Cadena Ramírez Miguel Ángel

Grupo 4

Practica 3

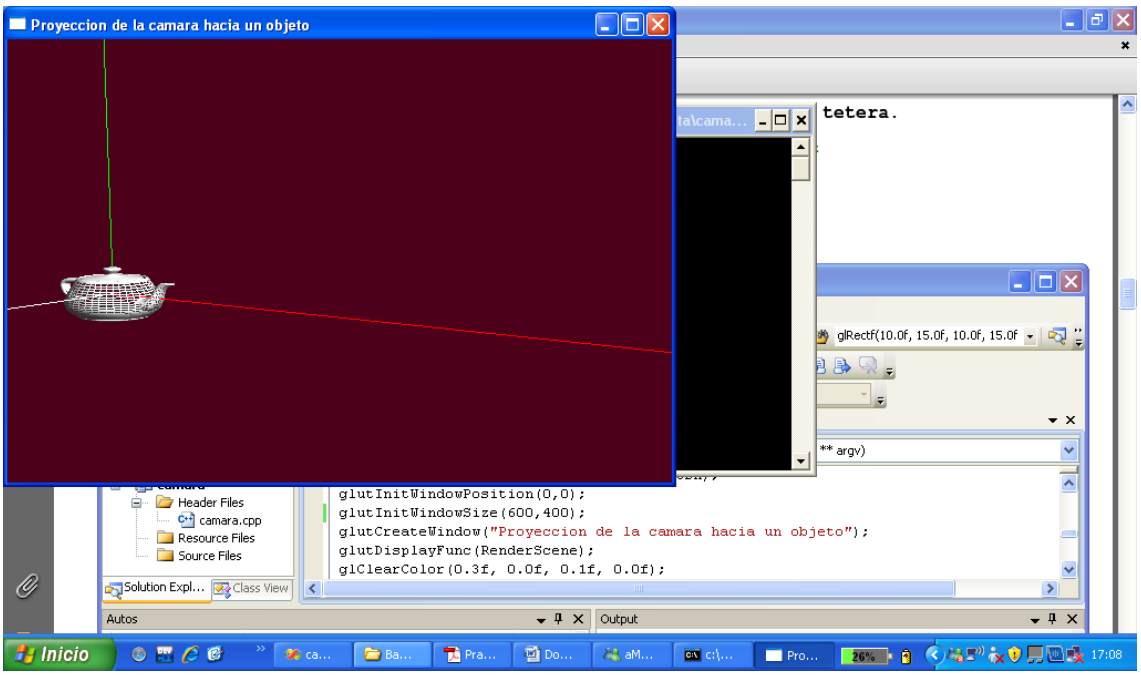
"Cámara"

Objetivo: conocer e utilizar adecuadamente la cámara, mediante gluLookAt.

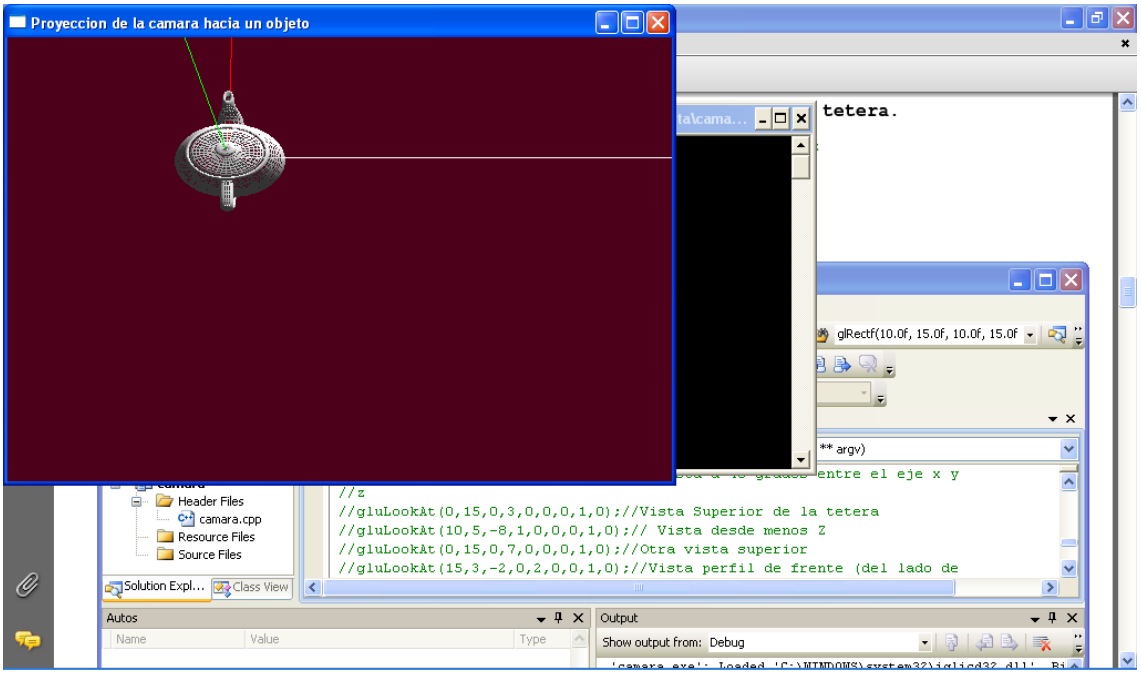
Su forma de expresarse es:
`void` **gluLookAt**(GLdouble eyeX,
GLdouble eyeY,
GLdouble eyeZ,
GLdouble centerX,
GLdouble centerY,
GLdouble centerZ,
GLdouble upX,
GLdouble upY,
GLdouble upZ);
Parámetros
eyeX, *eyeY*, *eyeZ* Posición de punto de vista (desde donde queremos observar)
centerX, *centerY*, *centerZ* Posición del punto de referencia (hacia donde apuntamos)
upX, *upY*, *upZ* Vector de posición (en "x", en "y" o en "z")

Probe todas y cada una de las vistas de ejemplo de la practica para asi darme una idea de cómo utilizar la cámara en opengl lo que note es que nuestro punto de referencia debe estar muy próximo a nuestro objeto que queramos enfocar y este punto no es mas que las proyecciones escalares en la dirección de los ejes coordenados x, y y z del vector de posición de nuestro punto de referencia, también las coordenadas del vector de punto de vista coincide con las componentes del vector de posición de este punto.

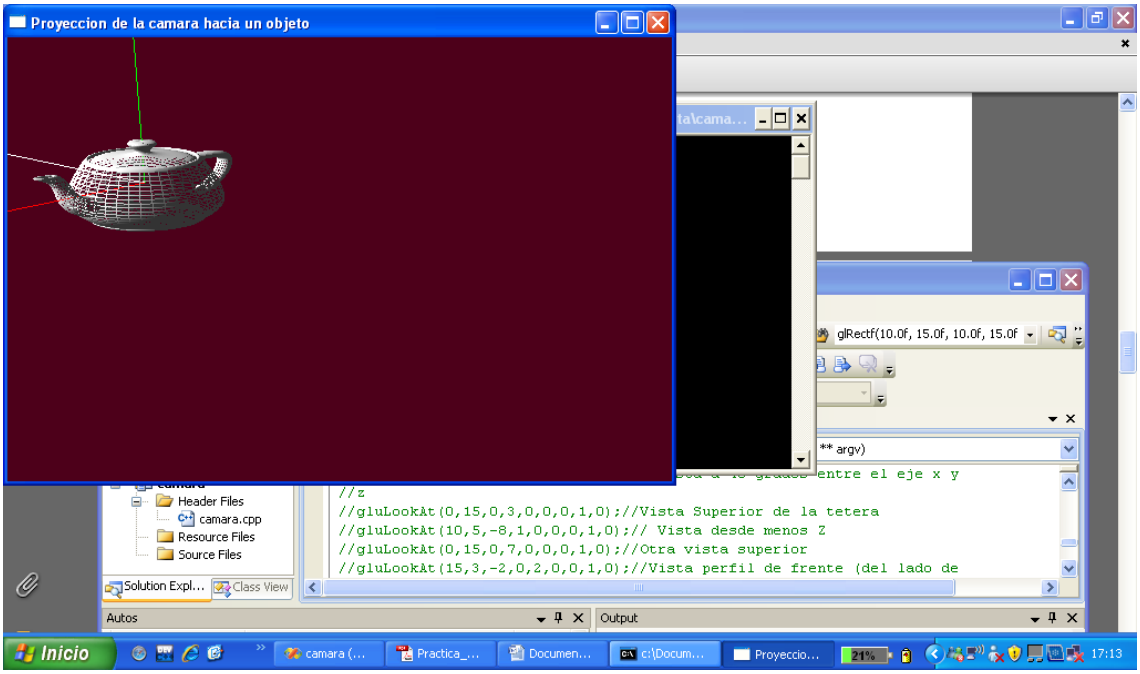
```
gluLookAt(15,5,10,4,2,-5,0,1,0);//vista a 45 grados entre el eje x y z
```



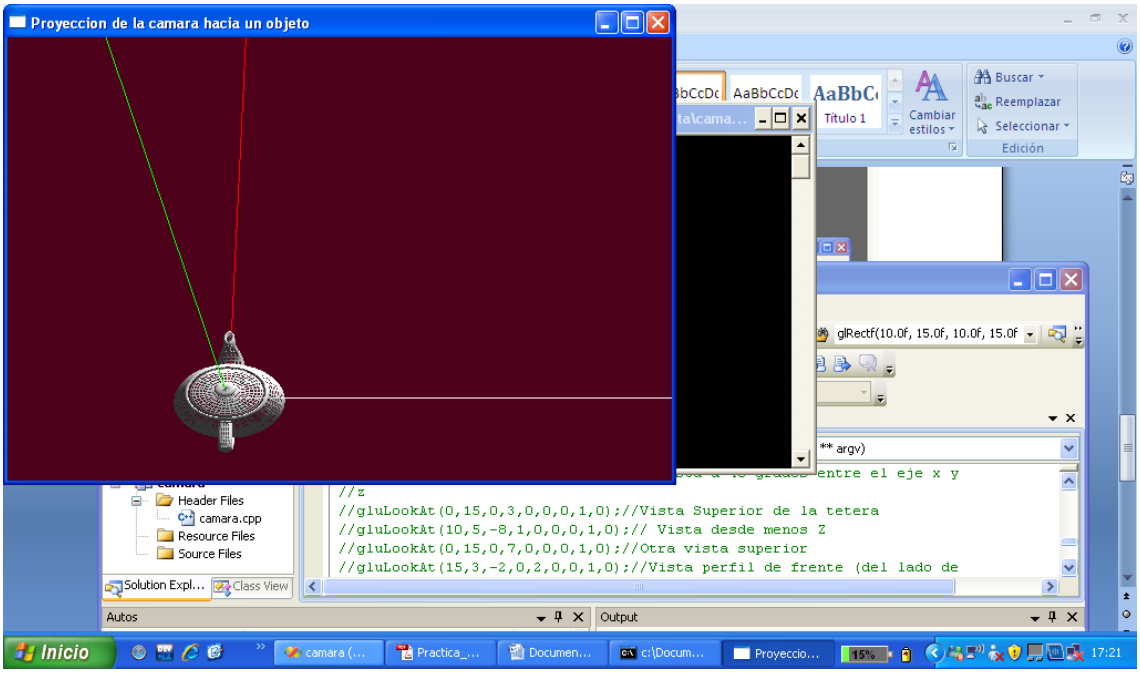

```
gluLookAt(0,15,0,3,0,0,0,1,0);//Vista Superior de la tetera
```



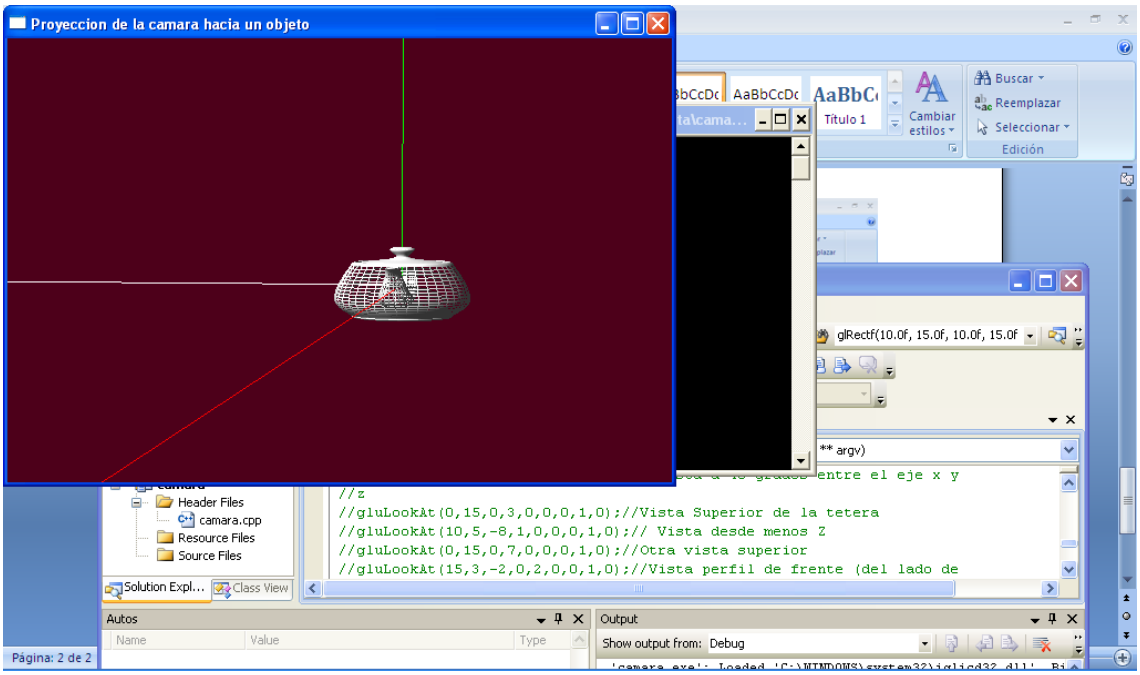

```
gluLookAt(10,5,-8,1,0,0,0,1,0);// Vista desde menos Z
```



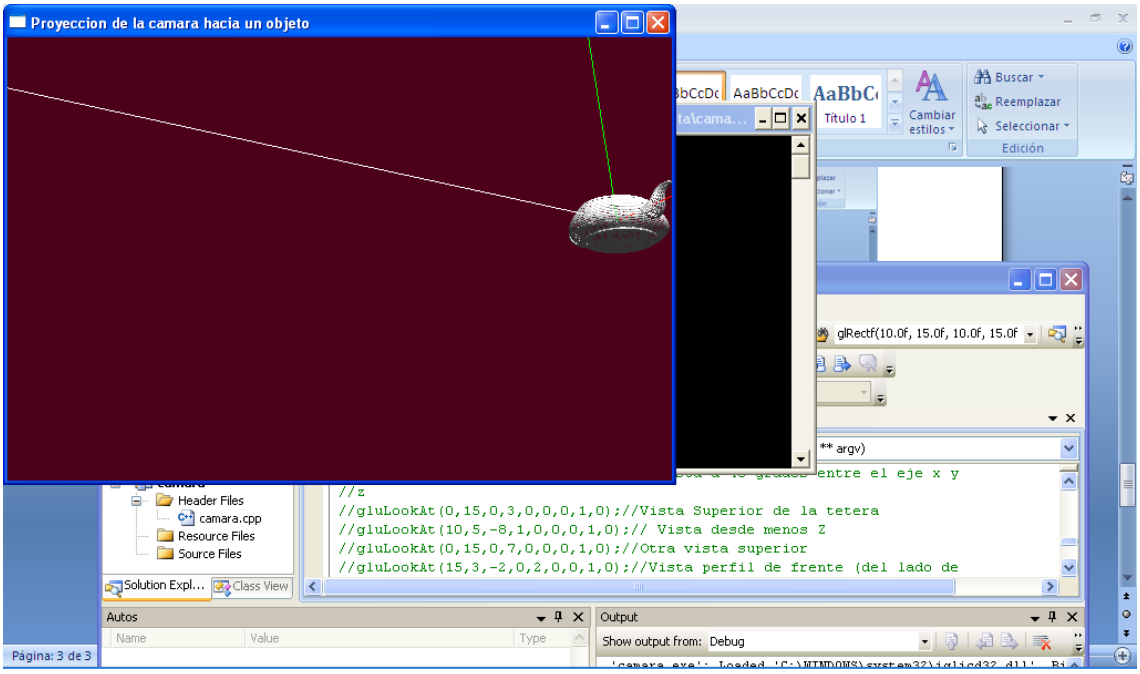
`gluLookAt(0,15,0,7,0,0,0,1,0);//Otra vista superior` hacia el eje x



```
gluLookAt(15,3,-2,0,2,0,0,1,0);//Vista perfil de frente (del lado de la
escurridera de la tetera)
```

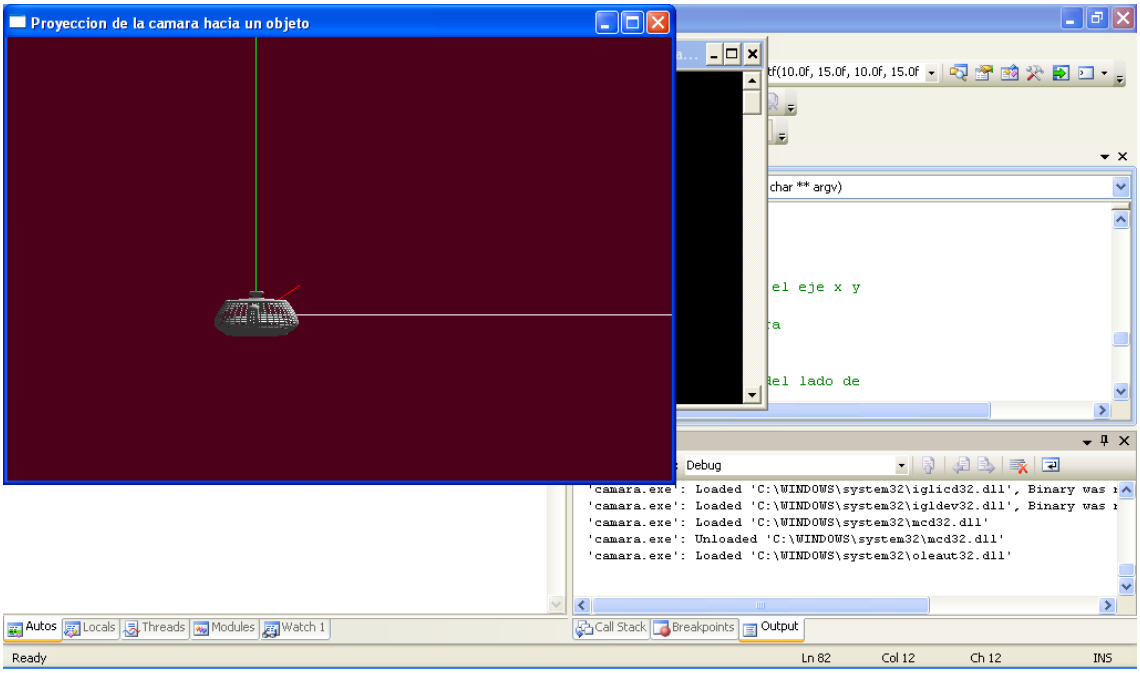



```
gluLookAt(14,-6,8,0,3,0,0,1,0); //Este LookAt nos deja ver la tetera por
debajo. Con mayor luz.
```



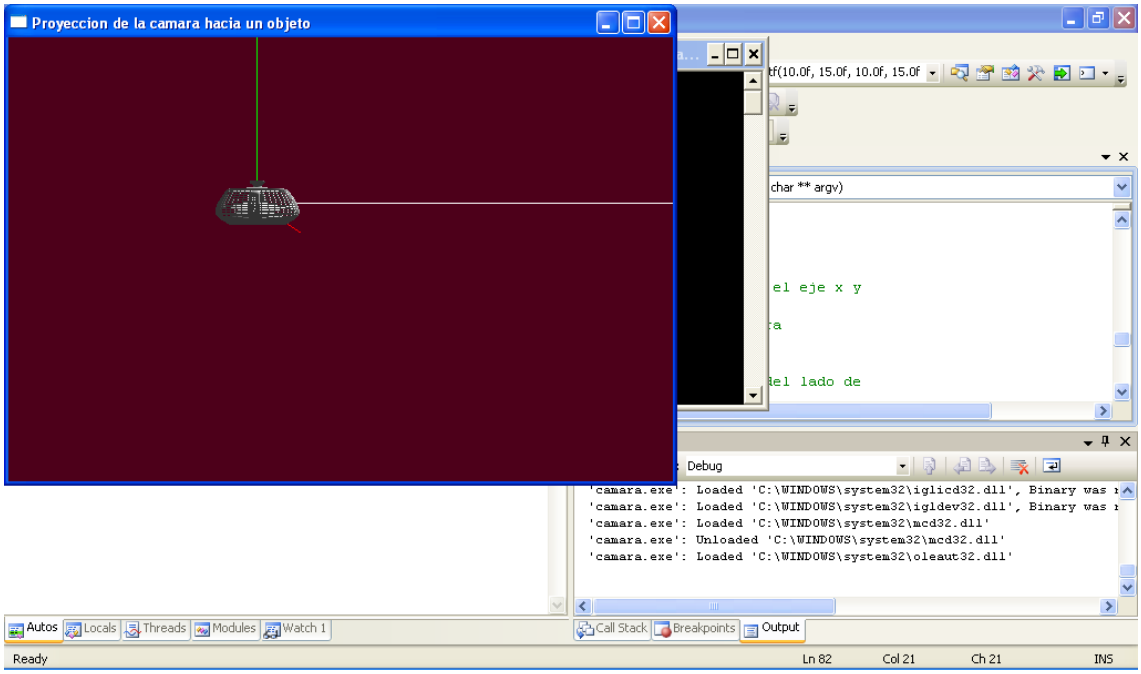
En esta parte de la practica mi punto de vista le di como coordenadas la terna ordenada siguiente -14,3,0 y mi punto de referencia tiene como coordenadas 0,3,0 esto en código es lo siguiente.

```
gluLookAt(-14,3,0,0,3,0,0,1,0);
```



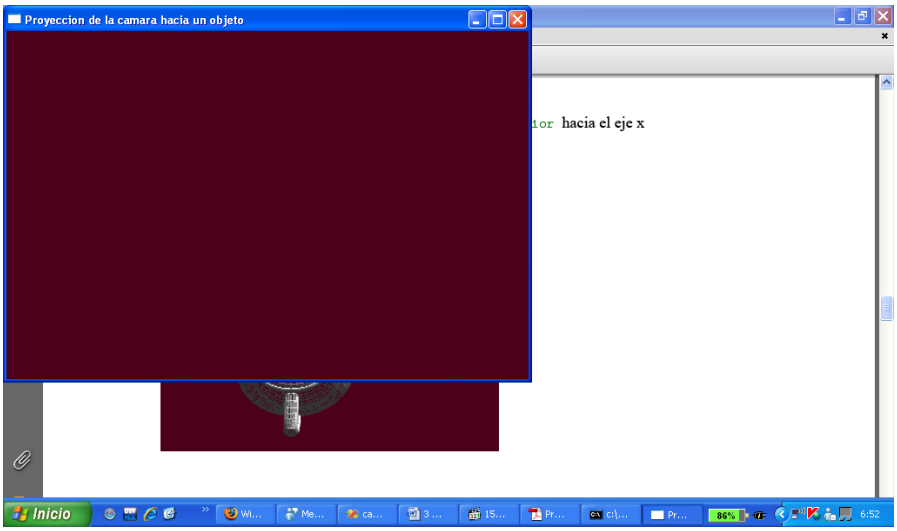
En esta parte de la practica mi punto de vista le di como coordenadas la terna ordenada siguiente -14,1,0 y mi punto de referencia tiene como coordenadas -9,1,0 esto en código es lo siguiente.

```
gluLookAt(-14,1,0,-9,1,0,0,1,0);
```



Para lo especificado en la practica para que mi punto de vista sea desde arriba no se ve el objeto en la pantalla ya que el objeto esta siempre en el origen de nuestro sistema de referencia para poder ver nuestro objeto nuestro punto de referencia debe de estar lo mas proximo a nuestro objeto. Yo probe con la siguiente configuracion pero no se observa el objeto

```
gluLookAt(-5,1,0,-10,3,0,0,1,0);
```



“Luces”

1. Vamos a crear un nuevo proyecto dentro de la carpeta "prácticas_gl". Asi que presione en secuencia menú "**File->New Project**" o simplemente presione simultáneamente las teclas "**ctrl+shift+N**". El nuevo proyecto se llamará "ejer04".

2. Localice el archivo el archivo "**ejemplo_luces_1.cpp**" y cópielo de su origen a la carpeta "…**prácticas_gl\ejer04**".

3. Presione la secuencia menú "**Project->Add Existing_Item**" o simplemente presione la combinación de teclas "**Shift+Alt+A**".

4. El programa consta de una sección de

    a. Variables globales para definir luces y materiales

    b. Subrutina "luces()" que habilita las ecuaciones de luz de oOpenGL

    c. Subrutina "EjesReferencia()" que dibuja los eje4s del sistema de referencia.

    d. Subrutina "Cilindro()" que dibuja un cilindro (es posible aún no entienda el código".

    e. Subrutina "RenderScene()" que ordena invoca a las subrutinas anteriores para dibujar la escena.

    f. Subrutina "main()" que prepara la ventana Windows para OpenGL.

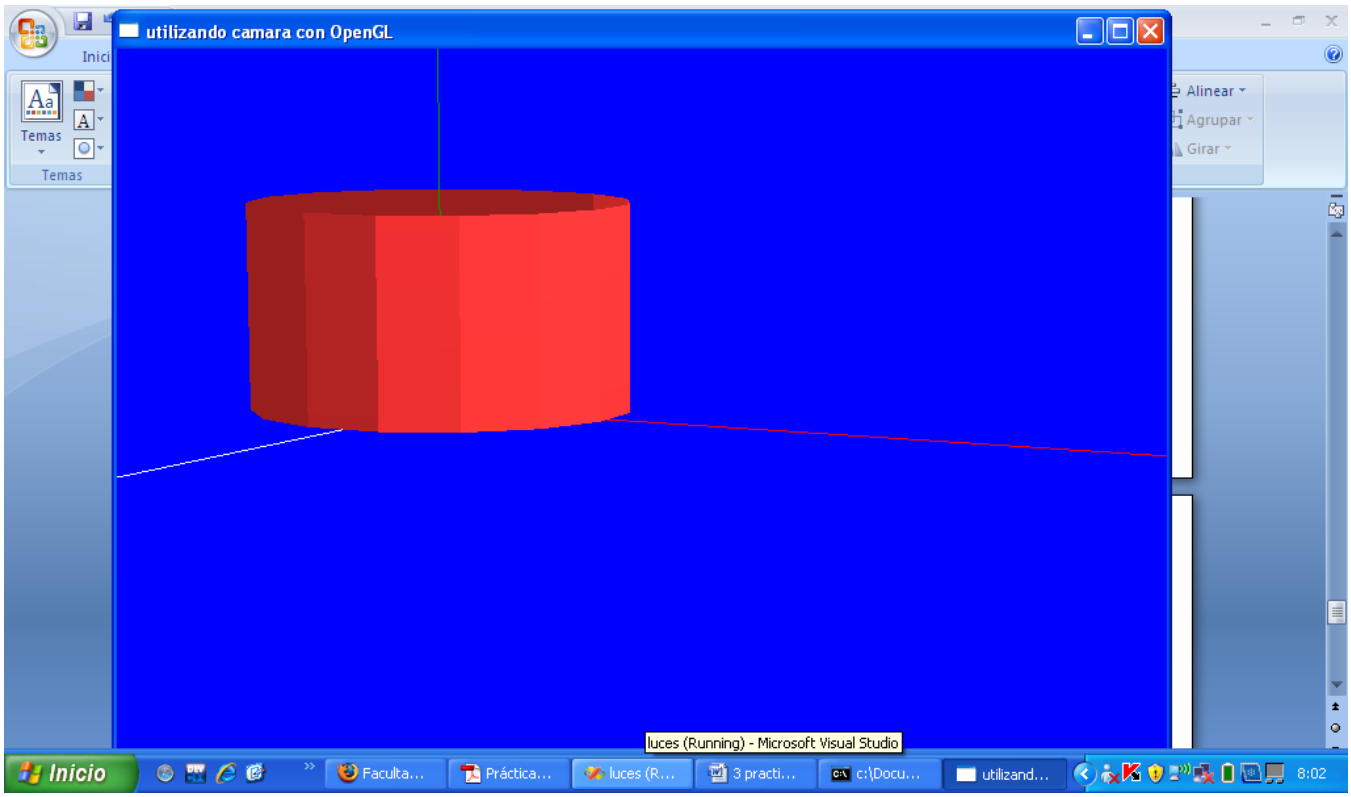
5. Construya y ejecute.

6. Reporte la gráfica obtenida

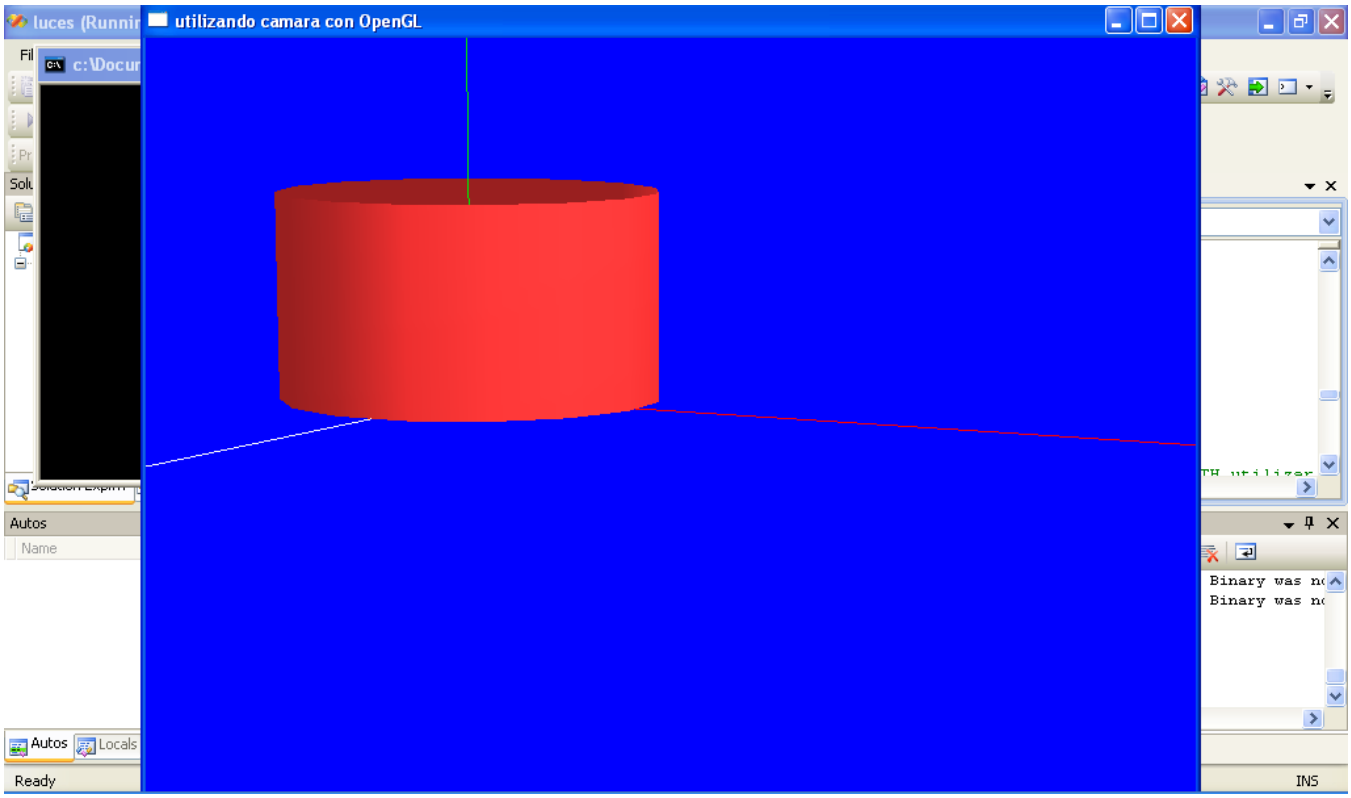
7. Localice dentro de la rutina "RenderScene()" la línea "glShadeModel (GL_FLAT)" y cambie "GL_FLAT" por "GL_SMOOTH". Construya y ejecute.

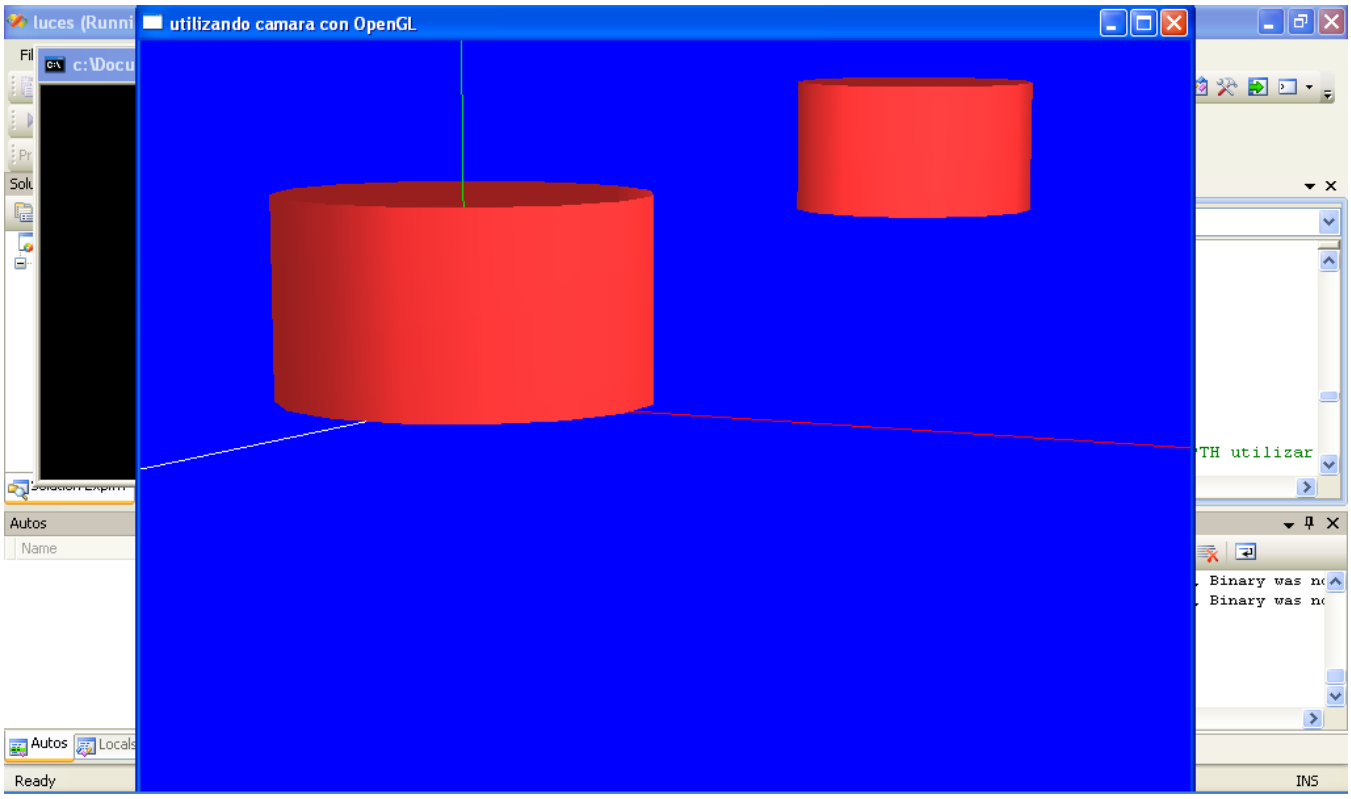
8. Reporte la figura obtenida.

Primero cree mi proyecto al cual llame luces, compile el programa sin cambios presionando f5 y la salida fue la siguiente.
De esta forma nuestro objeto está recubierto por varios rectángulos

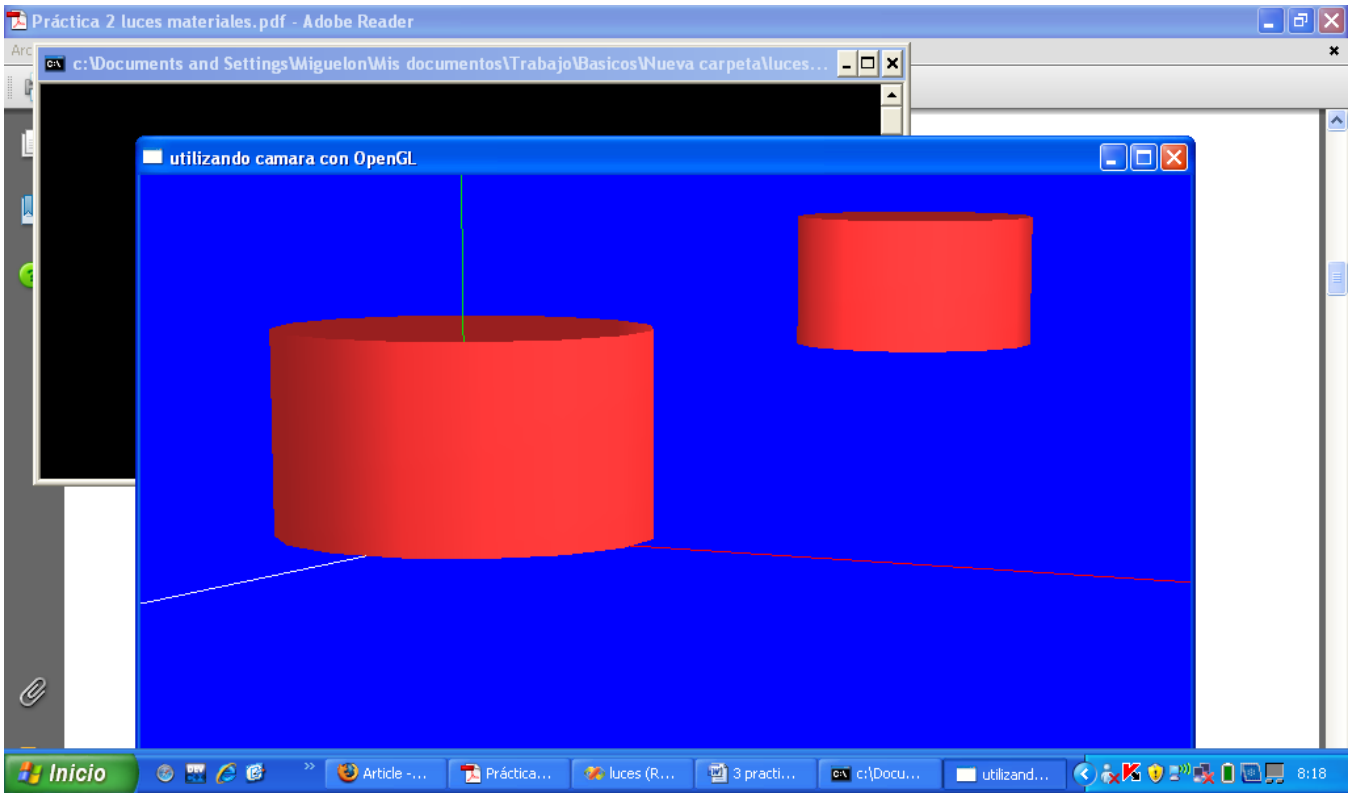


Haciendo la modificación pedida la salida es la siguiente y esta recubierta por una superficie lisa

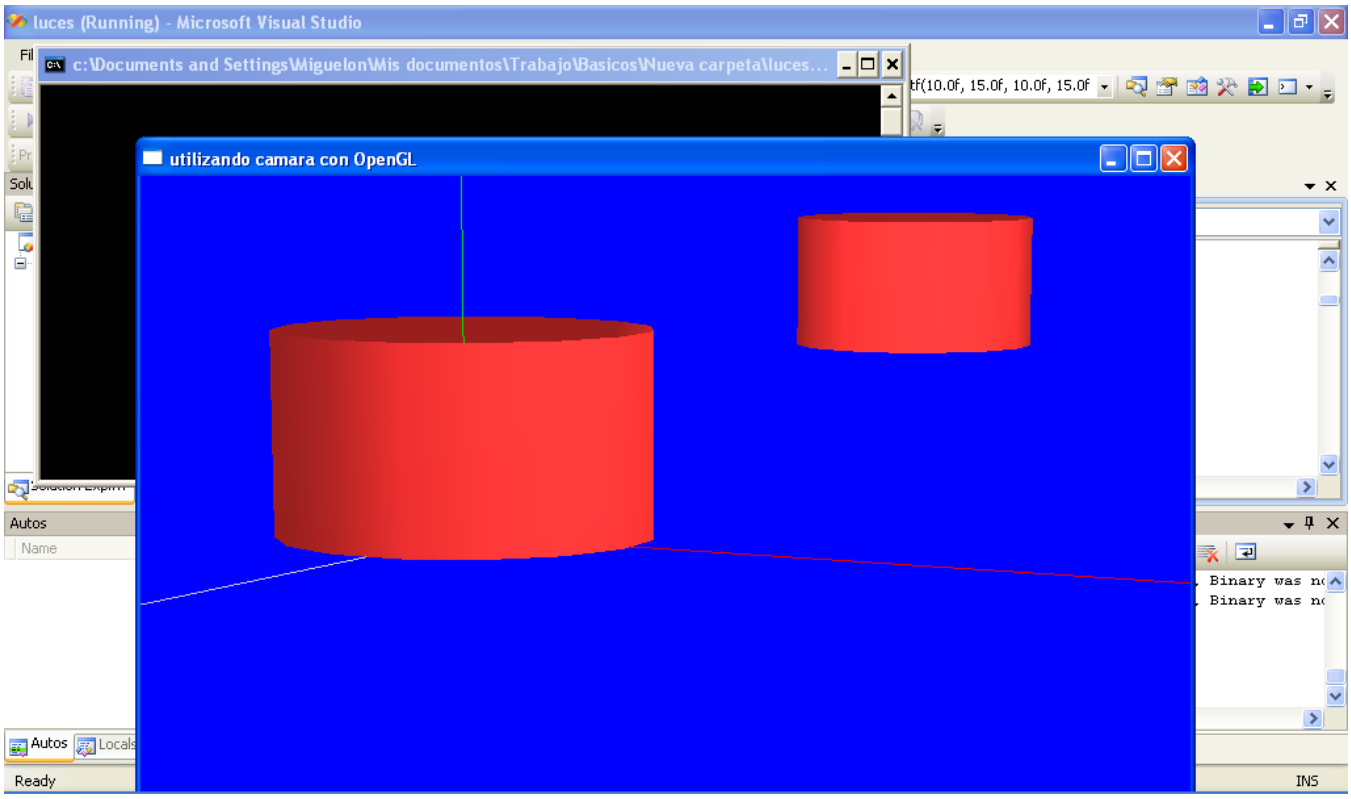




1 Probe el programa agregando la línea `glEnable (GL_CONSTANT_ATTENUATION);` y la salida fue la siguiente.



2 Modifique el p´rograma y agregue la línea siguiente `glEnable (GL_LINEAR_ATTENUATION);` y el resultado de la compilacion fue el siguiente:



3

Probe con esta linea y el programa no me compile pero no se por que glEnable(GL_QUADRIC_ATTENUATION);

No se me hizo muy clara la diferencia hay entre `glEnable (GL_CONSTANT_ATTENUATION);` y `glEnable (GL_LINEAR_ATTENUATION);` las impresiones de pantalla las veo iguales.

"Listas"

Ejercicio 1

1. Vamos a crear un nuevo proyecto dentro de la carpeta "prácticas_gl". Así que presione en secuencia menú "**File->New Project**" o simplemente presione simultáneamente las teclas "**ctrl+shift+N**". El nuevo proyecto se llamará "**ejer05**".

2. Localice el archivo el archivo "**ejemplo_listas_01.cpp**" y cópielo de su origen a la carpeta "…**prácticas_gl\ejer05**".

3. Presione la secuencia menú "**Project->Add Existing_Item**" o simplemente presione la combinación de teclas "**Shift+Alt+A**".

4. El programa consta de las siguientes partes

    a. Sección de variables globales.

    b. Subrutina "luces()" para habilitar la ecuación de iluminación de OpenGL.

    c. Subrutina "EjesReferencia()" para dibujar los ejes del sistema de referencia de OpenGL.habilitar

    d. Subrutina "cubo()" Par a dibujar un cubo centrado en el origen de coordenadas.

    e. Subrutina "RenderScene()" Para invocar las subrutinas anteriores.

    f. Subrutina "main()" Para crear una ventana Windows para us o con OpenGL.

5. A su vez,la subrutina "RenderScene" tiene tres secciones en desorden correspondientes a los ejercicios de estas práctica. Las secciones son:

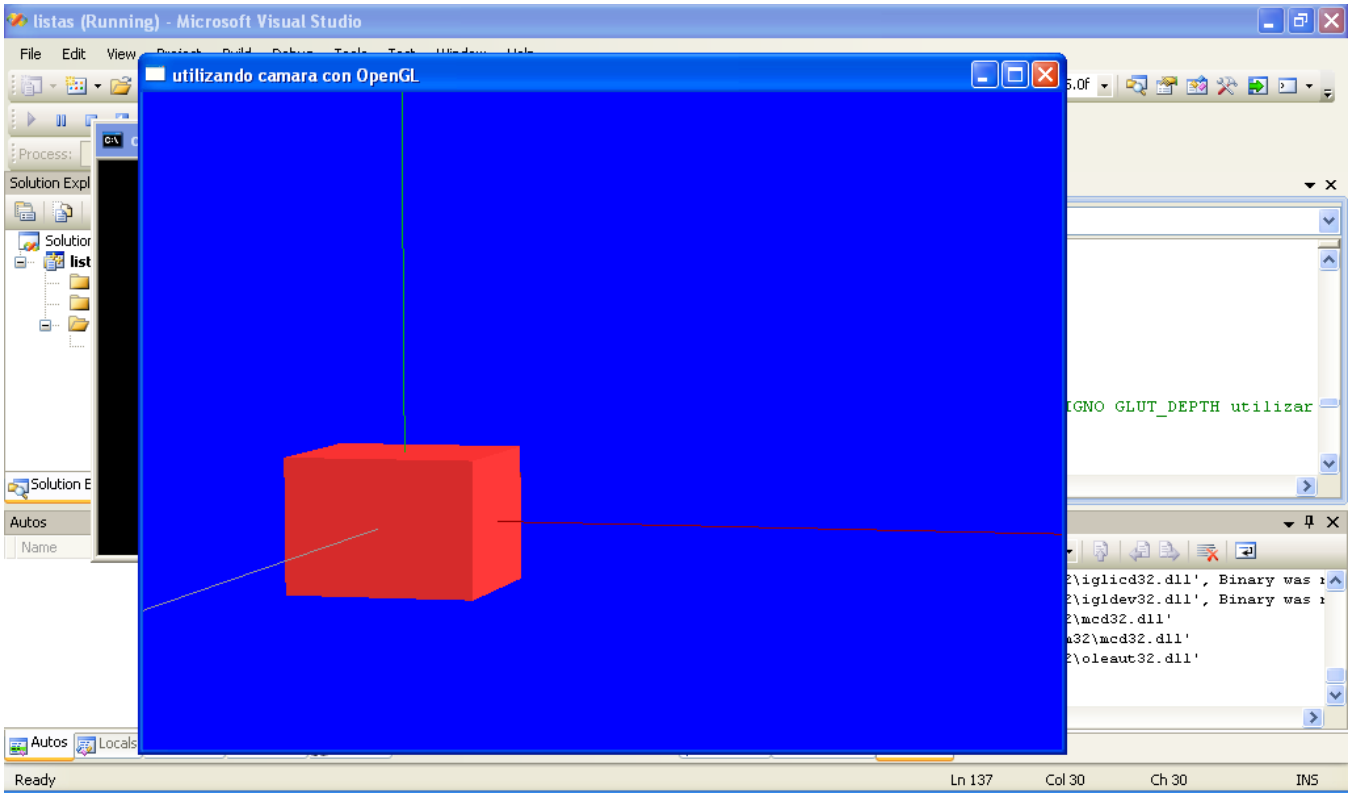
    a. Ejercicio 3

    b. Ejercicio 1

    c. Ejercicio 2

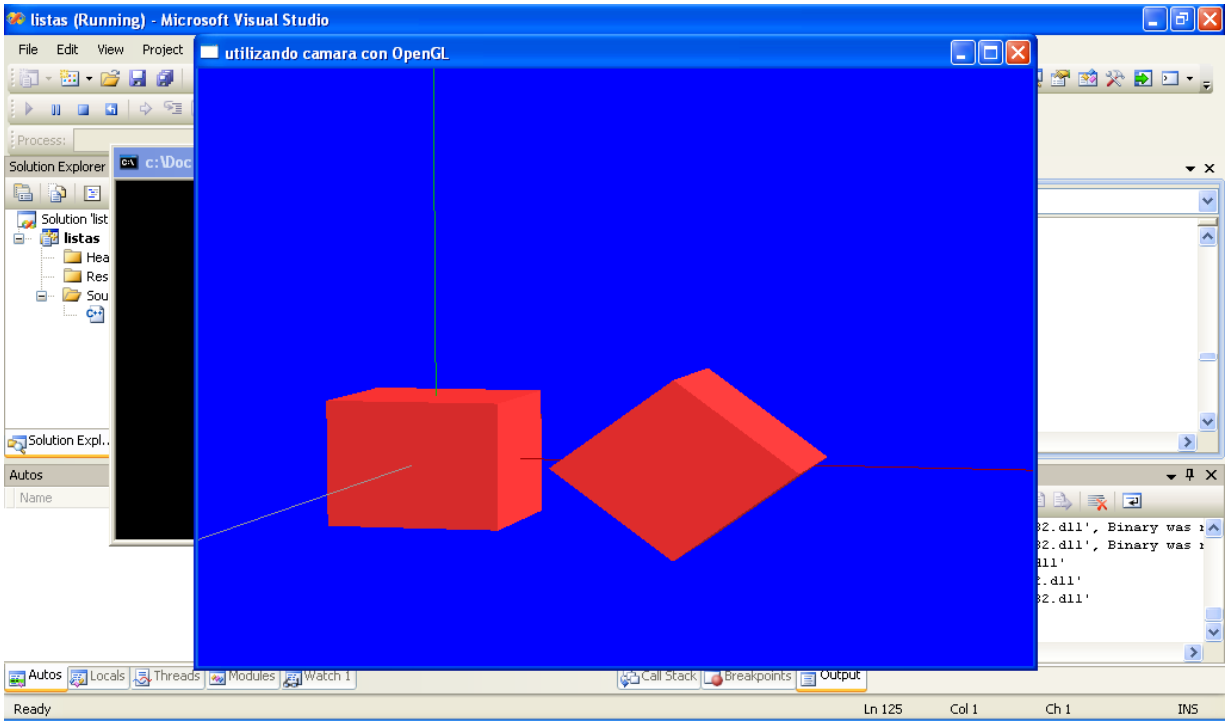
6. Construya y ejecute.

7. Reporte la gráfica obtenida



Ejercicio 2

Aquí descomente la parte que dice ejercicio 2 y la compilacion fue la siguente

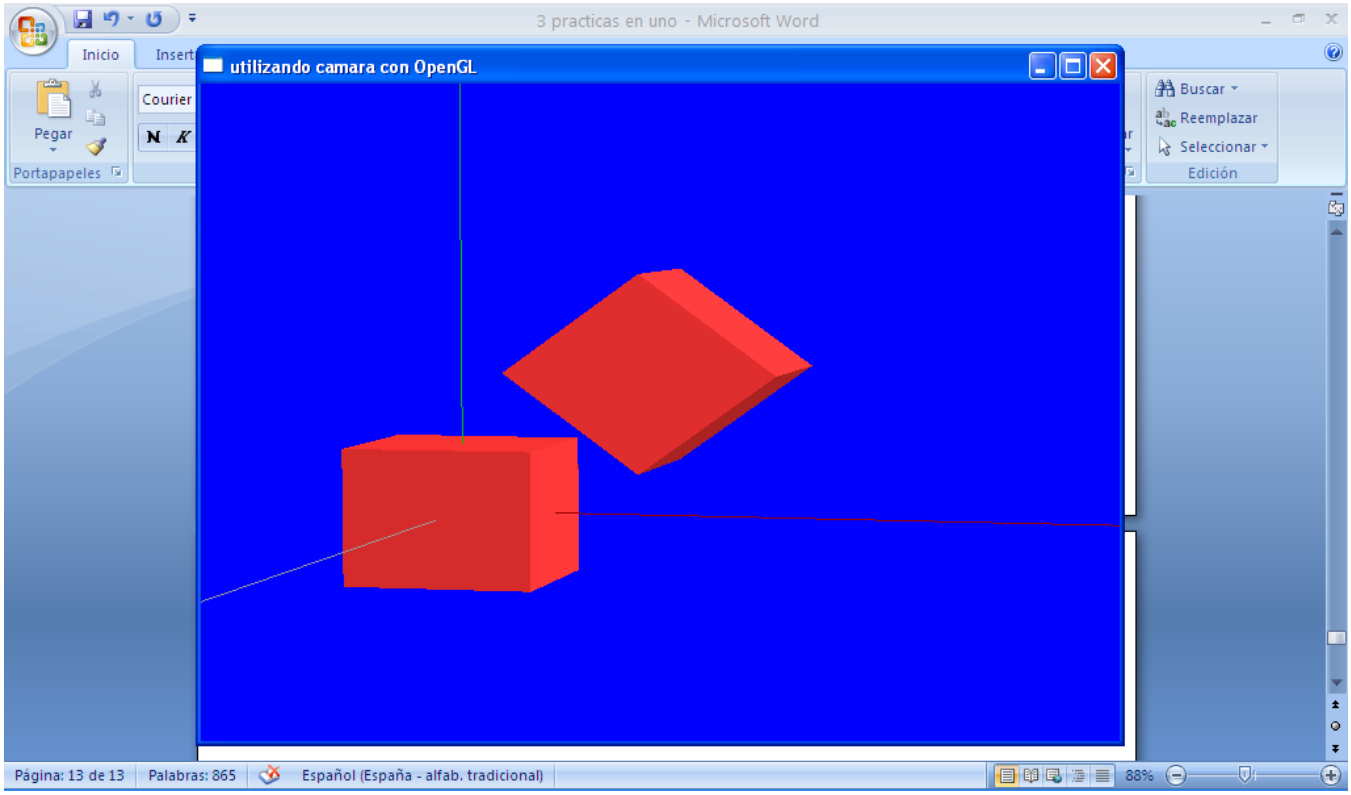


```
//glRotatef(45,0.0,0.0,1.0);
    glTranslatef(3,0,0);
   glRotatef(45,0.0,0.0,1.0);
```

Aquí inverti las lineas y el resultado de la compilacion fue el siguiente

```
glRotatef(45,0.0,0.0,1.0);
    glTranslatef(3,0,0);
 //glRotatef(45,0.0,0.0,1.0);
```

Aquí primero se roto alrededor del eje de las z (cotas) 45 grados y despues se translado al punto 3,0,0



El orden en que se ponen las instrucciones en nuestro programa es importante ya que la instrucción que esta primero siempre tendra preferencia aquí se traslado el cubo y luego se roto 45

Ejercicio 3

Para este ejercicio debe dejar el código correspondiente al ejercicio 2 sin comentarios, con lo cual podrá observar la secuencia de instrucciones que se requieren para construir una escena.

14. Dentro de la subrutina "RenderScene()" se hay código entre comentarios titulado "Ejercicio_3".

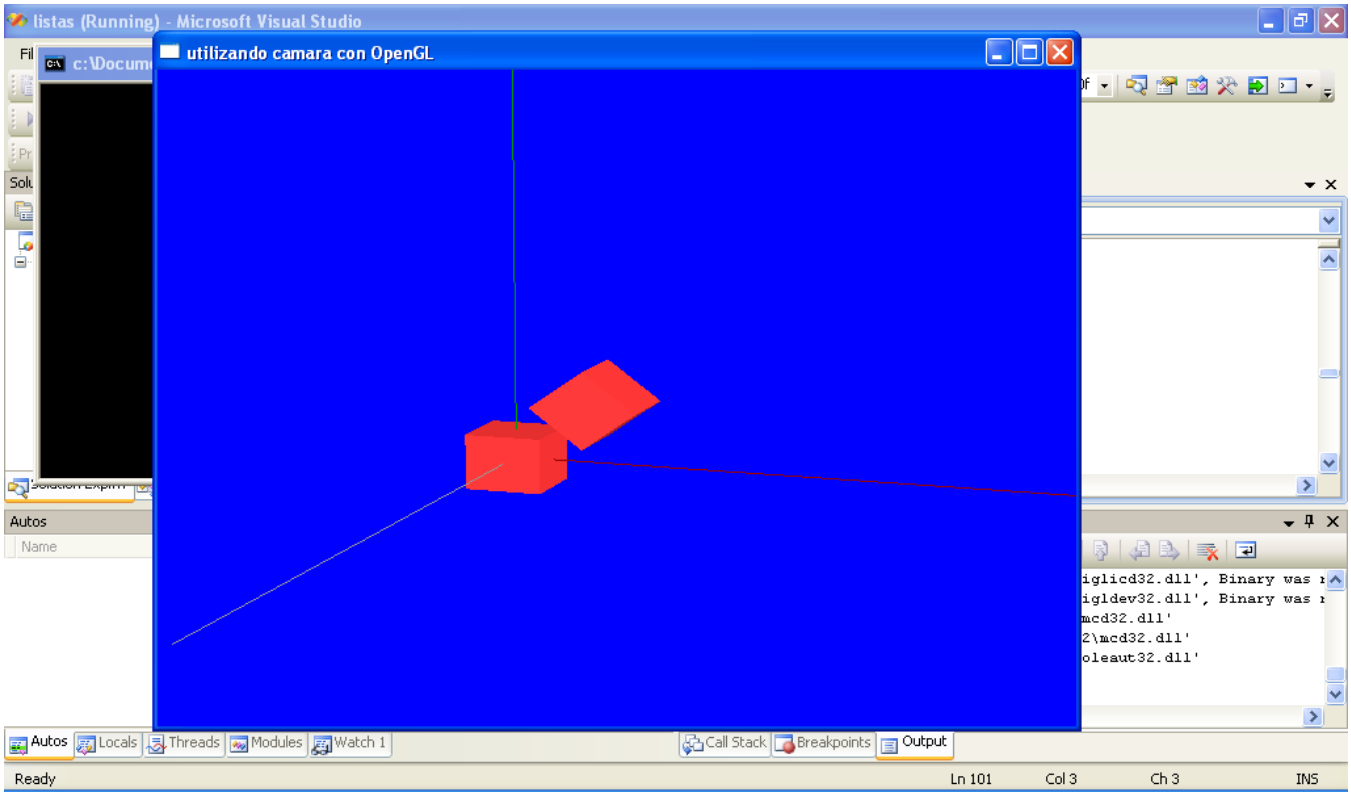
15. Retire los comentarios, construya y ejecute.

16. Reporte la gráfica obtenida.

17. Dentro de los argumentos de la linea "gluLookAt()", cambie las coordenadas del "eye", del "At" y del "Up".
At (punto de mira) y Up (Vector de orientacion)
18. Reporte la gráfica obtenida.

19. Elabore y reporte algún principio que permite explicar el orden que se definen las transformaciones dentro de la función "RenderScene()".

En esta parte de la practica se des comento la parte que esta entre comentarios que pertenece al ejercicio 3 y el resultado de la compilacion fue el siguiente



Cambie el vector de orientacion por el vector 1i+0j+0k
Tambien en punto de referencia por 1,1,1

Con el siguiente codigo

```
gluLookAt(30,25,80,1,1,1,1,0,0);
```

En la ventana podemos apreciar por el color rojo que ahora el eje x es el que apunta hacia arriba