainLoop();
//Destruir la ventana y el contexto GL
glutDestroyWindow(IdeWindow);
return 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 );

```

```

} // funcion de luces
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();
} // funcion de ejes de referencia
void LeeArchivo(void){
lon=0 ;
for(int h=0; h<30; h++)
{
vertices[h] = ' ';
puntos[h] = 0.0;
}
std::ifstream myfile("file.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  $f = 1/l$ 
//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 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 OpenGL*/
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();
}

```

Conclusión

En esta practica aprendimos a importar datos de un archivo con extensión .txt para graficar por computadora y se utilizo lo aprendido en practicas anteriores lo mas notable a la hora de graficar es que entre mas pequeña sea la longitud de onda el movimiento de las partículas es mas rápido y si la longitud de onda es por ejemplo uno el movimiento es muy lento.