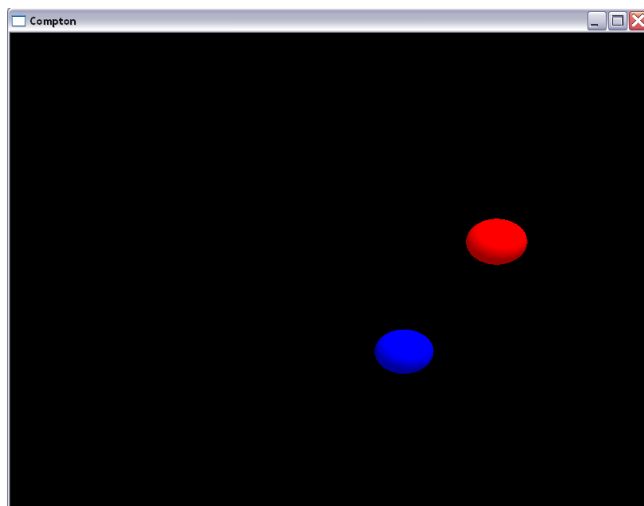
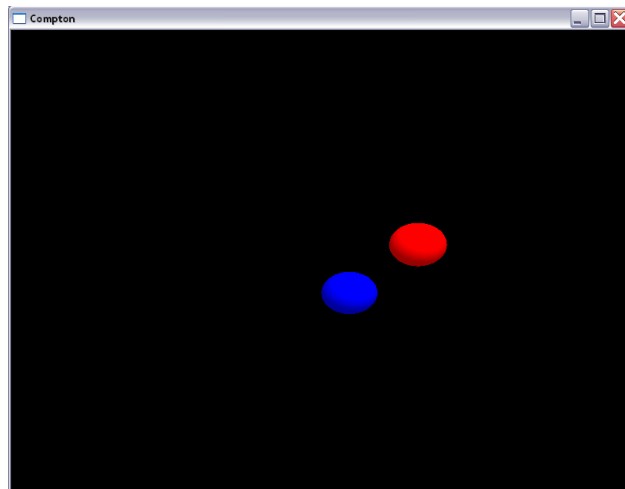
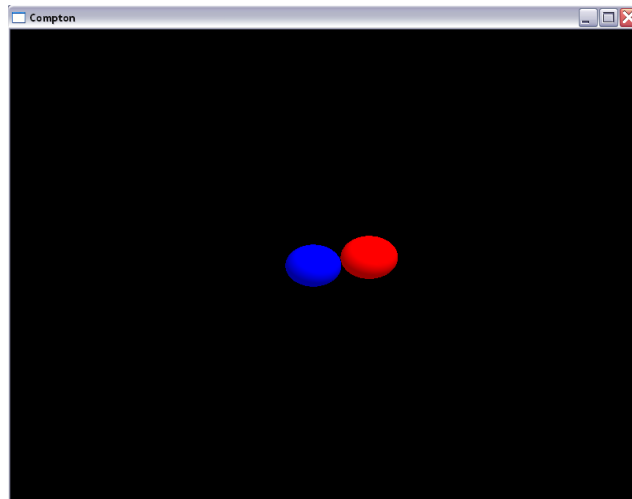




**Examen : Efecto Compton**

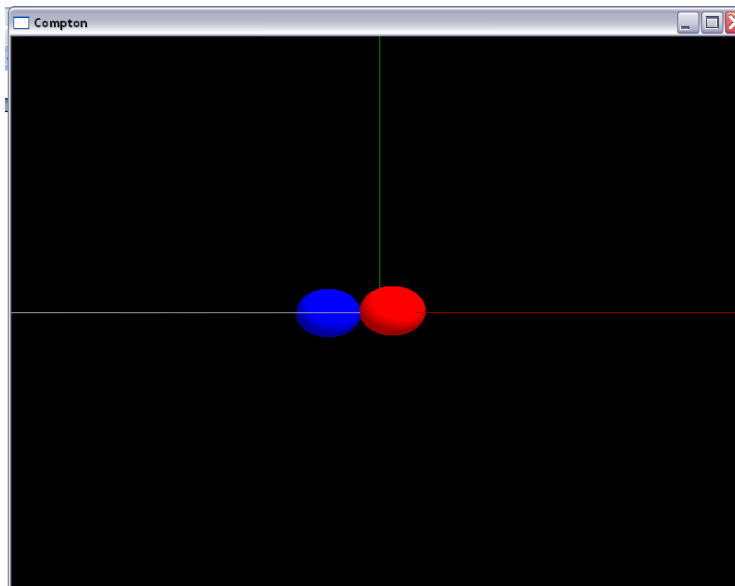
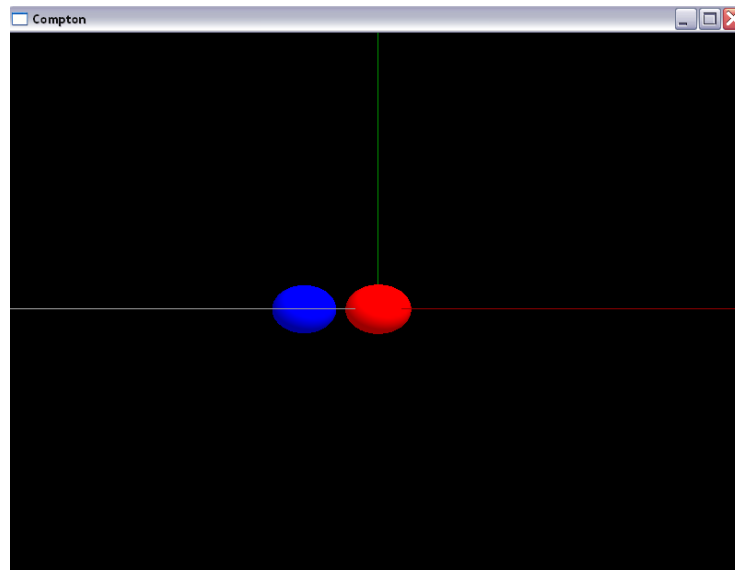
---



Examen : Efecto Compton

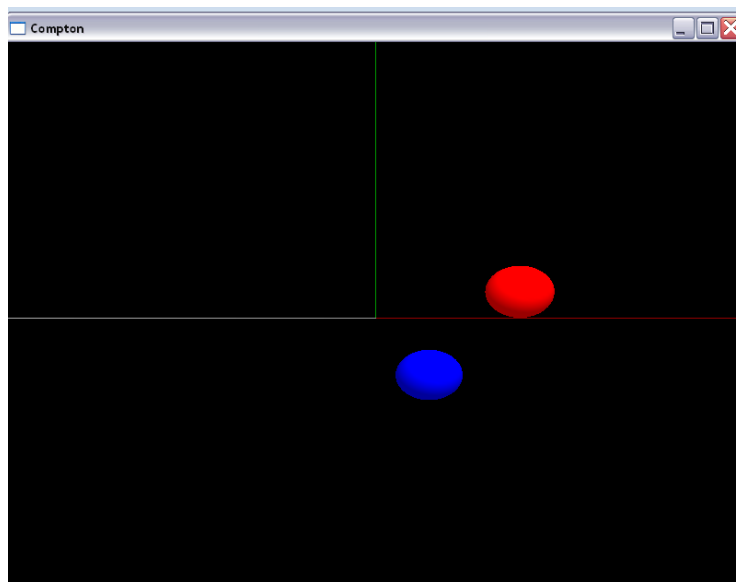
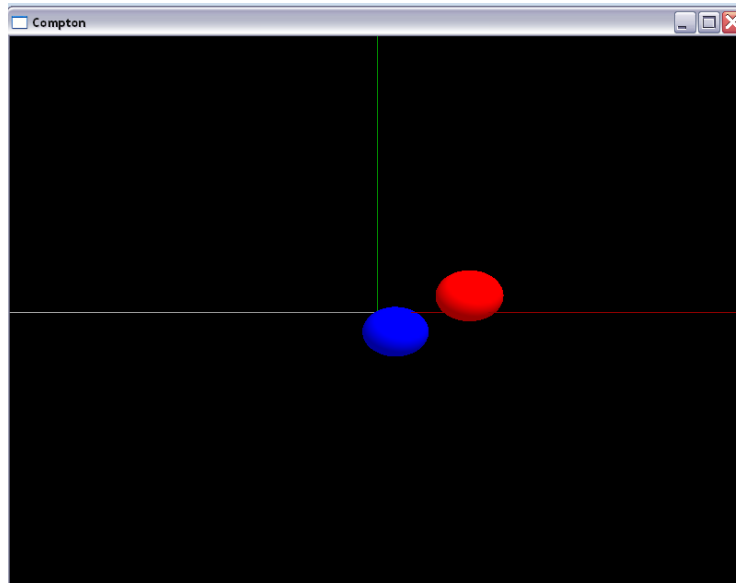
---

Con ejes de referencia:



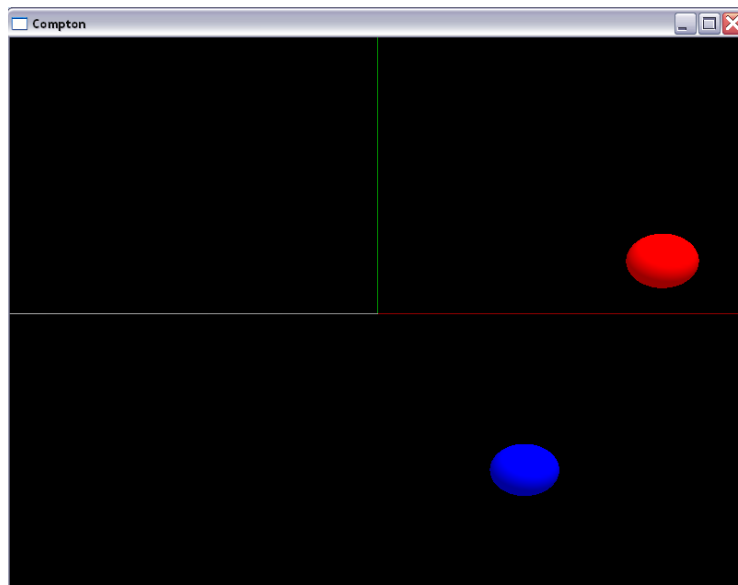
Examen : Efecto Compton

---



Examen : Efecto Compton

---



**Examen : Efecto Compton**

---

**Código:**

```
#include <gl/openglut.h>
#include <math.h>
#include <fstream>
#include <string>
using std::ifstream;

//Se declara un cuadrics para generar las esferas
GLUquadricObj *quadobj, *discobj;

//Declaración de variables necesarias para el funcionamiento del programa
long double h=6.63E-34;//Declaración de la cte h
long double w=0, E=0;//Declaracion de las variables frecuencia y energía
long double desf=-4, dese=0;
long double LO;
long double foton;
long double electron;
long double Afoton=0, Aelectron=0;

//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 };

/* los valores de los materiales se van a sumar para tener una totalidad,
que antes se multiplico a la luz
ambiente+difuso + especular*/

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 );
}

//Función que define por medio de listas un quadrics de tipo esfera
void pelota()
{
glNewList(1, GL_COMPILE);

quadobj = gluNewQuadric();
```

**Examen : Efecto Compton**

---

```
        gluQuadricDrawStyle( quadobj, GLU_FILL);
        gluQuadricNormals( quadobj, GLU_SMOOTH);

        gluSphere(quadobj, 0.5, 50, 50);

        gluDeleteQuadric(quadobj);
    glEndList();
}

//Función que nos permite dar movimiento al fotón y al electrón de la
animación
void idle(void)
{

    glutPostRedisplay();

    desf=desf+E;
    if (desf>=-1.4)
    {

        if (desf>=0)
            Afoton=foton;

        Aelectron=electron;
        dese=dese+E;

        if(desf>=5)
        {
            desf=-4;
            Afoton=0;
            dese=0;
        }
    }
}

//Función que lee los datos que ingresemos en el archivo datos.txt
void LeerArchivo()
{
    char LOF[40];
    char fotonf[40];
    char electronf[40];

    ifstream texto("datos.txt");

        //Variables que guardan los datos de longitud de onda,
ángulo del fotón y ángulo del neutrón
        texto >> LOF;
        texto >> fotonf;
        texto >> electronf;

//Cerramos el archivo
    texto.close();
}
```

**Examen : Efecto Compton**

---

```

//Convertimos las variables obtenidas de char a long double
LO=atof(LOf)*1e-10;
foton=atof(fotonf);
electron=atof(electronf);

//Definición de la frecuencia y la energía
w=3.1416*2*(1/LO);
E=h*w;
}

void EjesReferencia()
{
    glBegin (GL_LINES);
    glMaterialfv(GL_FRONT, GL_AMBIENT_AND_DIFFUSE, RedMaterial );
    glVertex3f ( -20.0, 0.0, 0.0);
    glVertex3f (20.0, 0.0, 0.0); //eje x

    glMaterialfv(GL_FRONT, GL_AMBIENT_AND_DIFFUSE, GreenMaterial );
    glVertex3f ( 0.0, 0.0, 0.0);
    glVertex3f ( 0.0,20.0, 0.0); //eje y

    glMaterialfv(GL_FRONT, GL_AMBIENT_AND_DIFFUSE, WhiteMaterial );
    glVertex3f ( 0.0, 0.0, 0.0);
    glVertex3f ( 0.0, 0.0,20.0); //eje z: profundidad
    glEnd();
}

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

    EjesReferencia();
    glMaterialfv(GL_FRONT, GL_AMBIENT_AND_DIFFUSE, RedMaterial
);
    LeerArchivo();
    //En este lugar se declara el electrón por medio de listas y
se le asigna el color
    glMaterialfv(GL_FRONT, GL_AMBIENT_AND_DIFFUSE, RedMaterial );
    glMatrixMode(GL_MODELVIEW);
    glPushMatrix();
    glRotatef(Aelectron,0.0,0.0,1.0); //Rotación que hará el
electrón
    glTranslatef(dese,0,0); //Desplazamiento del electrón
    glCallList(1);
    glPopMatrix();
    //En este lugar se declara el fotón por medio de listas y se
le asigna el color
    glMaterialfv(GL_FRONT, GL_AMBIENT_AND_DIFFUSE, BlueMaterial );

    glMatrixMode(GL_MODELVIEW);
    glPushMatrix();
    glRotatef(Afoton,0.0,0.0,1.0); //Rotación que hará el fotón
    glTranslatef(desf,0,0); //Desplazamiento del fotón

```

**Examen : Efecto Compton**

---

```
        glCallList(1);
        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(800,600);
    glutInitWindowPosition(300,100);
    IdeWindow=glutCreateWindow("Compton");
    glutDisplayFunc(RenderScene);
    glutIdleFunc( idle );

    glClearColor(0.0f, 0.0f, 0.0f, 0.0f);

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

    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    gluPerspective(15,800/600,.1,1000);
    glMatrixMode(GL_MODELVIEW);
    glLoadIdentity();
    gluLookAt(30,0,30,
              0,0,0, // si pones 5,5,-5 se ve cortado
              0,1,0);

    /*Prepara la luz*/
    luces();

    /*Prepara las funciones que van a ser ejecutadas*/
    pelota();
    EjesReferencia();
    LeerArchivo();

    glutMainLoop();

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

    return 0;
}
```

**Examen : Efecto Compton**

---

**Conclusiones:** Con la realización de este examen se dio un repaso general a cada una de las prácticas realizadas en el laboratorio ya que se aplicó el uso de quadrics, listas, animación, uso de archivos, rotación, traslación, push y pop, etc., además se realizó un programa más dinámico pues desde un archivo txt podemos manipular la funcionalidad de nuestra animación.

Lo complicado del proyecto fue el manejo de las variables ya que como la mayoría de los datos que se obtenían eran muy pequeños al principio no se veía reflejado el resultado en la animación, o la computadora tenía volcado de memoria, pero con la programación correcta ese error se solucionó.