

Proyecto Final

Efecto Compton

Objetivo:

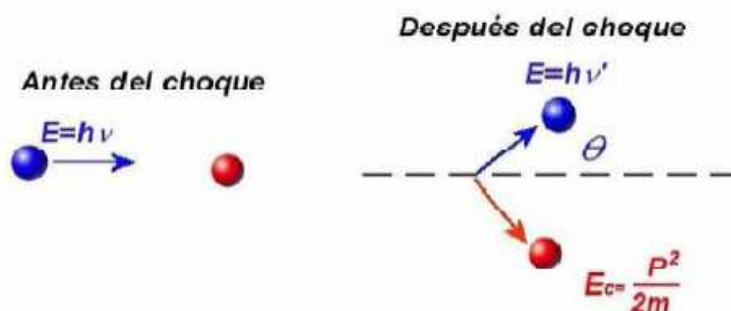
Realizar una animación del efecto Compton con una molécula. Enseguida se muestran imágenes del programa en funcionamiento: Aquí se muestran los datos ingresados en el archivo que será leído por el programa, cabe destacar que se comprobó que mientras más grande era la longitud de onda más lento avanzaba el fotón, de tal manera que si se asignaba el valor de 1 el avance del fotón era casi imperceptible para este caso, mientras al contrario si era más pequeña el fotón aumentaba su velocidad. Los datos están acomodados como lo solicita el documento del examen: longitud de onda, ángulo del fotón, ángulo del neutrón.

Introducción

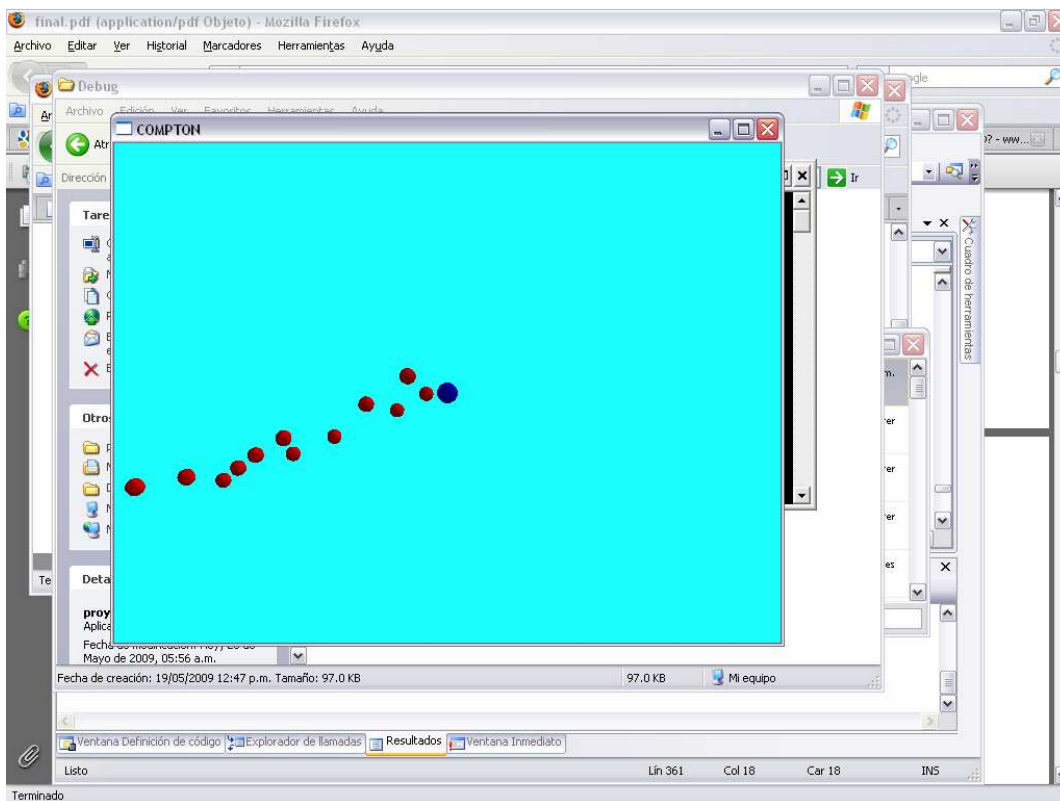
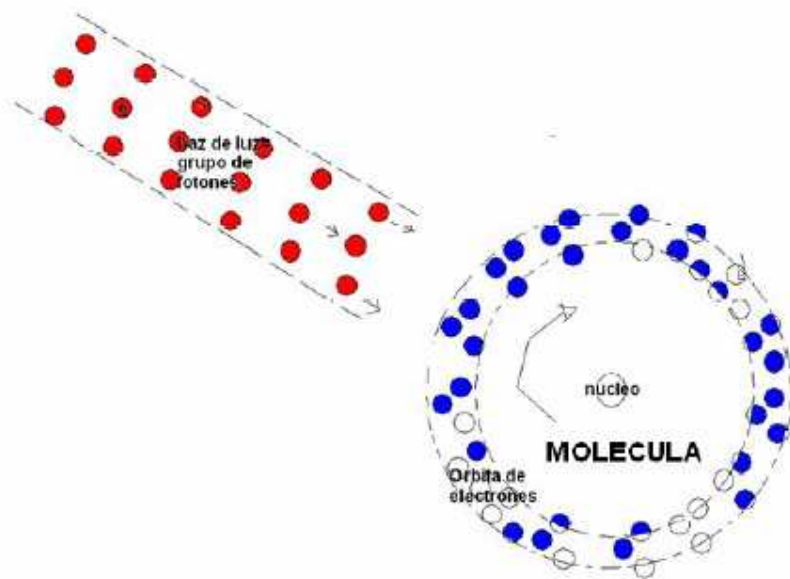
¿Qué es Efecto Compton?

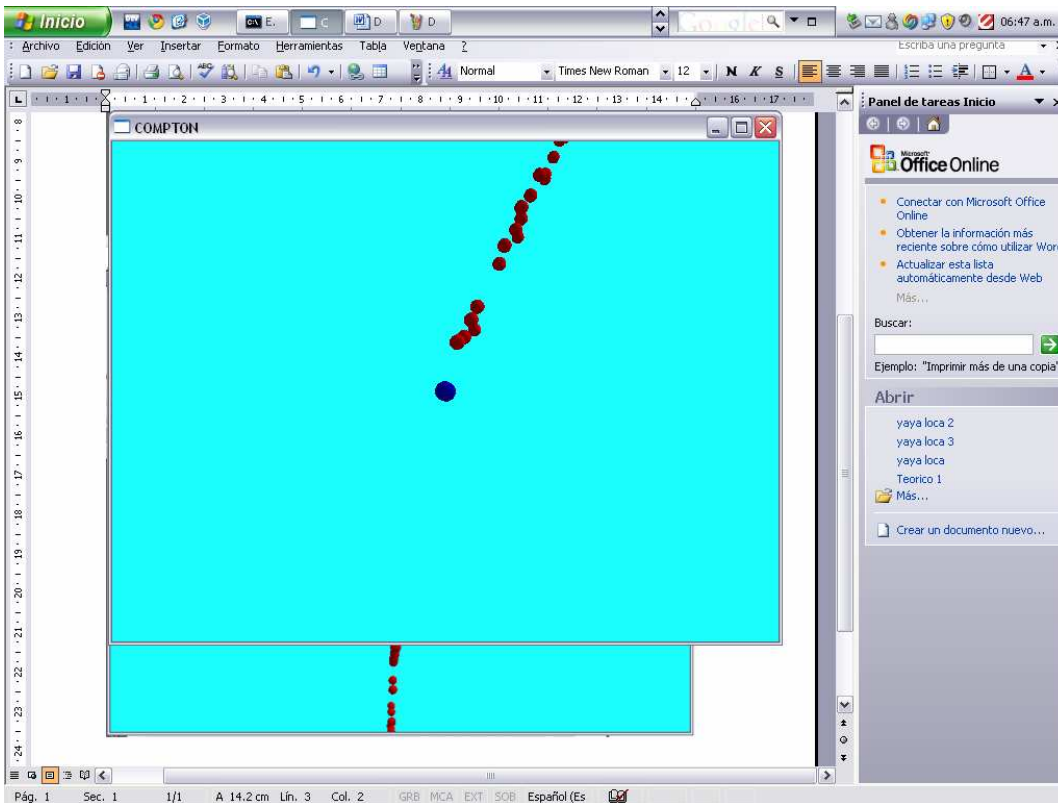
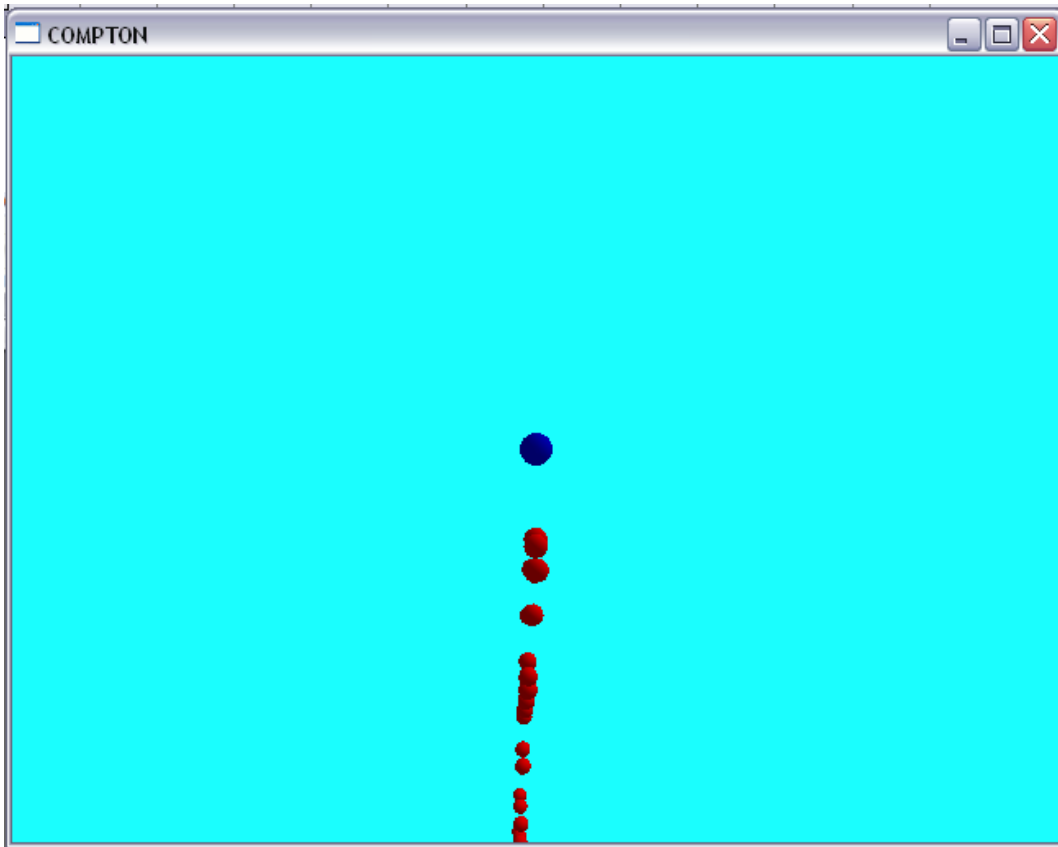
Es la desviación en movimiento del golpe de un fotón contra un electrón, donde ambos salen disparados en diferentes direcciones.

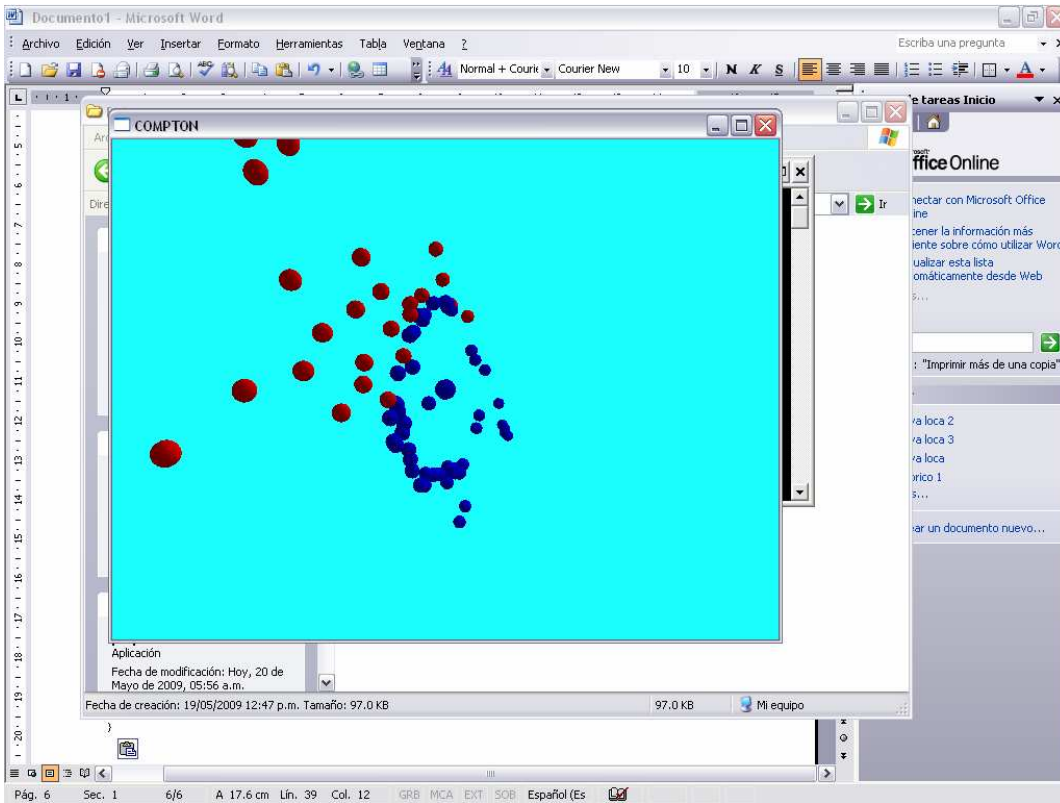
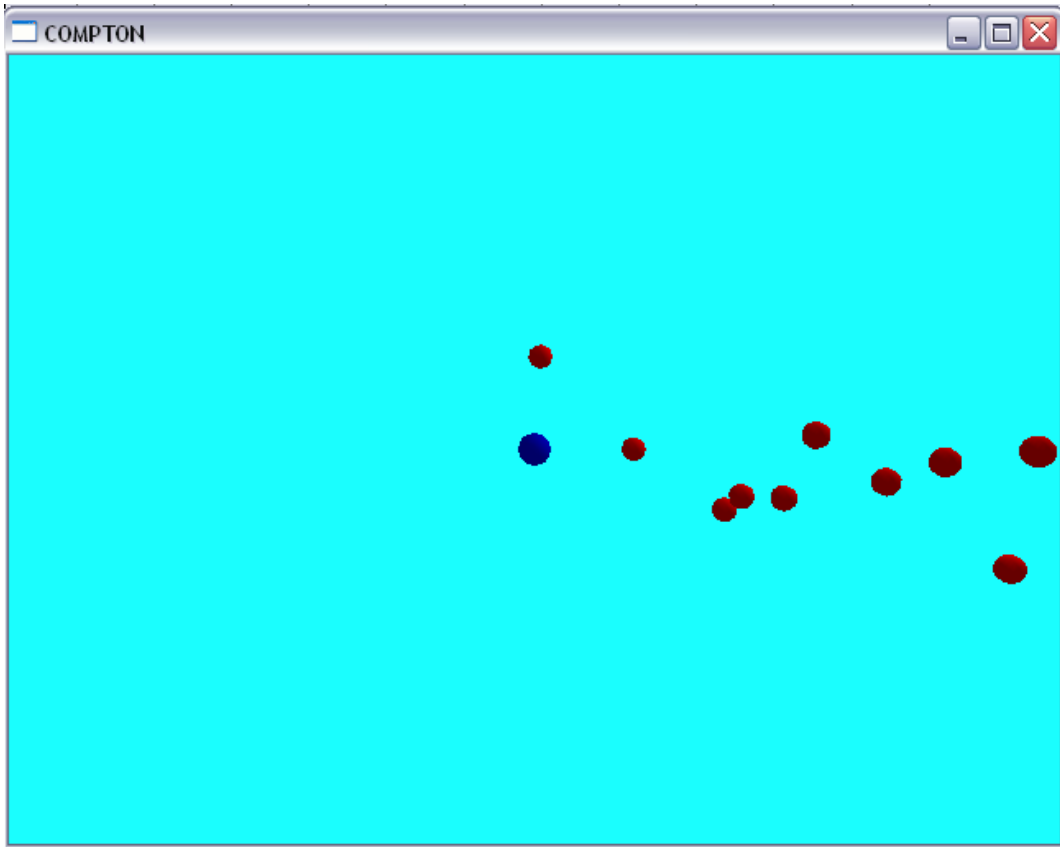
¿Cómo funciona?



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







Codigo

```
#include "openglut.h"
#include <iostream>
#include <fstream>
#include <vector>
#include <cmath>

void DibujandoParticulas();
void luces(void);
void Iniciar(void);
void Reset(void);
void Desplazar(void);
void Leer(std::string nombre);
void RenderScene(void);
void raton(int x, int y);
void idle(void);

long double desp,E,h,w,frec,salto,phi,theta;//VARIABLES DE ANIMACIÓN
float ang;
bool crush=false;
int PARTICULAS=50;
float angX=0.0f,angY=0.0f,escala=1.0f;//VARIABLES DEL PROGRAMA
int ResX=640,ResY=480;
std::string cadena;
enum{EJES=1,PEL1,PEL2};
//Definición del modelo de una luz
GLfloat light_Ambient [4] = { 0.2f, 0.2f, 0.2f, 1.0f};
GLfloat light_Diffuse [4] = { 0.5f, 0.5f, 0.5f, 1.0f};
GLfloat light_Position [4] = { 5.0f, 5.0f, -5.0f, 1.0f};
//Definición de las características opticas del material: coeficientes
dereflexión
GLfloat material [4] = {1.0f, 0.2f, 0.2f, 1.0f };
GLfloat RedMaterial [4] = {1.0f, 0.0f, 0.0f, 1.0f };
GLfloat GreenMaterial [4] = {0.0f, 1.0f, 0.0f, 1.0f };
GLfloat BlueMaterial [4] = {0.0f, 0.0f, 1.0f, 1.0f };
GLfloat WhiteMaterial [4] = {1.0f, 1.0f, 1.0f, 1.0f };
GLfloat BlackMaterial [4] = {0.0f, 0.0f, 0.0f, 1.0f };
GLfloat GrayMaterial [4] = {0.6f, 0.6f, 0.6f, 1.0f };

void DibujandoParticulas(){
glNewList(PEL1,GL_COMPILE);
glMaterialfv(GL_FRONT, GL_AMBIENT_AND_DIFFUSE, RedMaterial );
glutSolidSphere(0.07f,10,10);
glEndList();
glNewList(PEL2,GL_COMPILE);
glMaterialfv(GL_FRONT, GL_AMBIENT_AND_DIFFUSE, BlueMaterial );
glutSolidSphere(0.07f,10,10);
glEndList();
}
void luces(void){
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 );
```

```

}
class Posicion{
public:
long double x,y;
};
class Foton{
public:
bool crush;

Posicion pos;
void Iniciar(void){
Reset();
}
void Reset(void){
crush=false;
pos.x=- (long double)(rand()%1000)/100.0;
pos.y=(long double)(rand()%1000)/1000.0;
}
void Desplazar(void){
if(crush){
pos.x+=cos(theta)*(E*10e19);
pos.y+=sin(theta)*(E*10e19);
}else{
pos.x+=(E*10e19);
} if(pos.x>3.0f || pos.y>2.0f || pos.y<-1.0f){
Reset();
}
};
class Electron{
public:
bool crush;
Posicion pos;
long double ang,d;
void Iniciar(void){
Reset();
}
void Reset(void){
crush=false;
ang=360.0*(long double)(rand()%1000)/1000.0;
d=1-(long double)(rand()%200)/1000.0;
}
void Desplazar(void){
if(crush){
pos.x+=cos(phi)*(E*10e19);
pos.y-=sin(phi)*(E*10e19);
}else{
pos.x=d*sin(this->ang);
pos.y=d*cos(this->ang);
}
ang+=(E*10e19);
};
bool Colision(long double x1,long double y1,long double x2,long double
y2){
if(sqrt(pow(x2-x1,2)+pow(y2-y1,2))<0.1){
return true;
}else{
return false;
}
}

```

```

}
std::vector<Foton> fotones;
std::vector<Electron> electrones;
void Leer(std::string nombre)//Leyendo y calculando
{
std::ifstream archivo(nombre.c_str());
archivo >> salto;
archivo >> phi;
archivo >> theta;
archivo.close();
h=6.63e-34;
frec=1.0/salto;
w=2.0*3.1416*frec;
E=h*w;

std::cout<<E<<std::endl;
desp=-1;
}
void RenderScene(void){
glClear(GL_COLOR_BUFFER_BIT|GL_DEPTH_BUFFER_BIT);
glLoadIdentity( );//AJUSTANDO LA CÁMARA
glPushMatrix();
glTranslatef(-3.3f,2.2f,-5.0f);
glScalef(0.002f,0.002f,0.002f);
glMaterialfv(GL_FRONT, GL_AMBIENT_AND_DIFFUSE, WhiteMaterial );
glPopMatrix();
glTranslatef(0.0f,0.0f,-5.0f);
glRotatef(angY,1.0f,0.0f,0.0f);
glRotatef(angX,0.0f,1.0f,0.0f);
glPolygonMode(GL_FRONT_AND_BACK,GL_FILL);//DIBUJANDO ELEMENTOS
glPushMatrix();
glScalef(escala,escala,escala);
glPushMatrix();//centro
glScalef(1.5,1.5,1.5);
glCallList(PEL2);
glPopMatrix();
for(int i=0;i<PARTICULAS;i++){
glPushMatrix();
glTranslated(crush?1.0:fotones[i].pos.x,fotones[i].pos.y,0);
glRotatef(ang,0.0f,1.0f,0.0f);//Rota sobre su eje
glCallList(PEL1);
glPopMatrix();
}
for(int i=0;i<PARTICULAS;i++){
glPushMatrix();
glTranslated(crush?1.0:electrones[i].pos.x,electrones[i].pos.y,0);
glRotatef(ang,0.0f,1.0f,0.0f);//Rota sobre su eje
glCallList(PEL2);
glPopMatrix();
}
glTranslatef(1.0f,1.0f,1.0f);
glMaterialfv(GL_FRONT, GL_AMBIENT_AND_DIFFUSE, RedMaterial );
glPopMatrix();
glFlush();
glutSwapBuffers();
}
void raton(int x, int y){
angX=(float)x;
angY=(float)y;
}

```

```

void idle(void){
ang+=0.1f;
for(int i=0;i<PARTICULAS;i++){ //colision entre particulas
for(int j=0;j<PARTICULAS;j++){
if(Colision(fotones[i].pos.x,fotones[i].pos.y,electrones[j].pos.x,elec
trones[j
].pos.y)){
fotones[i].crush=true;
electrones[j].crush=true;
}
}
}
for(int i=0;i<PARTICULAS;i++){
fotones[i].Desplazar();
electrones[i].Desplazar();
}
glutPostRedisplay();
}

int main(int a, char *b[]){

glutInit(&a,b);
glutInitDisplayMode(GLUT_RGB | GLUT_DOUBLE | GLUT_DEPTH);
glutInitWindowSize(ResX,ResY);
glutInitWindowPosition(0, 0);
glutCreateWindow("COMPTON");
glClearColor( 0.1f, 1.0f, 1.0f, 1.0f );//COLOR AMARILLO
glutDisplayFunc(RenderScene);
glutPassiveMotionFunc(raton);
glutIdleFunc(idle);
//OPENGL
glViewport( 0, 0, ResX, ResY );
glMatrixMode( GL_PROJECTION );
glLoadIdentity( );
glFrustum(-(float)ResX/ResY, (float)ResX/ResY, -1.0, 1.0, 2, 10000.0);
glMatrixMode( GL_MODELVIEW );
glLoadIdentity( );
glEnable(GL_DEPTH_TEST);
glEnable(GL_LIGHTING);
glEnable(GL_LIGHT0);
glShadeModel(GL_SMOOTH);
Leer("files.txt");//lectura desde archivo
DibujandoParticulas();
fotones.resize(PARTICULAS);
electrones.resize(PARTICULAS);
for(int i=0;i<PARTICULAS;i++){
fotones[i].Iniciar();
electrones[i].Iniciar();
}
luces();
glutMainLoop();
return 0;
}

```

Conclusion

En este proyecto se hace lo que se hizo en el proyecto uno pero con varias partículas la introducción de los datos se realiza de una archivo de texto con los tres datos además que usamos los conocimientos que fuimos adquiriendo a lo largo del semestre