UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE INGENIERÍA

DIVISIÓN DE INGENIERÍA ELÉCTRICA

LABORATORIO DE COMPUTACIÓN GRÁFICA E INTERACCIÓN HUMANO-COMPUTADORA

GRUPO: 8

PROFESOR: CARLOS ALDAIR ROMAN BALBUENA

FECHA DE ENTREGA: 22-01-2020

MANUAL TÉCNICO

ALUMNO:

MARTÍNEZ RAMÍREZ PABLO CÉSAR

Introducción y Objetivos:

El proyecto presente es un trabajo hecho en Open GL, en este documento se planea hacer una recopilación del flujo de trabajo y explicar de manera puntual las funciones más importantes y complejas a nivel técnico, considerando que se tienen los conocimientos básicos de programación y de computación gráfica.

El proyecto presente se basó en el crustáceo cascarudo, el objetivo fue modelar su fachada y replicar algunos objetos que tenía este en su interior para recrear la escena ficticia, además se planeó implementar animaciones de distintos tipos a los modelos generados.

Flujo de trabajo:

Primero se tuvieron en cuenta los modelos a recrear y después se prosiguió al modelado con ayuda de un software especializado para el modelado, blender, para este proyecto se utilizó blender 2.90.1, el cual para las implementaciones hechas no generó ningún problema.

Una vez que se modelaron los objetos se buscaron texturas, algunas incluso fueron modificadas, ya que no tenían el color deseado, no eran cuadradas o el tamaño de la imagen no era potencia de 2. Cabe aclarar que durante el desarrollo del proyecto se verificaron estos errores por las texturas, ya que al cargarlas en Open GL se distorsionaban de una manera muy notoria. Cuando se estaban texturizando los objetos se utilizaron distintos tipos de mapeados, pero principalmente el cúbico y de manera manual, por ejemplo, para texturizar el bote de la caja registradora se siguió el procedimiento manual, ya que no había texturas como tal para el modelo, es por eso que se optó por tomar la textura de una imagen con intenciones muy distintas.

Una vez que se obtuvieron los modelos texturizados en blender se unieron, duplicaron, escalaron y trasladaron de tal modo que estuvieran en contexto unos con otros.

Para evitar la carga extra de modelos, todos los modelos que no tenían animación se unieron en uno sólo y se exportaron en formato “.obj”. Los archivos con extensión “.obj” traen consigo un archivo “.mtl”, archivo en el cual se indica la ruta de las texturas, sin embargo, hablamos de una ruta absoluta, para contar con los archivos en conjunto se modificó la ruta por una ruta relativa. Después de exportar el modelo estático, se exportaron los modelos que iban a estar animados, de este modo iba a ser posible contemplar un escenario con animaciones en algunos de sus objetos.

Después de exportar los modelos, se importaron a OpenGL y gracias a la exportación no fue necesario hacer cambios de escala ni traslaciones, ya que todo estaba en orden y acomodado previamente a la exportación.

Implementación de animaciones en Open GL:

En el proyecto implementamos las animaciones como transformaciones que se le aplican a un modelo después de presionar una tecla. Es decir, no es necesario mantenerla presionada, además hicimos la consideración de que si una animación estaba en proceso, no se podía interrumpir, es necesario que termine la animación de un objeto antes de que haga otra.

La primera animación se le aplicó a una almeja, dicha almeja rotaba sobre el origen y volvía a su lugar original después de dar toda la vuelta. La implementación fue la siguiente:

```
//Almeja
model = glm::mat4(1.0f);
model = glm::translate(model, glm::vec3(0.0f, 0.0f, 0.0f));
model = glm::rotate(model, glm::radians(anguloAlmeja), glm::vec3(0.0f, 1.0f, 0.0f));
glUniformMatrix4fv(modelLoc, 1, GL_FALSE, glm::value_ptr(model));
almeja.Draw(lightingShader);
```

Como apreciamos, parece ser sólo el dibujo de un modelo, lo único relevante es que tiene una rotación asociada a una variable. Cuando se presiona la tecla que activa la animación entonces se levanta una bandera y se realiza lo siguiente:

```
if (anguloAlmeja < 360)
    anguloAlmeja += 0.3;
else
{
    anguloAlmeja = 0;
    rAlmeja = false;
}
```

Se incrementa la variable asociada al ángulo de rotación del modelo de la almeja, así constantemente hasta que haya completado una vuelta, de este modo la almeja vuelve a la misma posición en la que estaba.

Otra animación simple que se implementó fue el abrir y cerrar de un cajón, el dibujado del modelo se muestra a continuación:

```
model = glm::mat4(1.0f);
model = glm::translate(model, glm::vec3(0.0f, 0.0, -tCajon));
glUniformMatrix4fv(modelLoc, 1, GL_FALSE, glm::value_ptr(model));
cajon.Draw(lightingShader);
```

Como observamos, de igual modo hay una variable asociada a la traslación del cajón, esta variable cambia cada que se activa la tecla para abrir o cerrarlo.

Se implementó la animación del movimiento de una lámpara, este movimiento se dio en un solo eje, además junto con la lámpara se trasladó un point light, para que diera el efecto de que la lámpara emanaba luz. El dibujado del modelo se muestra a continuación:

```
model = glm::mat4(1.0f);
model = glm::translate(model, glm::vec3(-tLampara, 0.0f, 0.0f));
glUniformMatrix4fv(modelLoc, 1, GL_FALSE, glm::value_ptr(model));
lampara.Draw(lightingShader);
```

En el eje “x” la lámpara está siendo trasladada, esta variable “tLampara” está inicializada en cero para que posteriormente sea modificada y surta efecto en el modelo. Esta animación es más compleja, ya que consideramos incrementos en un eje, sin embargo también puede haber incrementos en sentido contrario y aunado a eso se debe detectar cuando la tecla ha sido presionada y si ha sido presionada se verifica si el modelo está en movimiento o no. La implementación se muestra en seguida y en código se hacen las consideraciones mencionadas.

```
if (iLampara)
	{
		if (!mLampara)
		{

			if (dirLampara)
			{
				tLampara += 0.1;
			}
			else
			{
				tLampara -= 0.1;
			}
			mLampara = true;
		}
		else
		{
			if (dirLampara)
			{
				if (tLampara < 14)
				{
					tLampara += 0.1;
				}
				else
				{
					tLampara = 14;
					mLampara = false;
					dirLampara = false;
```

```
                                        iLampara = false;
                                }

                        }
                        else
                        {
                                if (tLampara > 0)
                                {
                                        tLampara -= 0.1;
                                }
                                else
                                {
                                        tLampara = 0;
                                        mLampara = false;
                                        dirLampara = true;
                                        iLampara = false;
                                }
                        }

                }
        }
```

La animación más compleja que se hizo fue trasladar un recipiente de aderezo de una mesa a otra, sin embargo el movimiento estaba descrito por dos rectas. Para esta implementación se debe tomar en cuenta la ecuación de la recta:

$$Y = mx + b$$

En este caso la ordenada al origen no es relevante, en la implementación lo útil es la pendiente. Se desarrolló un algoritmo considerando 2 variables booleanas principales, una definía el sentido en el que avanzaba la primera recta y la segunda para la otra, haciendo sólo esas consideraciones y que la animación debía de detenerse sólo cuando llegara a un extremo u otro se realizó el siguiente código:

```
if (iAderezo1)
        {
                if (!mAderezo1)
                {

                        if (dir1Aderezo1)
                        {
                                xAderezo1 -= dxAderezo1;
```

```
                yAderezo1 += (7 * dxAderezo1) / 9.75;
        }
        else
        {
                xAderezo1 += dxAderezo1;
                yAderezo1 += (7 * dxAderezo1) / 9.75;
        }
        mAderezo1 = true;
}
else
{
        if (dir1Aderezo1)
        {
                if (xAderezo1 > -9.75)
                {
                        xAderezo1 -= dxAderezo1;
                        yAderezo1 += (7 * dxAderezo1) / 9.75;
                }
                else
                {
                        xAderezo1 = -9.75;
                        yAderezo1 = 7;
                        dir1Aderezo1 = false;
                }
                mAderezo1 = true;
        }
        else if (dir2Aderezo1 && bAderezo1)
        {
                if (xAderezo1 > -19.5)
                {
                        xAderezo1 -= dxAderezo1;
                        yAderezo1 -= (7 * dxAderezo1) / 9.75;
                        mAderezo1 = true;
                }
                else
                {
                        xAderezo1 = -19.5;
                        yAderezo1 = 0;
                        dir2Aderezo1 = false;
                        iAderezo1 = false;
                        mAderezo1 = false;
                }
        }
        else if (!dir2Aderezo1)
        {
                if (xAderezo1 < -9.75)
                {
                        xAderezo1 += dxAderezo1;
                        yAderezo1 += (7 * dxAderezo1) / 9.75;
```

```
                        }
                        else
                        {
                                xAderezo1 = -9.75;
                                yAderezo1 = 7;
                                dir2Aderezo1 = true;
                                bAderezo1 = false;
                        }
                        mAderezo1 = true;

                }
                else if (!dir1Aderezo1)
                {

                        if (xAderezo1 < 0)
                        {
                                xAderezo1 += dxAderezo1;
                                yAderezo1 -= (7 * dxAderezo1) / 9.75;
                        }
                        else
                        {
                                xAderezo1 = 0;
                                yAderezo1 = 0;
                                dir1Aderezo1 = true;
                                dir2Aderezo1 = true;
                                iAderezo1 = false;
                                bAderezo1 = true;
                        }
                }

        }
}
```