PRÁCTICA FINAL

21714 - Informática Gráfica. 2021-2022

Joan Alcover Lladó – 45187596W
joan.alcover1@estudiant.uib.cat

Índice

Introducción

El objetivo principal de la realización de esta práctica es entender el funcionamiento de diferentes funcionalidades de la librería OpenGL y aplicarlas en su etapa correspondiente. De este modo, durante la realización de la práctica se trabajará usando diferentes técnicas como la del doble buffer, escalado de imágenes, además también se habrá trabajado creando escenarios con objetos en 3 dimensiones ya sean estáticos, en movimiento continuo o articulados. Dentro de esos escenarios en 3D se integrará el movimiento de la cámara para así poder tener perspectivas diferentes del escenario. Por otro lado, también se adquirirán conocimientos sobre luces y sombreados. Todas estas funcionalidades estarán controladas mediante el teclado.

Las etapas previas a la etapa 6 sirven para tener un entendimiento y comprensión máximos de todo lo explicado durante el curso. Una vez se dominan perfectamente estos conocimientos se procederá a realizar la etapa 6, la cual será el producto de juntar todas las funcionalidades ya aprendidas en las etapas anteriores.

En este caso, se pide recrear para la etapa final (etapa 6) una escena en concreto. Esta escena es un homenaje a Pixar y el vídeo de animación que fue creado en el 1986. En dicho vídeo de animación aparece una lámpara del tipo flexo y una pelota.

Para recrear esta animación se representará una lámpara totalmente articulada con una bombilla que producirá un foco de luz simulando así la luz que ejerce una lámpara del tipo flexo. Además de la lámpara también se representará una pelota en constante movimiento con la posibilidad de pararla. Esta pelota hace referencia a la pelota que aparece en el vídeo. Por otro lado, se añadirán las letras de la palabra “PIXAR” haciendo referencia a otro vídeo de animación donde la lámpara aplasta a la letra “I”. Es por eso que la posición de la lámpara será encima de dicha letra. Para finalizar se añadirán varios objetos en la escena, así como una luz móvil, una caja de madera y una cuchara dorada.

Etapas

Etapa 1

El programa que se ha utilizado para la realización de la práctica ha sido Visual Studio 2022. La descarga se ha realizado mediante la página oficial de Visual Studio y no ha supuesto ninguna complicación más allá de la inserción de los archivos correspondientes a las librerías necesarias.

Una vez instalado el programa, se ha creado una solución llamada “PracticasIG” en la cual se ha creado un proyecto para la etapa actual, la etapa 1. Para las siguientes etapas se han creado al momento de iniciar dichas etapas.

El objetivo de esta primera etapa es crear un triángulo que gire sobre sí mismo.

Se ha hecho uso de la función *glutInitDisplayMode(GLUT_RGBA | GLUD_SINGLE)* la cual es utilizada para crear ventanas y subventanas.

Cada uno de los triángulos se ha realizado usando la función *glBegin(GL_POLYGON)* y el uso de *glVertex3f().* La rotación se ha conseguido incrementando la primera variable de *glRotatef(fAngulo, 0.0f, 0.0f, 1.0f)* estableciendo el giro sobre el eje Z.

Para finalizar, se ha implementado una función *Idle* en la cual incrementamos el ángulo de rotación del triángulo.

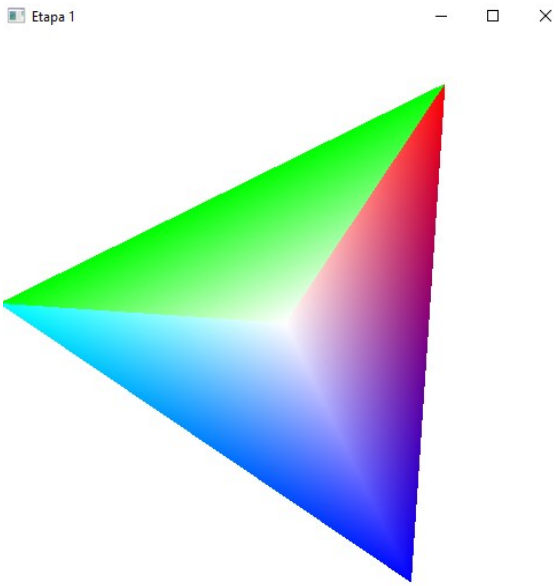

Ilustración 1. Ejecución de la etapa 1

Etapa 2

Modificación etapa 1

Para comenzar esta etapa se pide que se modifique el programa realizado en la etapa anterior, puesto que en ese las primitivas se realizan directamente sobre el buffer del dispositivo de pantalla por lo que se produce parpadeo al borrar la ventana y a continuación redibujarla. Gracias a la implementación del doble buffer, se visualiza sobre un segundo buffer y una vez se obtiene la imagen deseada se intercambia con el buffer de pantalla.

La función que se ha utilizado para la utilización del doble buffer ha sido *glutSwapBuffers()*. Además de la modificación de un parámetro de la función *glutInitDisplayMode()*, cambiando el parámetro *GLUT_SINGLE* por *GLUT_DOUBLE*.

A parte de utilizar el doble buffer, en esta etapa se pide que se realice una función para mantener el aspect ratio del dibujo implementado. Puesto que, en la etapa anterior, si se modificaba el tamaño de la pantalla, la figura se deformaba.

La función llamada *myReshape()* es la encargada de realizar dicha tarea de mantener el aspect ratio. Esta función se llama desde la función *main* mediante la función *glutReshapeFunc(myReshape)*.

Para la correcta comprobación de la función *myReshape()* se ha creado un cuadrado estático en el centro del triángulo, puesto que un cuadrado es más sencillo de comprobar la alteración de su forma.

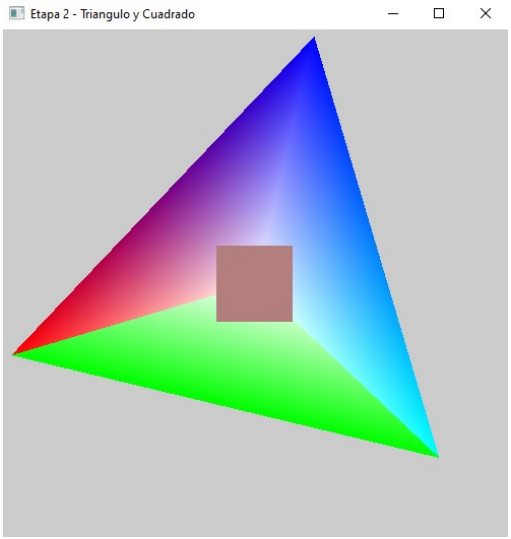

Ilustración 2. Ejecución de la etapa 2

Estructura articulada

Para profundizar los conocimientos adquiridos durante esta etapa y referentes a la creación de objetos 2D en movimiento, se ha decido realizar otro programa el cual simula un brazo articulado.

La realización de este programa ayudará a estudiar las transformaciones anidadas y el uso de las funciones de la pila de matrices (push y pop). En este caso se han utilizado las funciones push y pop para el movimiento global del brazo, simulando el giro de un brazo completo; para el movimiento de la parte superior del brazo, otra para la parte inferior y otra vez para el codo del brazo.

En este caso la función de reescalado incorpora tanto *GL_PROJECTION* como *GL_MODELVIEW* y *gluPerspective().*

Por último, se ha creado la función *lecturaEsc()*, mediante la cual se puede mover el brazo gracias a la interacción con las letras “q”, “a”, “w” y “s” del teclado. Las funciones que realizan las teclas son las siguientes:

Tecla	Función
q	Giro del brazo en sentido antihorario
w	Giro del brazo en sentido horario
a	Giro del antebrazo en sentido antihorario
s	Giro del antebrazo en sentido horario

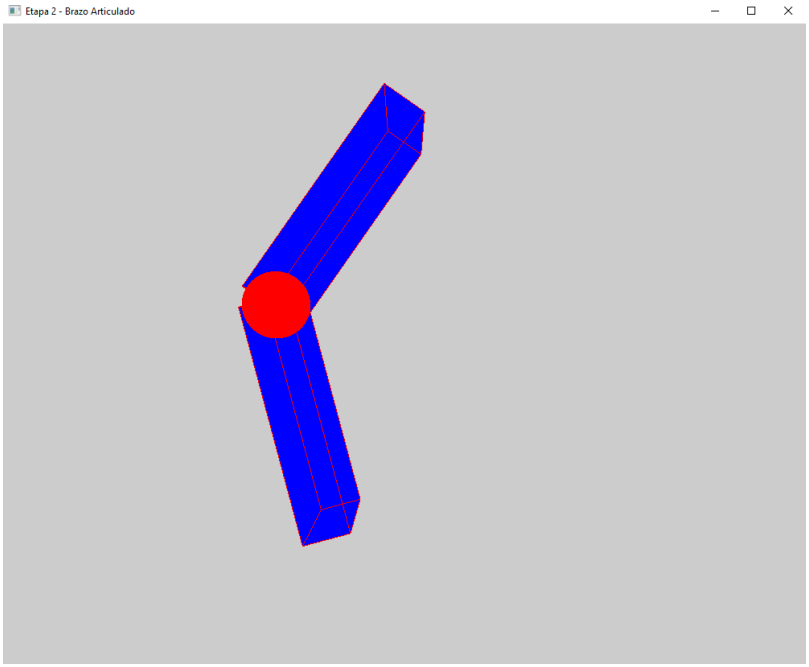

Ilustración 3. Ejecución de la etapa 2. Brazo articulado

Etapa 3

El objetivo de esta etapa es definir un sistema de coordenadas cartesiano con los ejes (X, Y, Z). Para ello se han definido las funciones de ejes. Se ha creado una escena con primitivas 3D y se han aplicado las funciones de transformación y visualización.

Como ya se sabe desde un principio cual iba a ser la escena de la etapa 6, se ha decidido empezar a implementar la lámpara en esta etapa con motivo de ir perfeccionándola a medida que se avancen las etapas.

La lámpara se ha realizado mediante el uso de *glutWireCone()* para la parte superior de la lámpara, *glutSolidCone()* para la base, *glutWireCylinder()* para la estructura de la lámpara (brazo superior e inferior) y para finalizar *glutWireSphere()* para las esquinas de los cilindros. Cabe mencionar que las esferas son de diferente color para remarcar al usuario los puntos articulados que tiene la lámpara.

Además, se ha utilizado la técnica del doble buffer explicada en etapas anteriores y la función *reshape()*. En esta función se ha añadido *gluPerspective()* el cual puede ser substituido por *glFrustum()* y de esta forma se pueden ver las diferentes perspectivas.

Para finalizar, se ha activado el buffer de profundidad para la percepción de proximidad de los objetos mediante *glEnable(GL_DEPTH_TEST)* y *glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT).*

El movimiento de la lámpara se realiza mediante la función *lecturaEsc().* Las funciones que realizan las teclas son las siguientes:

Tecla	Función
q	Subir brazo inferior
a	Bajar brazo inferior
w	Subir brazo superior
s	Bajar brazo superior
e	Inclinar foco hacia arriba
d	Inclinar foco hacia abajo
r	Girar foco hacia la izquierda
f	Girar foco hacia la derecha
t	Girar lámpara hacia la izquierda
g	Girar lámpara hacia la derecha

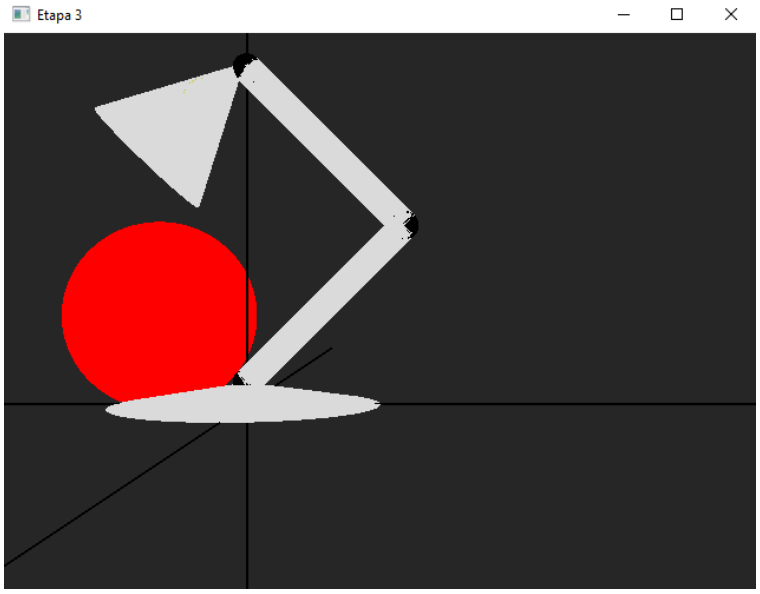

Ilustración 4. Posición inicial etapa 3

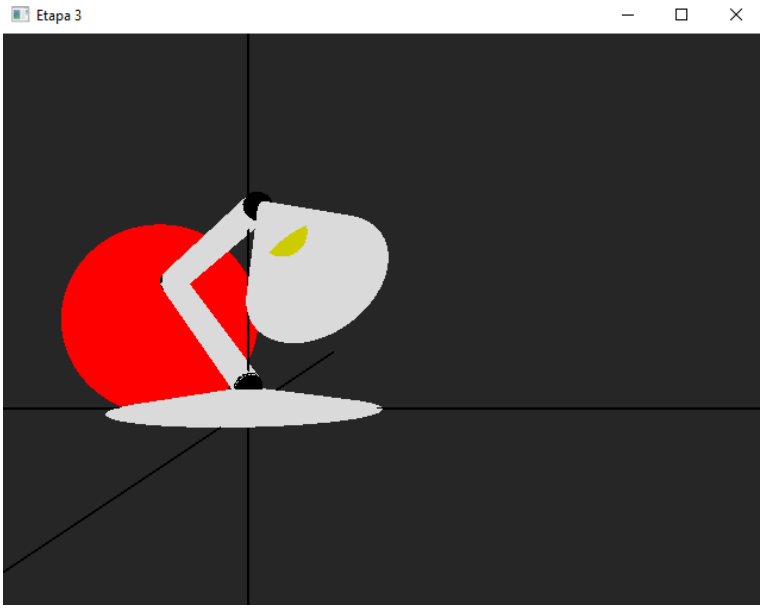

Ilustración 5. Posición modificada de la lámpara

Etapa 4

El objetivo de esta etapa es que el usuario pueda moverse libremente por la escena. Es decir, la cámara virtual que nos da la vista de nuestra escena podrá variar de acuerdo con una serie de funciones.

La escena a visualizar es la misma que en la etapa anterior, eliminando los ejes de coordenadas y añadiendo una cuadrícula que simula el suelo. Las opciones de movimiento de la lámpara siguen implementadas en esta etapa, con lo cual el usuario podrá mover libremente la lámpara.

Gracias a la interacción por teclado, el usuario podrá mover la cámara por toda la escena. Pudiendo así, ver cualquier ángulo de esta.

Mediante la interacción de las flechas del teclado se ha implementado un movimiento de Paneo y otro de Tilt (giro sobre sí mismo vertical y horizontalmente). Este movimiento se consigue modificando unos atributos los cuales, al repintar la pantalla van a modificar los atributos de la función *LookAt().* De esta forma se consigue variar el punto donde la cámara está enfocando. Además de estos movimientos, también se han añadido movimientos del tipo subir y bajar la cámara y rotaciones hacia los lados.

Gracias a estas funcionalidades implementadas, se ha conseguido un movimiento completo por toda la escena; permitiendo así tener una vista desde diferentes perspectivas del mismo objeto.

Se ha hecho uso de la función *glutSpecialFunc()* para la interacción con las flechas del teclado además de *glutPerspective()* y *gluLookAt()* para definir las vistas de la cámara.

Para finalizar se ha usado el doble buffer y el reshape de las etapas anteriores.

El movimiento de la lámpara se hace de la misma manera que en la etapa anterior. En cuanto al movimiento de la cámara, estas son las funciones que realiza cada tecla:

Tecla	Función
Flecha izquierda	Paneo a la izquierda
Flecha derecha	Paneo a la derecha
Flecha arriba	Inclinar hacia arriba (tilt)
Flecha abajo	Inclinar hacia abajo(tilt)
y	Subir la cámara
h	Bajar la cámara
u	Rotar hacia la izquierda
j	Rotar hacia la derecha
i	Zoom hacia adelante
k	Zoom hacia atrás
1	Reset de la cámara

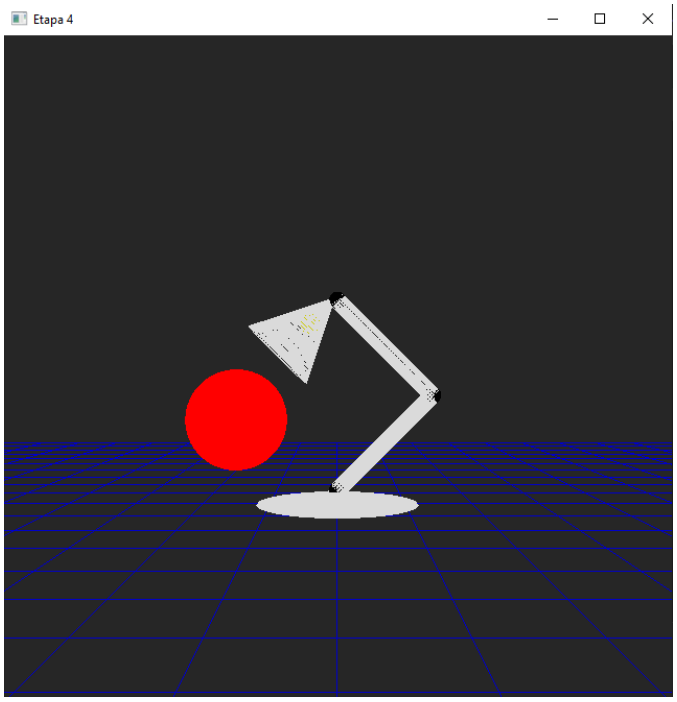

Ilustración 6. Posición inicial etapa 4

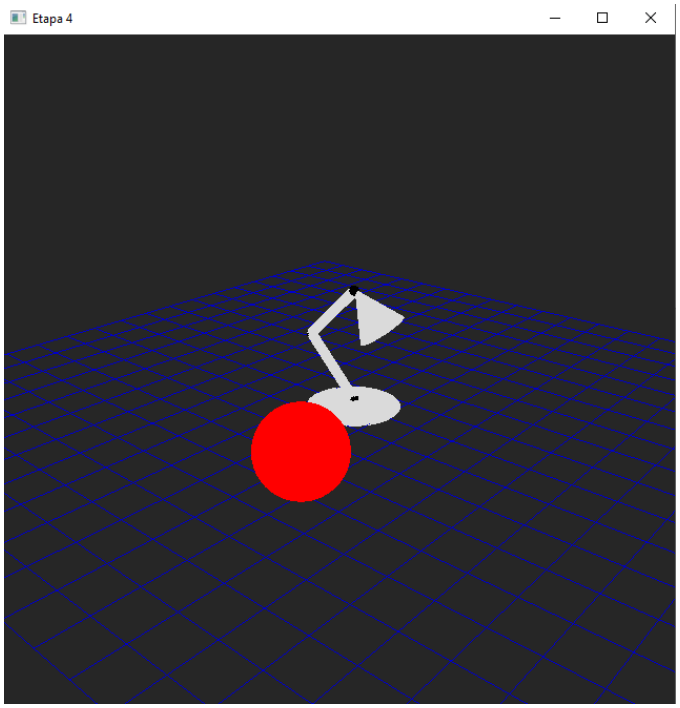

Ilustración 7. Vista modificada etapa 4

Etapa 5

El objetivo de esta etapa es iniciarse en el proceso de realismo de la escena. Para ello se añadirán luces a la escena y materiales a los objetos.

La escena realizada consiste en la lámpara de lasa anteriores etapas situada encima de una letra "I" que simula la letra "I" de la animación de Pixar, además contiene una esfera roja y una figura 3D con la forma de un donut. Para finalizar, contiene una esfera más pequeña que nos indica la posición de la luz móvil que se va a implementar. En la lámpara se ha añadido una luz de foco en la bombilla de esta. Además, se ha añadido otra luz que dará iluminación genérica a la escena. Estas luces se reflejarán en los elementos de la escena.

Para ello se ha habilitado la renderización de la luz usando *glEnable(GL_LIGHTING)*, *glEnable(GL_COLOR_MATERIAL)* y *glColorMaterial(GL_FRONT_AND_BACK, GL_AMBIENT_AND_DIFFUSE)*. Dentro del método *initRender()* se hacen los *glEnable()* de las luces para que funcionen.

Para la interacción de las luces se ha realizado mediante la entrada por teclado, permitiendo así el encendido y apagado de las luces y el movimiento de la luz móvil (bola roja pequeña). También se ha implementado mediante la tecla *espacio* que se diferencie entre *GL_FLAT* y *GL_SMOOTH*, trabajando los sombreados con *glShadeModel().*

Además de estas nuevas implementaciones, se conservan todos los movimientos de la lámpara y de al cámara de las etapas anteriores. También se conservan la función de reshape y se ha usado doble buffer.

La interacción con las luces se hace con las siguientes teclas:

Tecla	Función
z	Enciende/apaga la luz de la lámpara
x	Enciende/apaga la luz general
c	Enciende/apaga la luz móvil
v	Mueve la luz hacia la izquierda
b	Mueve la luz hacia la derecha
n	Mueve la luz hacia arriba
m	Mueve la luz hacia abajo
o	Mueve la luz hacia adelante
l	Mueve la luz hacia atrás
espacio	Cambia las superficies de los objetos

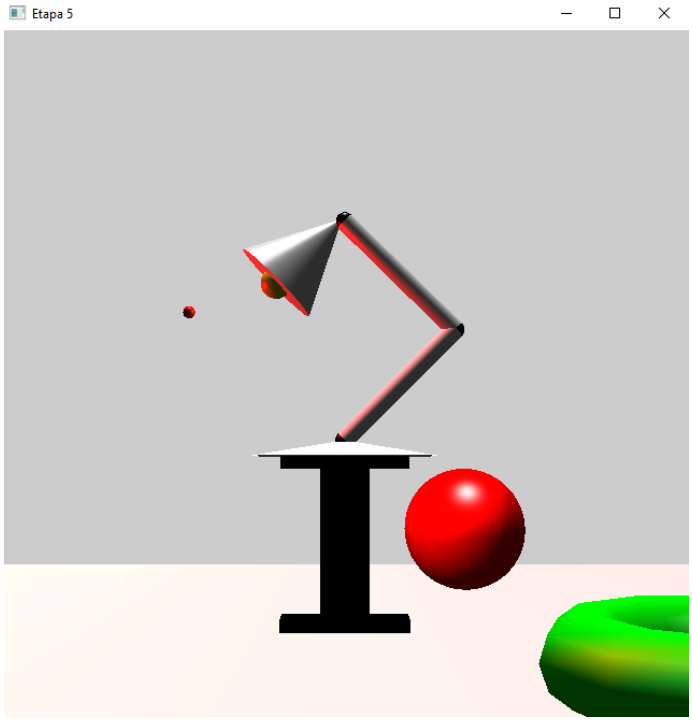

Ilustración 8. Posición inicial etapa 5

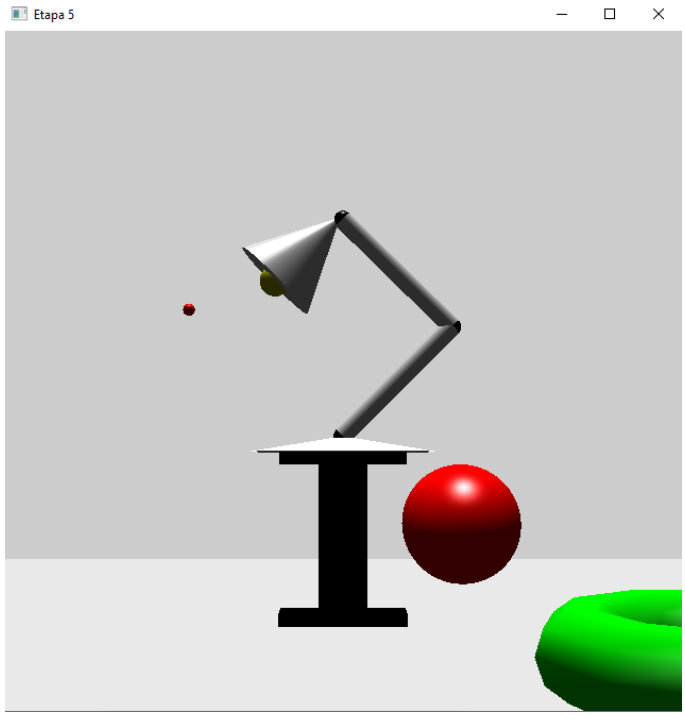

Ilustración 9. Sólo luz general activada

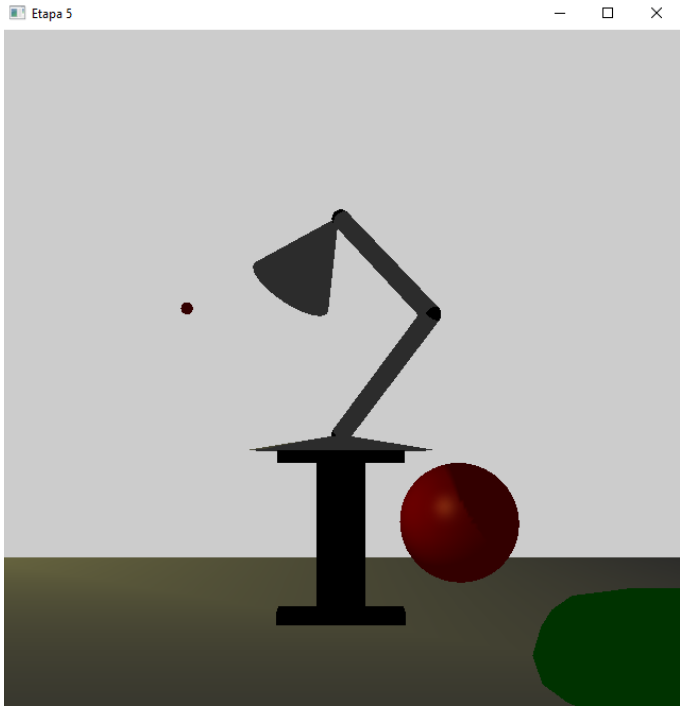

Ilustración 10. Luz del foco empieza a dar a la bola

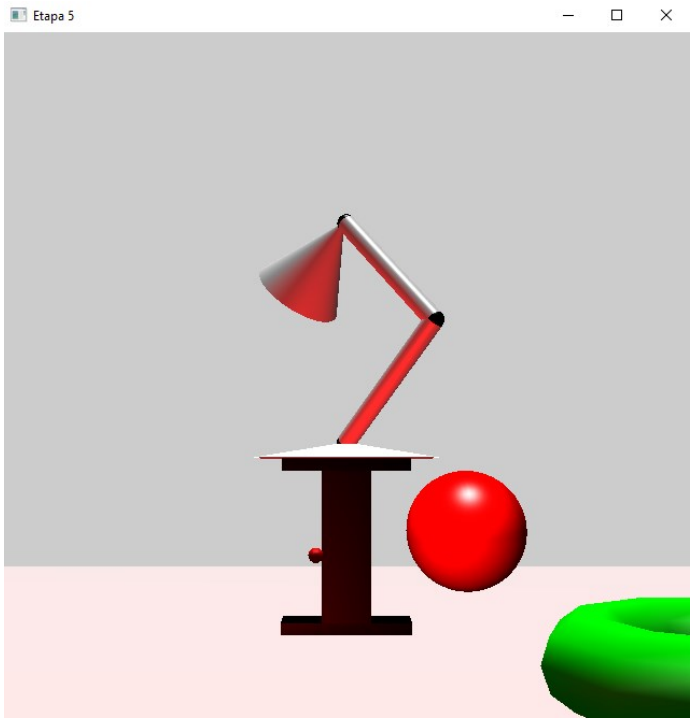

Ilustración 11. Luz general y luz móvil

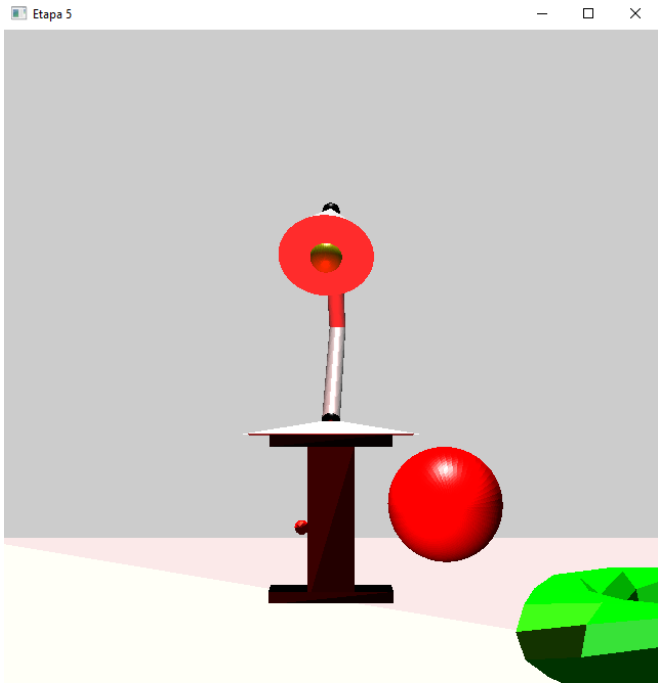

Ilustración 12. Todas las luces y GL_FLAT

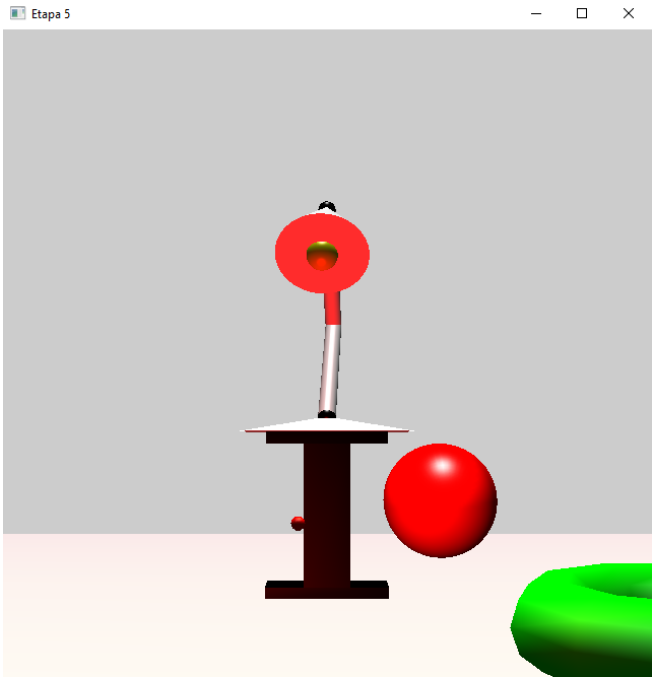

Ilustración 13. Todas las luces y GL_SMOOTH

Etapa 6

El objetivo de esta etapa es crear una escena donde se junten todos los conocimientos adquiridos en las etapas anteriores y se vean reflejados en la misma.

La escena, como ya se ha dicho anteriormente, consistirá en realizar un homenaje a Pixar y al vídeo de animación que fue creado en el 1986. En la escena a realizar aparecerá la lámpara totalmente articulada, las letras de la palabra "PIXAR", una luz móvil, una pelota (luxo ball), una caja de madera y una cuchara dorada.

Main

En la función *main* es donde se llaman a todas las funciones que son necesarias para pintar la escena deseada. En primer lugar, se inicializa el *glut*. Seguidamente, inicializamos el *glutInitDisplayMode()* con los parámetros *GLUT_DEPTH* (máscara de bits para seleccionar una ventana con un buffer de profundidad), *GLUT_DOUBLE* (máscara de bits para seleccionar una ventana con doble buffer), *GLUT_RGBA* (máscara de bits para seleccionar una ventana de modo RGBA) y *GLUT_MULTISAMPLE* (máscara de bits para anti-aliasing).

A continuación, indicamos la posición de la ventana mediante la función *glutInitWindowPosition(),* indicamos el tamaño de la ventana con *glutInitWindowSize(),* indicamos el nombre con *glutCreateWindow().* Seguidamente se llama a la función *initRender().*

Una vez se ha realizado esto, se llaman a las funciones necesarias para dibujar la escena e interactuar con ella.

En primer lugar se llama a la función *glutDisplayFunc(Display),* que incluye toda la escena. Después se aplica el reshape mediante *glutReshapeFunc(myReshape).* Llamamos a la función *glutIdleFunc(Display)* y se recogerán las teclas por teclado mediante *glutKeyboardFunc(lecturaEsc)* y *glutSpecialFunc(lecturaFlechas)*. Una vez realizado esto, se llama a la función *cargarTexturas().*

Finalmente hacemos un *glutMainLoop()* para entrar en un ciclo de procesamientos.

Texturas (parte opcional)

En esta etapa se han realizado figuras 3D con texturas. Estas texturas son aplicadas mediante la librería externa "tgload". Esta librería permite implementar texturas a objetos de manera sencilla.

En la función *cargarTexturas()* se habilitan las texturas mediante el comando *glEnable(GL_TEXTURE_2D)* y luego se cargan las texturas mediante *glBindTexture()* y

tgaLoad(). Se permiten guardar hasta 8 texturas, en este caso es suficiente con 5. Las texturas elegidas son para la bola (luxo ball), para el suelo, para una caja de madera y para una bola de luz y para una cuchara.

Una vez cargadas las texturas, se deshabilitan mediante *glDisable(GL_TEXTURE_2D).*

Escalado

La función *myReshape()* se encarga de que las proporciones de los elementos de la escena no sean modificados por culpa de un cambio en la altura o anchura de la ventana.

En primer lugar, se comprueba que el ancho o el alto de la ventana sea diferente de 0.

A continuación, se usa la matriz de proyección *glMatrixMode(GL_PROJECTION)*, esto viene seguido de un reset de la matriz mediante *glLoadIdentity()*. Seguidamente se establece la perspectiva con *gluPerspective()* y volvemos al modelview. Finalmente definimos la ventana completa con *glViewport()*.

Render

En el método *initRender()* se han activado los siguientes elementos:

- *glEnable(GL_DEPTH_TEST):* Cada vez que se va a renderizar un pixel, se comprueba que no se haya pintado antes en esa posición un pixel que se encuentre más cerca de la posición de la cámara.
- *glEnable(GL_COLOR_MATERIAL):* Se utiliza para especificar qué parámetros de material siguen el color actual.
- *glEnable(GL_LIGHTING):* Se utiliza para activar las diferentes luces. En este caso se han activado las luces 0 y 1 mediante *glEnable(GL_LIGHT0)* y *glEnable(GL_LIGHT1).*
- *glEnable(GL_NORMALIZE):* Se utiliza para normalizar los vectores normales a la longitud unitaria después de la transformación y antes de la iluminación.
- *glEnable(GL_SHADE_MODEL):* se utiliza para indicar la técnica de sombreado. En este caso se ha *utilizado glShadeModel(GL_SMOOTH).*

Además, en este método, se pueden encontrar una llamada a una función setLights(), las propiedades de los materiales de la escena y todo lo relacionado con la niebla que se detallará más adelante.

Lectura de teclado

Con la finalidad de que el usuario de la aplicación pueda interaccionar con la escena, se han utilizado diferentes teclas del teclado para realizar funciones en concreto.

En el caso de la interacción con las flechas del teclado se ha hecho uso de la función glutKeyboardFunc(), mientras que las teclas restantes se ha hecho uso de la función glutSpecialFunc(). De esta forma se ha conseguido una interacción mediante cualquier tecla del teclado del ordenador.

SetLights

En la función *setLights()* es donde se inicializan las luces con sus parámetros iniciales.

- Luz 0 (foco), le asignamos el ángulo y el exponente de la spot light. También se le asigna el *GL_DIFFUSE* y *el GL_SPECULAR*.
- Luz 1 (global), se le asigna el *GL_DIFFUSE* y *el GL_SPECULAR*.
- Luz 2 (móvil), se le asigna el *GL_DIFFUSE* y *el GL_SPECULAR*.

Lámpara

La lámpara ha sido creada mediante la unión de diferentes figuras 3D. Entre ellas, tenemos *glutSolidCone()* para la base de la lámpara y el foco, *glutSolidCylinder()* para los brazos de la lámpara y *glutSolidSphere()* para los puntos de articulación y para la bombilla.

Para saber cómo se tiene que crear la lámpara, en cada iteración se actualizan los ángulos de los brazos y del foco, así como el ángulo de rotación total. De esta forma, se puede saber la posición de cada elemento mediante las operaciones de seno y coseno de los ángulos.

En cuanto a la luz de la bombilla de la lámpara, es decir la luz 0, es en este método dónde se asigna la posición de la misma además de su dirección de apuntado. Si la luz está encendida, se deshabilitan los efectos de la luz para crear la esfera que conformará la bombilla y, una vez creada, se vuelven a habilitar dichos efectos. Esto proporciona una sensación de encendido y apagado de la bombilla.

Cabe mencionar que los ángulos de la lámpara pueden ser modificados por el usuario utilizando las teclas correspondientes a los movimientos de la lámpara.

Letras

Las letras se han realizado mediante la unión de *glutSolidCylinder()* para las letras “A” y “X”, *glutSolidCylinder()* y *glutSolidTorus()* para las letras “P” y “R”, y *glutSolidCylinder()* y *glutSolidCube()* para la letra “I”. Todas estas figuras han sido modificadas mediante las funciones *glScalef()* y *glRotatef()* y trasladadas mediante *glTranslatef()* para su correcta disposición y tamaño.

Cada letra ha sido creada dentro de su propia función y la función *letras()* agrupa todas las letras para una mayor comprensión.

Pelota (luxo ball)

En la función *pelota()* es donde se crea la pelota con textura que gira alrededor de las letras.

Para simular que la pelota está rodando por el suelo se le han aplicado una serie de rotaciones que permiten que gire sobre sí misma a la vez que va dando la vuelta en las letras. Esto es posible gracias a las variables *anguloPelota* y *ruedaPelota* que son modificadas en cada iteración.

A continuación, se habilitan las texturas mediante *glEnable(GL_TEXTURE_2D)* y seguidamente se establecen los parámetros de entorno de textura mediante *glTexEnvi(GL_TEXTURE_2D, GL_TEXTURE_ENV_MODE, (int)GL_DECAL).* Después se vincula la textura que se desea mediante *glBindTexture(GL_TEXTURE_2D, tex[0])* donde *tex[0]* es la textura de la pelota.

En último lugar, se crea un objeto del tipo *quadric* mediante *p = gluNewQuadric()* y luego se le asigna la textura previamente cargada mediante *gluQuadricTexture(p, GL_TRUE).* A continuación, se crea la esfera con *gluSphere(p, radPelota, 40, 40).* Finalmente, se deshabilitan las texturas con *glDisable(GL_TEXTURE_2D).*

Bola de la luz

En la función *bolaLuz()* es donde se crea la bola que representa la luz móvil. Esta bola también es creada con texturas. Es por eso que el proceso de creado de la esfera es el mismo que en el de la pelota anteriormente mencionada, exceptuando los valores de la posición los cuales son variables que dependen de la posición de la luz.

Caja de madera y suelo

En la función *cube(nro_da_textura, x, y, z)* es donde se crean el suelo y la caja de madera.

En primer lugar, habilitamos las texturas con *glEnable(GL_TEXTURE_2D)* y cargamos la textura correspondiente (la pasada por parámetro) mediante *glBindTexture(GL_TEXTURE_2D, tex[nro_da_textura]).*

A continuación se crean las 6 caras de un cubo mediante la concatenación de *glTexCoord2f()* y *glVertex3f().* Los parámetros de *glVertex3f()* son los correspondientes con los pasados por parámetro en la función (x, y, z).

Para finalizar, se deshabilitan las texturas con *glDisable(GL_TEXTURE_2D).*

Cuchara

La cuchara es el último elemento que contiene texturas. Este objeto se crea dentro de la función *cuchara()*, la cual lo primero que hace es trasladar la cuchara a la posición

deseada mediante *glTranslatef()* y a continuación, se rota para que aparezca tumbada en el suelo mediante *glRotatef().*

Seguidamente se habilitan las texturas como en las funciones anteriores, se carga la textura correspondiente y se crea un objeto del tipo *quadric* como en el caso de las esferas. Se le asigna la textura al objeto con *gluQuadricTexture(p, GL_TRUE)* y finalmente, se crea la cuchara con *glutSolidTeaspoon(0.6).*

Para finalizar se deshabilitan las texturas con *glDisable(GL_TEXTURE_2D).*

Dibujar la escena

Toda la escena está pintada en el la función *Display().*

En esta función se ha hecho uso de *gluLookAt()* con la finalidad de definir una transformación de visualización. Gracias a *gluLookAt()* se crea una matriz de visualización derivada de un punto de vista, un punto de referencia que indica el centro de la escena y un vector UP. Los dos primeros vectores están formados por variables ya que el movimiento de la cámara se efectuará mediante el teclado y se irá actualizando en cada iteración. El usuario podrá modificar tanto la posición de la cámara como los ángulos de visión de esta.

A continuación, y antes de pintar los objetos de la escena, asignamos las posiciones de las luces. Seguidamente, encontramos la gestión del giro de la pelota, donde si la pelota no está parada, modificará los atributos de los ángulos.

Seguidamente, se procede a dibujar todos los objetos de la escena, en primer lugar se llama a *bolaLuz(),* a continuación se crea el suelo, la pelota que gira, la cuchara, la caja, las letras y finalmente la lámpara.

El método finaliza introduciendo el doble buffer con *glutSwapBuffers(),* para realizar un intercambio de buffer en la capa en uso para la ventana actual promoviendo el contenido del buffer posterior de la capa en uso de la ventana actual para convertirse en el contenido del buffer frontal.

Niebla (parte opcional)

Se ha incorporado niebla a la escena usando la función *glFog().*

En primer lugar, se declara al principio del programa una variable del tipo *float* que representará el color de la niebla, después otra variable del tipo *float* que indicará la densidad de la niebla. Esta densidad podrá ser modificada mediante el teclado con la función *glFogf(FL_FOG_DENSITY, fogDensity),* sonde dependiendo del valor de la intensidad de la niebla, esta se mostrará con mayor o menos intensidad.

Finalmente, en la función *initRender()* se hace un *glEnable(GL_FOG)* para habilitarla. A continuación, se hace un *glFogi(GL_FOG_MODE, GL_EXP)* donde se declara el modo de niebla inicial y finalmente se añade la densidad y el color de la misma con *GL_FOG_DENSITY* y *GL_FOG_COLOR*.

Anti-aliasing (parte opcional)

Se ha incorporado anti-aliasing a la escena mediante, en primer lugar aplicando *GLUT_MULTISAMPLE* en la función *glutInitDisplayMode* y posteriormente activando o desactivando este efecto mediante las funciones *glEnable(GL_MULTISAMPLE)* y *glDisable(GL_MULTISAMPLE).*

Alisado de las figuras

Se ha implementado dos versiones del alisado de figuras, la primera por defecto mediante la función *glShadeModel(GL_SMOOTH)* donde el brillo aparecerá difuminado y la segunda mediante *glShadeModel(GL_FLAT)* donde se verá toda la cara iluminada.

Interacción con la cámara

Paneos laterales

Los paneos laterales se consiguen modificando el segundo vector del *gluLookAt(),* que indicaba hacia donde miraba la cámara. Para ello, se ha creado una variable ángulo, y por cada iteración, se aplicará el seno del mismo en el eje X y el coseno negativo en el eje Z.

Zoom

El zoom se consigue actualizando el primer vector del *gluLookAt(),* ya que se busca mover la cámara de sitio, no donde esta apunta como en los paneos. El aumento de las coordenadas del primer vector se consigue sumando el valor del eje correspondiente del primer vector al resultado de la multiplicación de las coordenadas del segundo vector del mismo eje por 0.1. En el caso de que el zoom sea hacia atrás, habría que realizar una resta en lugar de la suma.

Movimiento vertical

El movimiento de la posición de la cámara se ha conseguido sumando 0.01 cada vez al valor de la Y del primer vector del *gluLookAt()* para ascender y restando para descender.

Rotación

La rotación de la cámara se consigue modificando el ángulo de rotación de un *glRotate()* situado debajo del *gluLookAt().* En el caso de incrementar el ángulo se rotará en sentido antihorario y viceversa al realizar la resta.

Tilt

El giro de la cámara sobre su propio eje hacia arriba y abajo se consigue de igual forma que los paneos, pero cambiando del eje Y en el segundo vector.

Distintas posiciones

Se han creado distintas posiciones (vistas) predefinidas modificando los valores de los dos primeros vectores de la función *gluLookAt().* Hay 4 vistas predefinidas en total.

Interacción con la lámpara

La interacción con la lámpara se a hecho modificando las variables de los ángulos de cada parte de la lámpara. De este modo, se pueden hacer todos los movimientos realizando la función *glRotate()* indicando el ángulo deseado.

Interacción con las luces

La interacción que se realiza con las luces (encendido/apagado) se realiza con las funciones *glEnable()* y *glDisable()* indicando la luz afectada.

Además, la interacción de movimiento de la luz móvil se ha realizado incrementando o decrementando la posición de la luz en 0.01.

Para finalizar cabe destacar que esta práctica se ha realizado utilizando la librería "freeglut" para los gráficos y la librería "tgload" para las texturas.

Resumen interacción con teclas

- **Cámara**

Tecla	Función
Flecha izquierda	Paneo a la izquierda
Flecha derecha	Paneo a la derecha
Flecha arriba	Inclinar hacia arriba (tilt)
Flecha abajo	Inclinar hacia abajo(tilt)
y	Subir la cámara
h	Bajar la cámara
u	Rotar hacia la izquierda
j	Rotar hacia la derecha
i	Zoom hacia adelante
k	Zoom hacia atrás
1	Posición cámara 1 (por defecto)
2	Posición cámara 2
3	Posición cámara 3
4	Posición cámara 4

- **Lámpara**

Tecla	Función
q	Subir brazo inferior
a	Bajar brazo inferior
w	Subir brazo superior
s	Bajar brazo superior
e	Inclinar foco hacia arriba
d	Inclinar foco hacia abajo
r	Girar foco hacia la izquierda
f	Girar foco hacia la derecha
t	Girar lámpara hacia la izquierda
g	Girar lámpara hacia la derecha

- **Luces**

Tecla	Función
z	Enciende/apaga la luz de la lámpara
x	Enciende/apaga la luz general
c	Enciende/apaga la luz móvil
v	Mueve la luz hacia la izquierda
b	Mueve la luz hacia la derecha
n	Mueve la luz hacia arriba
m	Mueve la luz hacia abajo
o	Mueve la luz hacia adelante
l	Mueve la luz hacia atrás

- **Otros**

Tecla	Función
5	Aumentar niebla
6	Disminuir niebla
7	Activar/Desactivar Anti-aliasing
p	Pausar/Reanudar la pelota
espacio	Cambia los alisados de las figuras

Ejemplos de ejecución

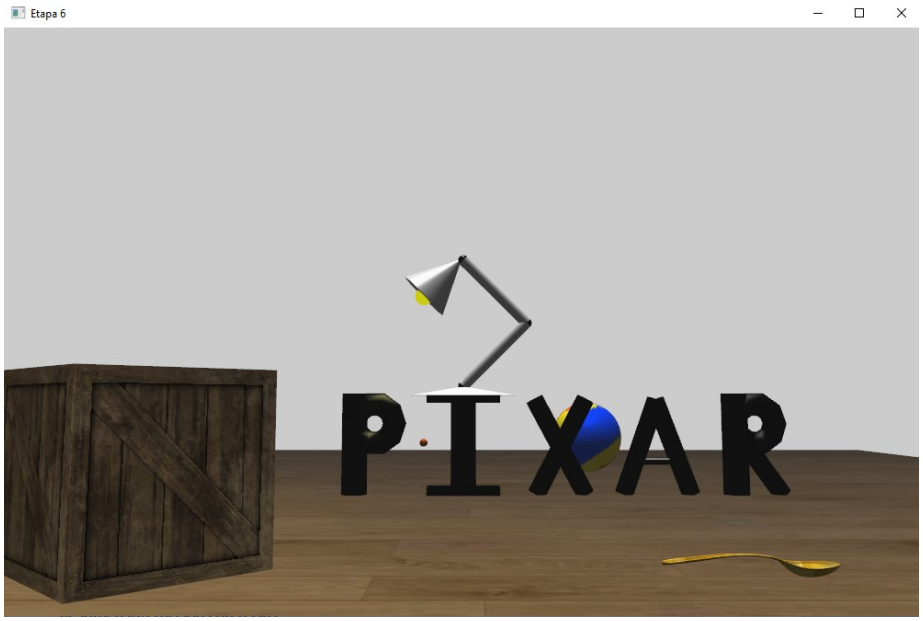

Ilustración 14. Escena inicial con luz global y luz de la lámpara.

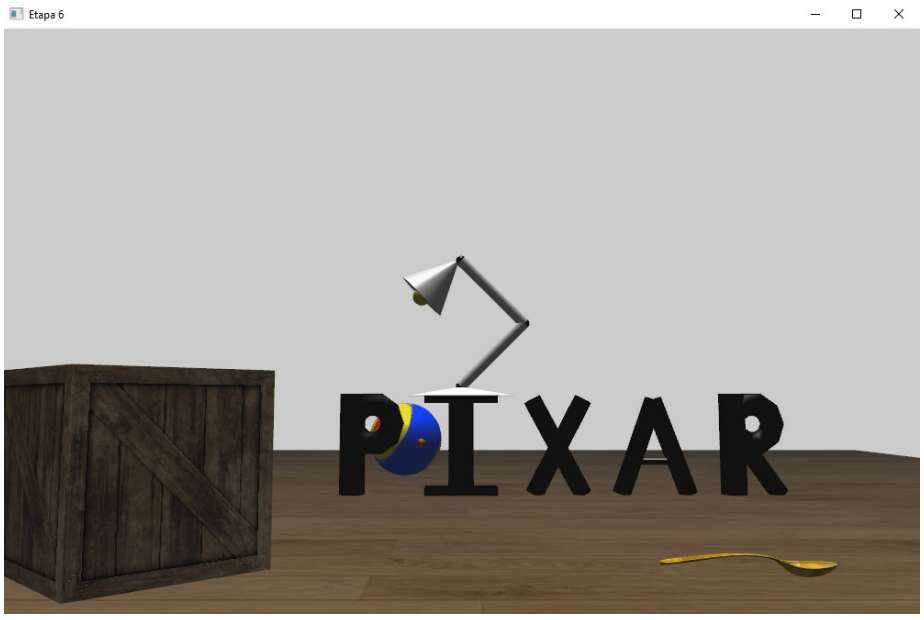

Ilustración 15. Escena inicial con luz global.

Ilustración 16. Escena inicial con luz de la lámpara.

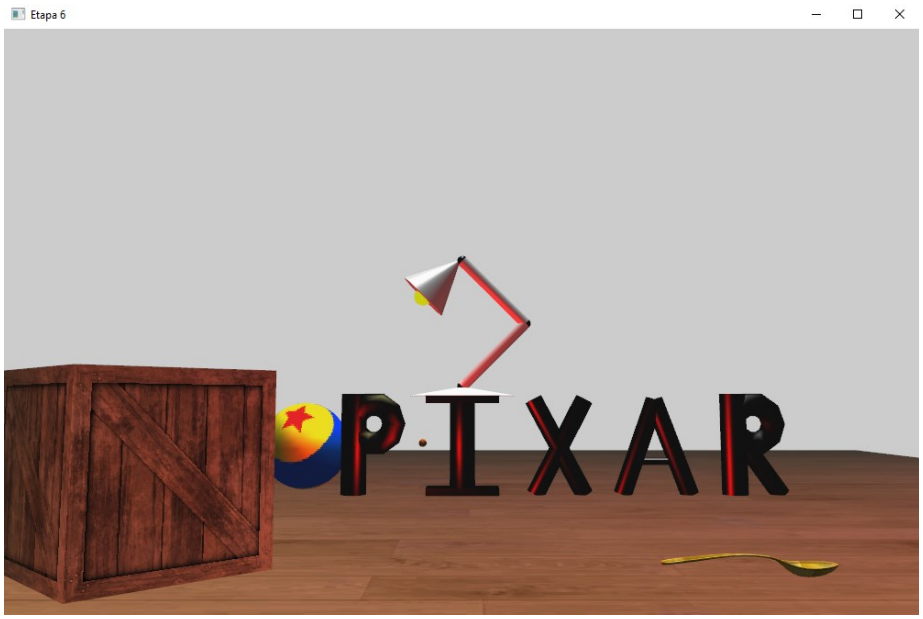

Ilustración 17. Escena inicial con luz global, luz de la lámpara y luz móvil.

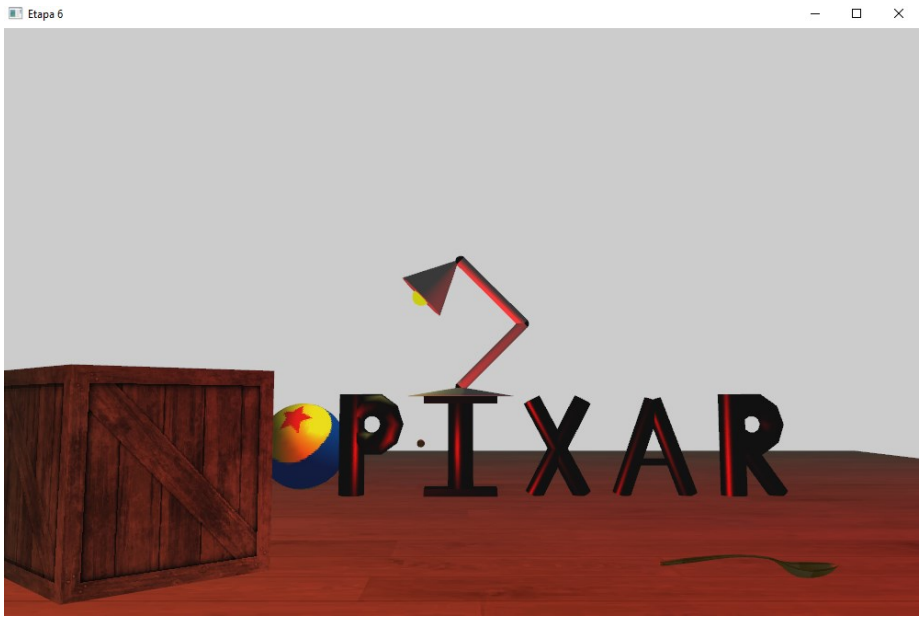

Ilustración 18. Escena inicial con luz de la lámpara y luz móvil.

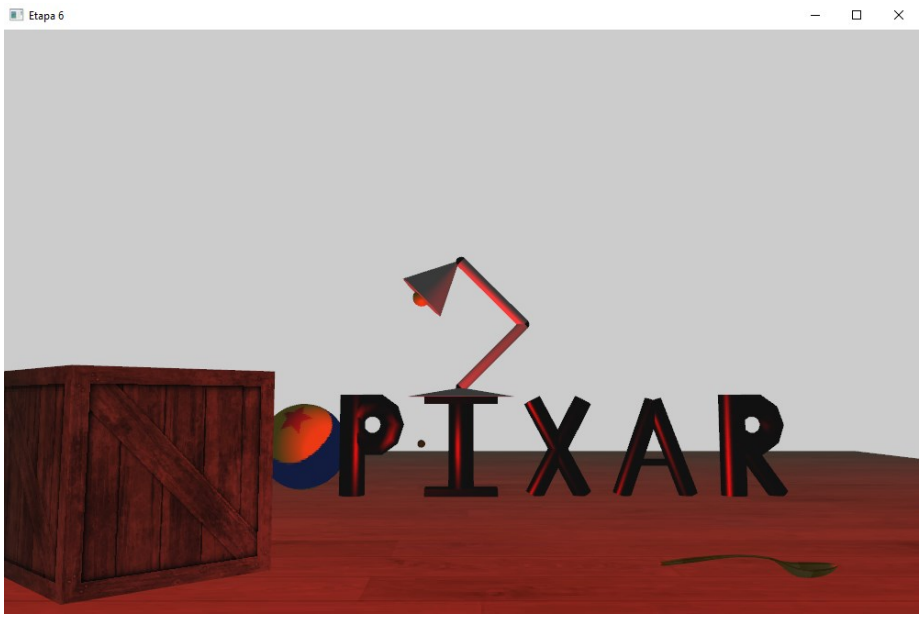

Ilustración 19. Escena inicial con luz móvil.

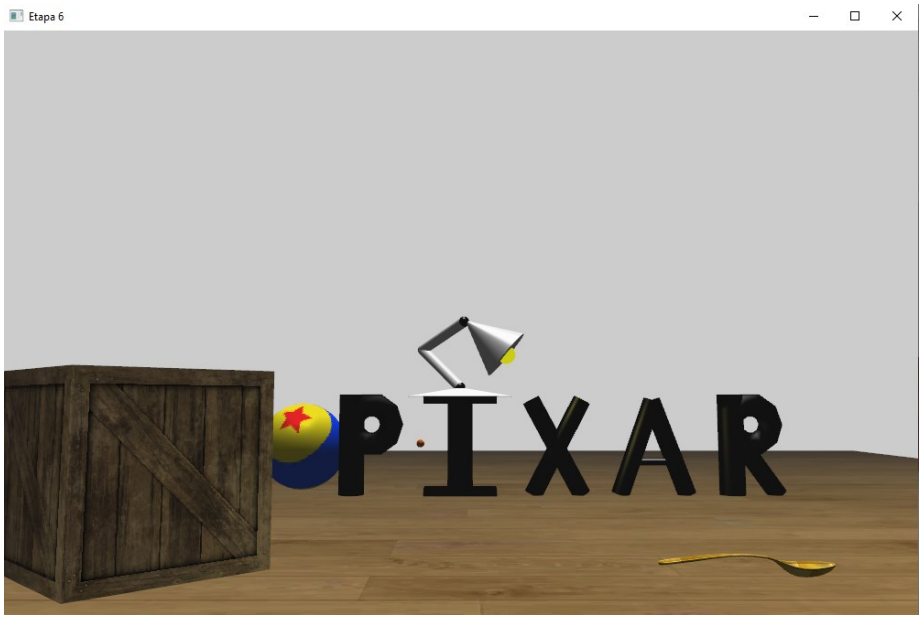

Ilustración 20. Cambio de posición de la lámpara.

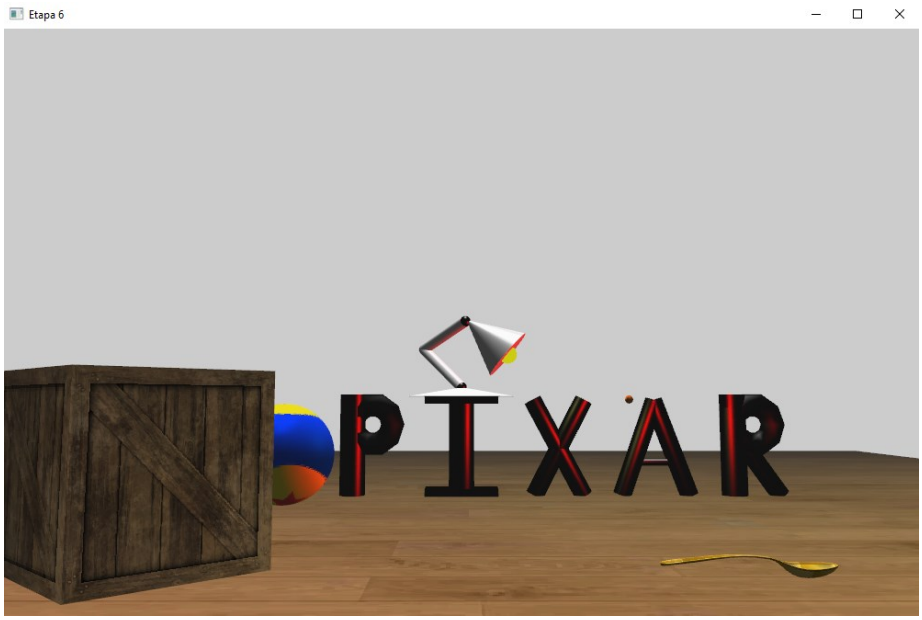

Ilustración 21. Cambio de posición luz móvil.

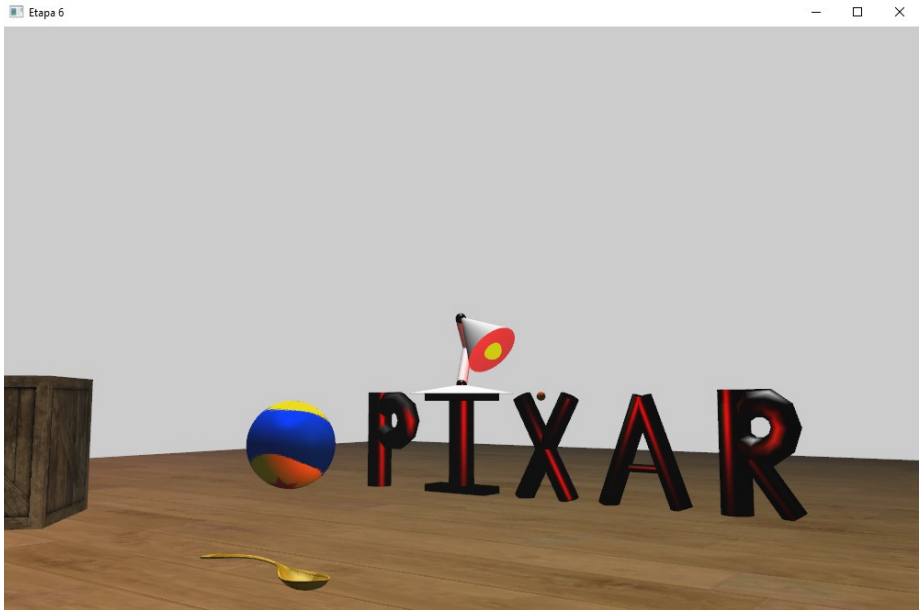

Ilustración 22. Rotación sobre la escena.

Ilustración 23. Movimiento vertical de la cámara + tilt.

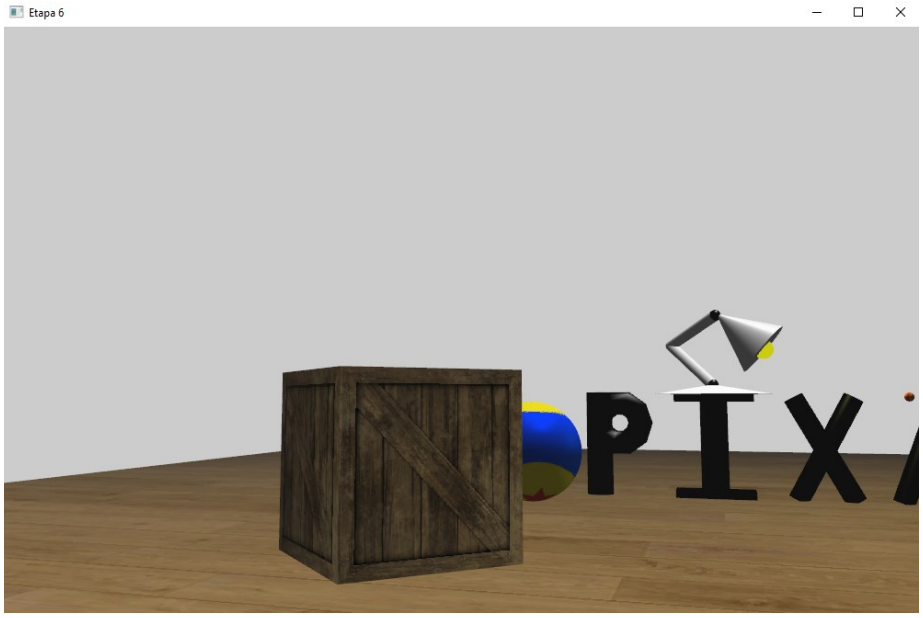

Ilustración 24. Movimiento horizontal de la cámara (paneo).

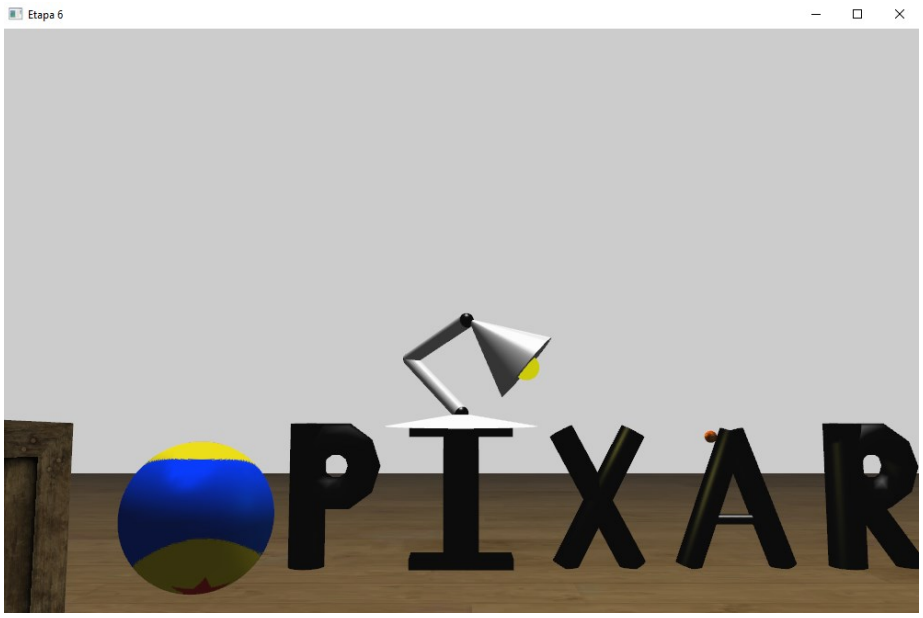

Ilustración 25. Zoom de la cámara.

Ilustración 26. Niebla activada.

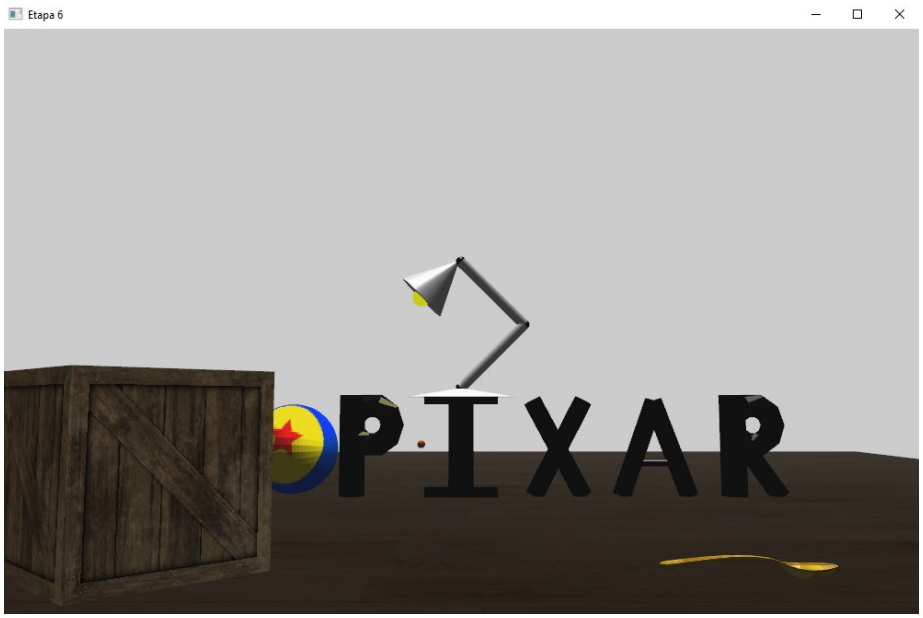

Ilustración 27. Alisado de figuras modo GL_FLAT.

Ilustración 28. Anti-aliasing desactivado.

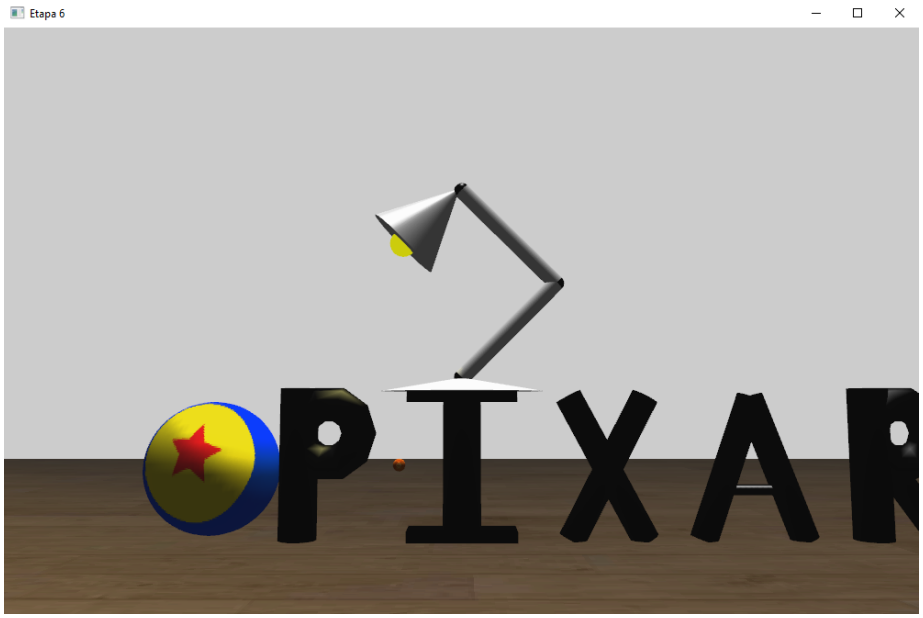

Ilustración 29. Vista número 2.

Ilustración 30. Vista número 3 + luz de la lámpara.

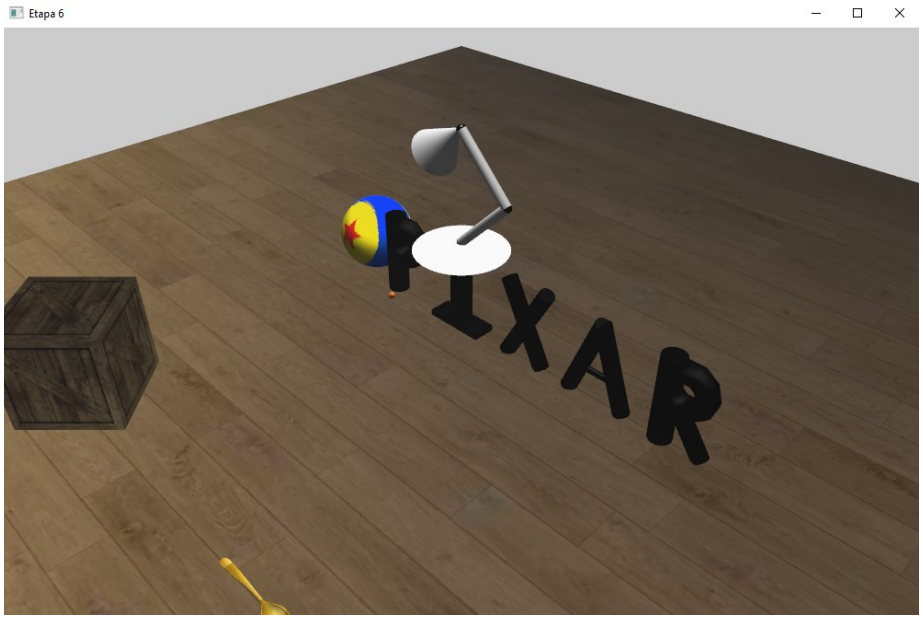

Ilustración 31. Vista número 4.

Conclusiones y comentarios

La realización de esta práctica ha sido de mi agrado puesto que se ha tenido que lidiar con un contenido que me apasiona y con el que me gustaría poder trabajar en un futuro. Es por eso que ha resultado entretenida a la par que interesante.

Considero que los contenidos de la práctica se adaptan de forma total a los contenidos de la asignatura. Además, la distribución de esta en etapas ha resultado ser un punto clave puesto que de esta manera se va trabajando desde lo más básico y luego, poco a poco ir subiendo de nivel de complejidad.

Esta asignatura es de las que más me ha aportado a nivel práctico de todas las asignaturas del curso, pues el temario con el que se ha tratado es el que se corresponde con varias escenas que vemos en el día a día. Además de eso, se ha aprendido a utilizar correctamente los buffers, a realizar diferentes funciones como las rotaciones o traslaciones entre otras. También se ha aprendido el funcionamiento de las vistas de la cámara. Por último, se han ampliado los conocimientos sobre los tipos de luces y como interactúan dichas luces con los objetos.

Otro punto a favor de la planificación de la práctica es la de dar al alumno una referencia para realizar como etapa final que en este caso ha sido un homenaje a Pixar. Si no se hubiera dado esta idea, habría resultado más complicado pensar una escena puesto que no tendríamos una idea del nivel de dificultad de esta. Además, la práctica reunía todas las funciones que se debían aplicar, lo cual ha resultado muy cómodo a la hora de enfrentarse a cada etapa. Finalmente, decir que la bibliografía elegida para esta asignatura es totalmente adecuada ya que en algún momento en el que no se puede avanzar, buscando la información en ella puedes solucionar el problema rápidamente. Es por eso que esta práctica también fomenta el auto aprendizaje ya que cuando tienes una idea de una cosa y no sabes como realizarla, es muy fácil buscar en internet y encontrar todo lo que se necesita.

En resumen, esta práctica ha sido completamente de mi agrado ya que se ha trabajado con unos temas que me apasionan como son los gráficos por ordenador y además considero que ha sido una buena forma de ampliar y reforzar, no sólo los conocimientos de la asignatura; sino también las competencias adquiridas.

Bibliografía

Para la realización de la práctica se ha hecho uso del material proporcionado por la asignatura de Informática Gráfica de la Universitat de les Illes Balears. Además, se ha hecho uso de las siguientes páginas:

https://www.opengl.org/resources/libraries/glut/spec3/node12.html

https://www.khronos.org/registry/OpenGL-Refpages/es1.1/xhtml/glLight.xml

https://www.khronos.org/registry/OpenGL-Refpages/gl2.1/xhtml/gluLookAt.xml

https://stackoverflow.com/questions/49236745/opengl-translation-before-and-after-a-rotation

https://learnopengl.com/Getting-started/Textures

https://open.gl/textures

https://www.khronos.org/opengl/wiki/Texture

https://opengl-notes.readthedocs.io/en/latest/topics/texturing/aliasing.html