Proyecto Final

Manual Técnico

Alumno: 316332046

Grupo: 4

Semestre: 2022-2

Fecha de entrega: 27/05/2022

Contenido

Objetivo

El alumno deberá aplicar y demostrar los conocimientos adquiridos durante todo el curso de Computación Gráfica e Interacción Humano-Computadora.

Alcance y limitantes

Con el fin de cumplir el objetivo de este proyecto, se planteó recrear un espacio de nuestra preferencia contando con una habitación, diferentes objetos, también se añadieron animaciones para algunos objetos, etc. Todo esto mediante el uso de la herramienta OpenGL y el uso de diferentes conocimientos al igual de materiales adquiridos durante el curso.

La habitación recreada debe tener siete objetos en 3D que concuerden con el diseño artístico del escenario. Cada uno de estos objetos esta correctamente texturizado para que se vean lo más realista posible de acuerdo con la imagen de referencia.

Por otro lado, con el fin de darle más realismo al escenario representado se añadieron cinco animaciones donde tres son sencillas y dos son complejas, dichas animaciones tienen un contexto y se comportan en armonía con el entorno que los rodea.

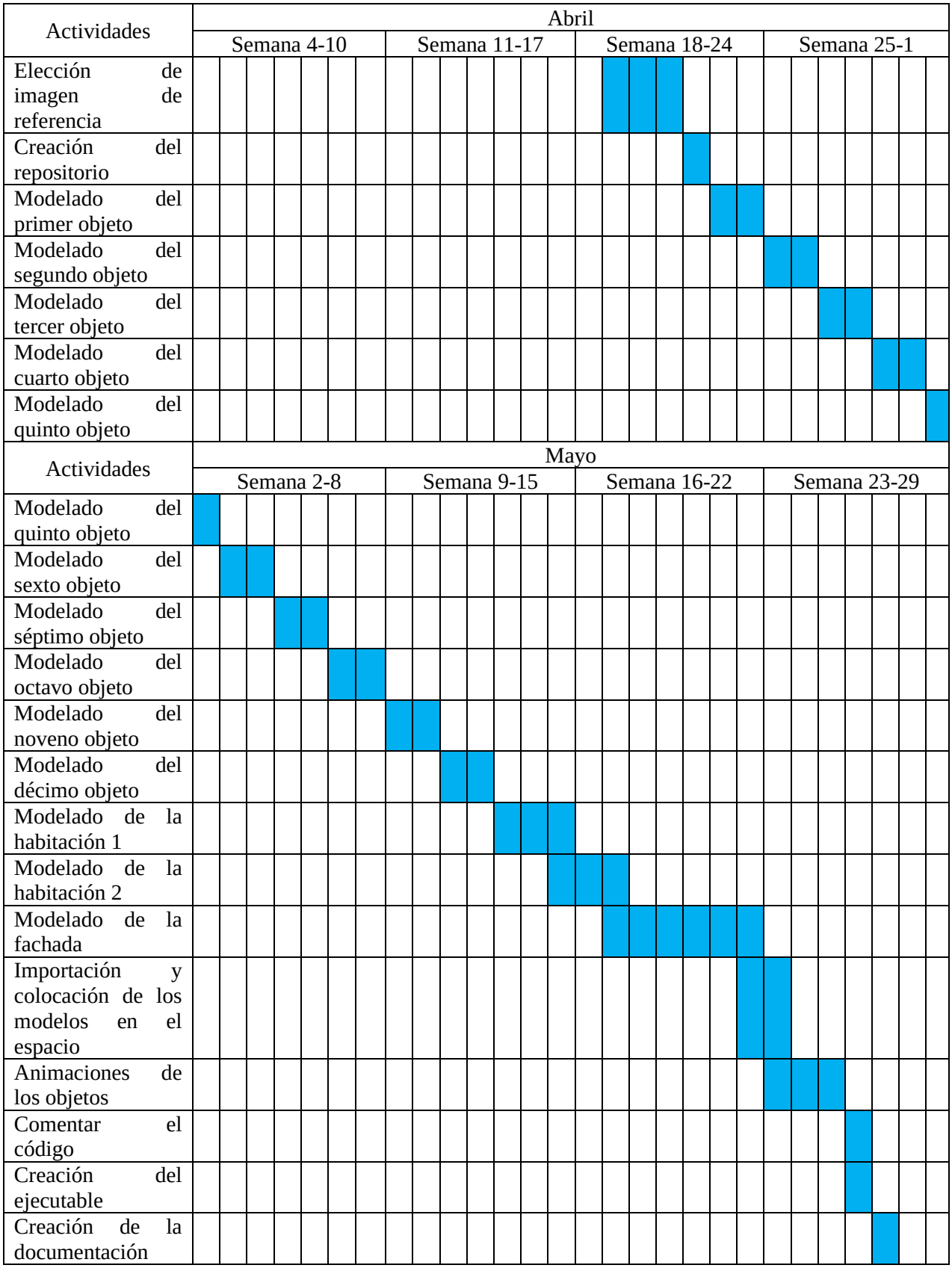
El proyecto no contará con la recreación completa del interior de la fachada solo se contará con la habitación que se haya decidió trabajar, otro aspecto a considerar es que el interior de la habitación cuenta con lo básico para representar la estructura, pero solo contará con los siete objetos que se seleccionaron para recrear por lo que la habitación no tendrá elementos extras salvo puertas o ventanas que son parte de la estructura, lo mismo aplica para la fachada.

Cronograma de actividades

Con el fin de distribuir el trabajo para poder cumplir con los objetivos del proyecto, así como evitar una sobresaturación de trabajo en los últimos días se creó un cronograma de las actividades a realizar que incluyera la fecha límite.

Como mi estilo de trabajo es un poco caótico ya que suelo realizar grandes porciones de trabajo de proyectos los fines de semana por lo que no suelo subdividir demasiado las tareas, en su lugar suelo dividir las actividades en un tamaño mediano que están compuestas por temas relacionados entre sí lo que me ayuda a establecer una relación entre dichas tareas permitiéndome ahorrar tiempo.

Nota: La celda de semana se subdivide en siete columnas, cada una corresponde con el día de la semana empezando por lunes y terminando en domingo.


Actividades
Abril
Semana 4-10
Semana 11-17
Semana 18-24
Semana 25-1
Elección de imagen de referencia
Creación del repositorio
Modelado del primer objeto
Modelado del segundo objeto
Modelado del tercer objeto
Modelado del cuarto objeto
Modelado del quinto objeto
Actividades
Mayo
Semana 2-8
Semana 9-15
Semana 16-22
Semana 23-29
Modelado del quinto objeto
Modelado del sexto objeto
Modelado del séptimo objeto
Modelado del octavo objeto
Modelado del noveno objeto
Modelado del décimo objeto
Modelado de la habitación 1
Modelado de la habitación 2
Modelado de la fachada
Importación y colocación de los modelos en el espacio
Animaciones de los objetos
Comentar el código
Creación del ejecutable
Creación de la documentación

Análisis de costos

Para calcular el costo de las actividades realizadas se tomará que cada hora de trabajo en 118 pesos. Cabe aclarar que los preciso son en moneda mexicana

Actividades realizadas	Horas	Precio $
Diseño del primer objeto	8	944
Diseño del segundo objeto	6	708
Diseño del tercer objeto	4	472
Diseño del cuarto objeto	4	472
Diseño del quinto objeto	3	354
Diseño del sexto objeto	3	354
Diseño del séptimo objeto	4	472
Diseño del octavo objeto	8	944
Diseño del noveno objeto	4	472
Diseño del décimo objeto	3	354
Diseño de la habitación 1	12	1416
Diseño de la habitación 2	12	1416
Diseño de la fachada	24	2832
Texturizado de los modelos	10	1180
Creación del código para implementación de los objetos	7	826
Creación de las animaciones de los objetos	14	1652
Creación de la documentación del proyecto	5	590
Total		15458

Otros costos y herramientas	Precio $	Precio Venta $
Licencia MAYA por 1 mes	1963.36	2420.45
Planeación del proyecto	1000	1250
Programación de todo el proyecto	4500	5625
Internet al mes	469	703.5
Electricidad al mes	200	350
Total		10348.95

Considerando una holgura del 5% por posibles gastos inesperados

Total = (15458+10348.95) +1290.35 =27097.3

Costo total del proyecto: $27097.3

Desarrollo de las actividades

Como el espacio a diseñar se basó en un videojuego se tuvo la ventaja de poder ver más detalladamente los objetos que se iban a recrear por lo que se tuvo un mejor resultado a la hora de modelar los objetos por lo que se obtuvieron resultados bastantes similares.

El problema de esto es que me veo limitado a la movilidad que ofrece el juego en cuanto a la cámara ya que en este juego no está pensado para la exploración impidiendo poder observar a detalle la fachada del dormitorio, en concreto la parte trasera debido a que no me era posible moverme alrededor del edificio por lo que tuve que imaginar el diseño de la parte trasera, no obstante para el resto de objetos se pudieron analizar sin mayor problema salvo mi problema para poder diferenciar los colores que tenían los objetos.

Para el texturizado se trató de conseguir el color más parecido al observado por medio de la pantalla, se usaron programas como GIMP para poder crear las texturas, así como editarlas para poder añadir el canal Alpha que servirá para el efecto de la transparencia, por medio de Maya se editaron los mapas UV para poder texturizar de manera correcta el objeto.

En el caso de las animaciones como se crearon cinco que tuvieran relación con el objeto evitando crear situaciones fuera de contexto.

- Primera animación: Puerta que se abre y se cierra
  En esta primera animación se cuenta con una puerta que se abre y cierra constantemente.
- Segunda animación: Tapa de laptop que se abre y cierra.
  Aquí se cuenta con una animación que abre y cierra de forma constante la tapa de la laptop.
- Tercera animación: Ventanas del cuarto que se cierran y abren.
  En esta ocasión se realiza la acción de abrir y cerrar las ventanas en sentido vertical de la habitación.
- Cuarta animación: Pájaro que vuela siguiendo una trayectoria recta y diagonal.
  El ave durante la animación sigue una trayectoria de vuelo en línea recta que varía en varios puntos pasando a un desplazamiento en diagonal, todo esto mientras mueve las alas para simular el comportamiento de un ave mientras vuela.
- Quinta animación: Conejo que se desplaza por medio de saltos.
  El conejo durante su animación se desplaza por medio de saltos siguiendo un recorrido en línea recta y diagonal, la acción al ser un salto tiene una trayectoria en parábola en la que el conejo mueve sus extremidades para simular un comportamiento más realista ya sea al saltar o cuando está a punto de caer.

Documentación de código

- Edición de la aplicación:
  Ahora es momento de explicar el código que se empleó, así como señalar que se puede modificar para cambiar el comportamiento del programa ya sea que se quiera ampliar el alcance del proyecto o quitar elementos que no se necesiten.
  En la línea 147 se comienza con la carga de los modelos a usar para recrear la fachada, la habitación y los objetos con la respectiva ruta de cada modelo

```
// Load models
	// Fachada
	Model barandillasMadFachada((char*)"Models/Fachada/Barandillas_Madera_Fachada.obj");
	Model barandillasMetFachada((char*)"Models/Fachada/Barandillas_Metal_Fachada.obj");
	Model baseFachada((char*)"Models/Fachada/Base_Fachada.obj");
	Model chimeneasFachada((char*)"Models/Fachada/Chimeneas_Fachada.obj");
	Model columnasFachada((char*)"Models/Fachada/Columnas_Fachada.obj");
	Model detallesFachada((char*)"Models/Fachada/Detalles_Fachada.obj");
	Model escalerasFachada((char*)"Models/Fachada/Escaleras_Fachada.obj");
	Model marcosFachada((char*)"Models/Fachada/Marcos_Fachada.obj");
	Model murosFachada((char*)"Models/Fachada/Muros_Fachada.obj");
	Model puertaAFachada((char*)"Models/Fachada/PuertaA_Fachada.obj");
	Model puertaB1Fachada((char*)"Models/Fachada/PuertaB1_Fachada.obj");
	Model puertaB2Fachada((char*)"Models/Fachada/PuertaB2_Fachada.obj");
	Model puertaSFachada((char*)"Models/Fachada/PuertaS_Fachada.obj");
	Model techoFachada((char*)"Models/Fachada/Techo_Fachada.obj");
	Model vidriosFachada((char*)"Models/Fachada/Vidrios_Fachada.obj");

	// Habitacion_1
	Model marcosH1((char*)"Models/Habitacion_1/Marcos_H1.obj");
	Model muroAtrH1((char*)"Models/Habitacion_1/Muro_Atr_H1.obj");
	Model muroDerH1((char*)"Models/Habitacion_1/Muro_Der_H1.obj");
	Model muroFreH1((char*)"Models/Habitacion_1/Muro_Fre_H1.obj");
	Model muroIzqH1((char*)"Models/Habitacion_1/Muro_Izq_H1.obj");
	Model pisoH1((char*)"Models/Habitacion_1/Piso_H1.obj");
	Model puertaH1((char*)"Models/Habitacion_1/Puerta_H1.obj");
	Model roperoH1((char*)"Models/Habitacion_1/Ropero_H1.obj");
	Model techoH1((char*)"Models/Habitacion_1/Techo_H1.obj");
	Model ventana1AH1((char*)"Models/Habitacion_1/VentanaA_H1.obj");
	Model ventana1BH1((char*)"Models/Habitacion_1/VentanaB_H1.obj");
	Model vidrioVentana1AH1((char*)"Models/Habitacion_1/Vidrio_VentanaA_H1.obj");
	Model vidrioVentana1BH1((char*)"Models/Habitacion_1/Vidrio_VentanaB_H1.obj");
```

Se recomienda que si se modifica la ruta de los modelos también se actualice la ruta del código ya que si bien no impedirá la ejecución del programa si evitará que se cargue dicho modelo. Por otro lado, si se desea evitar que se muestre algún objeto en concreto es necesario comentar con // todas las líneas de *Model* que constituyan a dicho objeto.

```
Model ventana2AH1((char*)"Models/Habitacion_1/VentanaA_H1.obj");
Model ventana2BH1((char*)"Models/Habitacion_1/VentanaB_H1.obj");
Model vidrioVentana2AH1((char*)"Models/Habitacion_1/Vidrio_VentanaA_H1.obj");
Model vidrioVentana2BH1((char*)"Models/Habitacion_1/Vidrio_VentanaB_H1.obj");

//Objetos Habitacion 1
Model camaH1((char*)"Models/Objetos_H1/Cama/Cama_H1.obj");
Model laptop1H1((char*)"Models/Objetos_H1/Laptop/Laptop1_H1.obj");
Model laptop2H1((char*)"Models/Objetos_H1/Laptop/Laptop2_H1.obj");
Model estanteH1((char*)"Models/Objetos_H1/Estante/Estante_H1.obj");
Model escritorioH1((char*)"Models/Objetos_H1/Escritorio/Escritorio_H1.obj");
Model libros1H1((char*)"Models/Objetos_H1/Libros/Libros1_H1.obj");
Model libros2H1((char*)"Models/Objetos_H1/Libros/Libros2_H1.obj");
Model sillaH1((char*)"Models/Objetos_H1/Silla/Silla_H1.obj");
Model sofaH1((char*)"Models/Objetos_H1/Sofa/Sofa_H1.obj");

//Habitación_2
Model paredDelH2((char*)"Models/Habitacion_2/Pared_Del_H2.obj");
Model paredDerH2((char*)"Models/Habitacion_2/Pared_Der_H2.obj");
Model paredIzqH2((char*)"Models/Habitacion_2/Pared_Izq_H2.obj");
Model paredTraH2((char*)"Models/Habitacion_2/Pared_Tra_H2.obj");
Model persiana1H2((char*)"Models/Habitacion_2/Persiana1_H2.obj");
Model persiana2H2((char*)"Models/Habitacion_2/Persiana2_H2.obj");
Model pisoH2((char*)"Models/Habitacion_2/Piso_H2.obj");
Model puertaH2((char*)"Models/Habitacion_2/Puerta_H2.obj");
Model roperoH2((char*)"Models/Habitacion_2/Ropero_H2.obj");
Model techoH2((char*)"Models/Habitacion_2/Techo_H2.obj");
Model ventanaMarcoH2((char*)"Models/Habitacion_2/Ventana_Marco_H2.obj");
Model ventanaVidrioH2((char*)"Models/Habitacion_2/Ventana_Vidrio_H2.obj");

//Objetos Habitacion 2
Model camaH2((char*)"Models/Objetos_H2/Cama/Cama_H2.obj");
Model comodaH2((char*)"Models/Objetos_H2/Comoda/Comoda_H2.obj");
Model escritorioH2((char*)"Models/Objetos_H2/Escritorio/Escritorio_H2.obj");
Model sillaH2((char*)"Models/Objetos_H2/Silla/Silla_H2.obj");
Model sofaH2((char*)"Models/Objetos_H2/Sofa/Sofa_H2.obj");
```

Finalmente, si se desea agregar un objeto creado, basta con mover el objeto a la carpeta del proyecto y crear la instancia en el código con la ruta en la que esté el objeto en cuestión; se recomienda que la textura del objeto se encuentre en la misma carpeta para evitar posibles fallos.

En la línea 271 se establecen los parámetros para establecer la iluminación que tendrá el ambiente por lo que si desea modificarla se recomienda modificar los valores de *dirLight.ambient* y *dirLight.diffuse* tomando en cuenta que 1.0 es el máximo y 0.1 el mínimo. El resto de los valores se recomienda evitar cambiar su valor salvo que lo crea necesario y sepa lo que está haciendo.

```
// Directional light
            glUniform3f(glGetUniformLocation(lightingShader.Program,
"dirLight.direction"), -0.2f, -1.0f, -0.3f);
            glUniform3f(glGetUniformLocation(lightingShader.Program, "dirLight.ambient"),
0.4f, 0.4f, 0.4f);
            glUniform3f(glGetUniformLocation(lightingShader.Program, "dirLight.diffuse"),
0.4f, 0.4f, 0.4f);
            glUniform3f(glGetUniformLocation(lightingShader.Program, "dirLight.specular"),
1.0f, 1.0f, 1.0f);
```

De la línea 294 a la línea 669 se empiezan a cargar las características de las instancias *Model* definiendo en algunos casos su posición, rotación o escala. En este proyecto por medio del programa Maya se estableció la escala y posición del objeto respecto al espacio para evitar un costo demasiado alto de recursos que si se hacía de forma directa en OpenGL por lo que se recomienda que si quiere ubicar el nuevo objeto sin animación lo haga por medio de Maya. Si el objeto en cuestión se va a animar se recomienda que lo haga por medio de transformaciones básicas en el código y solo en Maya se encargue de establecer la escala y el pivote del objeto, todo esto para evitar tener fallos por un mal manejo del pivote lo que afectaría la animación.

En el siguiente fragmento de código el objeto cama está sin animar y el objeto laptop si se anima ocasionando que este último cuente con transformaciones básicas.

```
//Carga de modelos de objetos de la Habitacion_1
            //Cama
            model = glm::mat4(1);
            glUniformMatrix4fv(modelLoc, 1, GL_FALSE, glm::value_ptr(model));
            glUniform1i(glGetUniformLocation(lightingShader.Program,
"activaTransparencia"), 0);
            camaH1.Draw(lightingShader);

            //Laptop
            model = glm::mat4(1);
            model = glm::translate(model, glm::vec3(12.008f, 4.827f, -1.868f));
            model = glm::rotate(model, glm::radians(47.256f), glm::vec3(0.0f, 1.0f,
0.0f));
            model = glm::rotate(model, glm::radians(laptop1H1Rot), glm::vec3(1.0f, 0.0f,
0.0f));
            glUniformMatrix4fv(modelLoc, 1, GL_FALSE, glm::value_ptr(model));
            glUniform1i(glGetUniformLocation(lightingShader.Program,
"activaTransparencia"), 0);
            laptop1H1.Draw(lightingShader);

            model = glm::mat4(1);
            model = glm::translate(model, glm::vec3(12.008f, 4.827f, -1.868f));
            model = glm::rotate(model, glm::radians(47.256f), glm::vec3(0.0f, 1.0f,
0.0f));
            glUniformMatrix4fv(modelLoc, 1, GL_FALSE, glm::value_ptr(model));
            glUniform1i(glGetUniformLocation(lightingShader.Program,
"activaTransparencia"), 0);
            laptop2H1.Draw(lightingShader);
```

De la línea 671 a la 720 se cargan de igual forma las características de las instancias *Model* con la diferencia que estos objetos tienen características de grados de transparencia como lo es un vidrio para una ventana permitiendo ver lo que hay del otro lado. Si desea modificar el nivel de transparencia basta con cambiar el cuarto valor del parámetro *colorAlfa* teniendo en cuenta que 1.0 es el máximo y 0.1 es el mínimo.

```
//Codigo para objetos semitrasnparentes
              glEnable(GL_BLEND);//Activa la funcionalidad para trabajar el canal alfa
              glBlendFunc(GL_SRC_ALPHA, GL_ONE_MINUS_SRC_ALPHA);

              //Carga de modelo de Habitacion_1
              model = glm::mat4(1);
              model = glm::translate(model, glm::vec3(12.209f, 5.525f, -1.548f));
              glUniformMatrix4fv(modelLoc, 1, GL_FALSE, glm::value_ptr(model));
              glUniform1i(glGetUniformLocation(lightingShader.Program,
"activaTransparencia"), 1);
              glUniform4f(glGetUniformLocation(lightingShader.Program, "colorAlfa"), 1.0f,
1.0f, 1.0f, 0.5f);
              vidrioVentana1AH1.Draw(lightingShader);

              model = glm::mat4(1);
              model = glm::translate(model, glm::vec3(12.209f, 5.525f+ventanaBH1Tra, -
1.548f));
              glUniformMatrix4fv(modelLoc, 1, GL_FALSE, glm::value_ptr(model));
              glUniform1i(glGetUniformLocation(lightingShader.Program,
"activaTransparencia"), 1);
              glUniform4f(glGetUniformLocation(lightingShader.Program, "colorAlfa"), 1.0f,
1.0f, 1.0f, 0.5f);
              vidrioVentana1BH1.Draw(lightingShader);
```

En la función *DoMovement()* se maneja la entrada por medio de las teclas que afecta el movimiento de la cámara con w, s, a, d, etc. Como ya se ha explicado en el uso de la aplicación. Si desea cambar la tecla que realiza estos movimientos solo se necesita modificar la condición estableciendo la tecla de su preferencia.

Por otra parte, la función al ser llamada de manera constante sirve para definir las animaciones que requiera ya sea de nuevos objetos que decida agregar o aquellos que ya se encuentran en el proyecto. Si decide eliminar una animación en particular puede optar por comentarla o desactivar la tecla de dicha animación, se explicará cómo realizarlo más adelante.

```
void DoMovement()
{
	// Camera controls
	if (keys[GLFW_KEY_W] || keys[GLFW_KEY_UP])
	{
		camera.ProcessKeyboard(FORWARD, deltaTime);
	}

	if (keys[GLFW_KEY_S] || keys[GLFW_KEY_DOWN])
	{
		camera.ProcessKeyboard(BACKWARD, deltaTime);
	}

	if (keys[GLFW_KEY_A] || keys[GLFW_KEY_LEFT])
	{
		camera.ProcessKeyboard(LEFT, deltaTime);
	}

	if (keys[GLFW_KEY_D] || keys[GLFW_KEY_RIGHT])
	{
		camera.ProcessKeyboard(RIGHT, deltaTime);
	}
```

La función *KeyCallback()* se llama cada vez que una tecla se presiona o se suelta lo que resulta ser muy útil a la hora de indicar si se activa o desactiva una animación.
Para definir una nueva animación basta con crear una condición en la que se indique la tecla que activara dicha animación, así como crear una variable tipo *bool* que permita verificar que se cumple la condición.
Como se mencionó antes para deshabilitar una animación basta con quitar la condición que la activa o cambiar el contenido de la condición por la nueva animación que quiera añadir

```
void KeyCallback(GLFWwindow* window, int key, int scancode, int action, int mode)
{
	if (GLFW_KEY_ESCAPE == key && GLFW_PRESS == action)
	{
		glfwSetWindowShouldClose(window, GL_TRUE);
	}

	if (key >= 0 && key < 1024)
	{
		if (action == GLFW_PRESS)
		{
			keys[key] = true;
		}
		else if (action == GLFW_RELEASE)
		{
			keys[key] = false;
		}
	}
```

```
	if (keys[GLFW_KEY_1])
	{
		puertaH1Anim = !puertaH1Anim;
	}

	if (keys[GLFW_KEY_2])
	{
		laptop1H1Anim = !laptop1H1Anim;
	}
	if (keys[GLFW_KEY_3])
	{
		ventanaBH1Anim = !ventanaBH1Anim;
	}

	if (keys[GLFW_KEY_4])
	{
		pajaroAnim = !pajaroAnim;
	}

	if (keys[GLFW_KEY_5])
	{
		conejoAnim = !conejoAnim;
	}
}
```

La función *MouseCallback()* se llama cada vez que se mueve el ratón, sirve para definir el movimiento de la cámara sintética por lo que no se recomienda modificar su código salvo que lo requiera y sepa lo que está haciendo.

```
void MouseCallback(GLFWwindow* window, double xPos, double yPos)
{
	if (firstMouse)
	{
		lastX = xPos;
		lastY = yPos;
		firstMouse = false;
	}

	GLfloat xOffset = xPos - lastX;
	GLfloat yOffset = lastY - yPos;  // Reversed since y-coordinates go from bottom to
left

	lastX = xPos;
	lastY = yPos;

	camera.ProcessMouseMovement(xOffset, yOffset);
}
```

Conclusiones

Con el desarrollo de este proyecto pude cumplir con el objetivo y el alcance establecido al inicio del documento ya que se aplicaron los conocimientos adquiridos por medio de las prácticas desde el modelado de un objeto por medio de código o de software así como su texturizado e iluminación para poder añadir mayor realismo así como el uso de varios conceptos como lo son las transformaciones básicas que permiten cambiar la apariencia del objeto y que puede empelarse para crear animaciones ya sean sencillas o complejas, se usó el modelado jerárquico para dichas animaciones que permite una mayor flexibilidad para trabajar, etc. Por otro lado, se cumplió con el alcance del proyecto ya que se recreó la fachada, la habitación y los 7 objetos elegidos al inicio del proyecto sin olvidar que se añadieron las 5 animaciones donde 3 fueron sencillas y 2 fueron complejas.

Para finalizar, en mi opinión fue un buen curso que me permitió acercarme un poco al desarrollo de aplicaciones relacionadas a la computación gráfica que me ayudó a comprender mejor el proceso que se lleva detrás de varios programas que solemos usar en la vida diaria pero que no solemos darle importancia, sin embargo, ahora puedo valorar y saber cómo como es el proceso que se realiza en cada juego, aplicación gráfica o programa por lo que me siento bastante satisfecho.

Referencias y programas usados.

Programas:

- Maya 2023 para modelado y texturizado de los objetos, también para definir su escala y posición respecto al espacio.
- GIMP para la creación y edición de las texturas.
- Visual Studio para el desarrollo del código del programa para poder recrear el espacio a modelar.
- Gitkraken para el manejo del repositorio de github

Referencias:

- Textures for 3D, graphic design and Photoshop!. Consultado el 23 de mayo de 2022, en https://www.textures.com/
- 3D Models for Professionals. Consultado el 23 de mayo de 2022, en https://www.turbosquid.com/
- Freepik| Recursos gráficos para todos. Consultado el 23 de mayo de 2022, en https://www.freepik.es/
- PILIIGON | Search 4000+ Assets for 3D Artists. Consultado el 23 de mayo de 2022, en https://www.poliigon.com/