Universidad Nacional Autónoma de México
Facultad de Ingeniería

PROYECTO FINAL
MANUAL TÉCNICO

Estudiante : Ceballos Equihua C. Nathaniel
Asignatura : Laboratorio de Computación Gráfica e Interacción Humano-Computadora
Profesor : Román Balbuena Carlos Aldair Ing.
Grupo : 08
Carrera : Ingeniería en Computación - 110
Semestre : 2021-I

Ciudad de México, 22 de enero de 2021

Índice

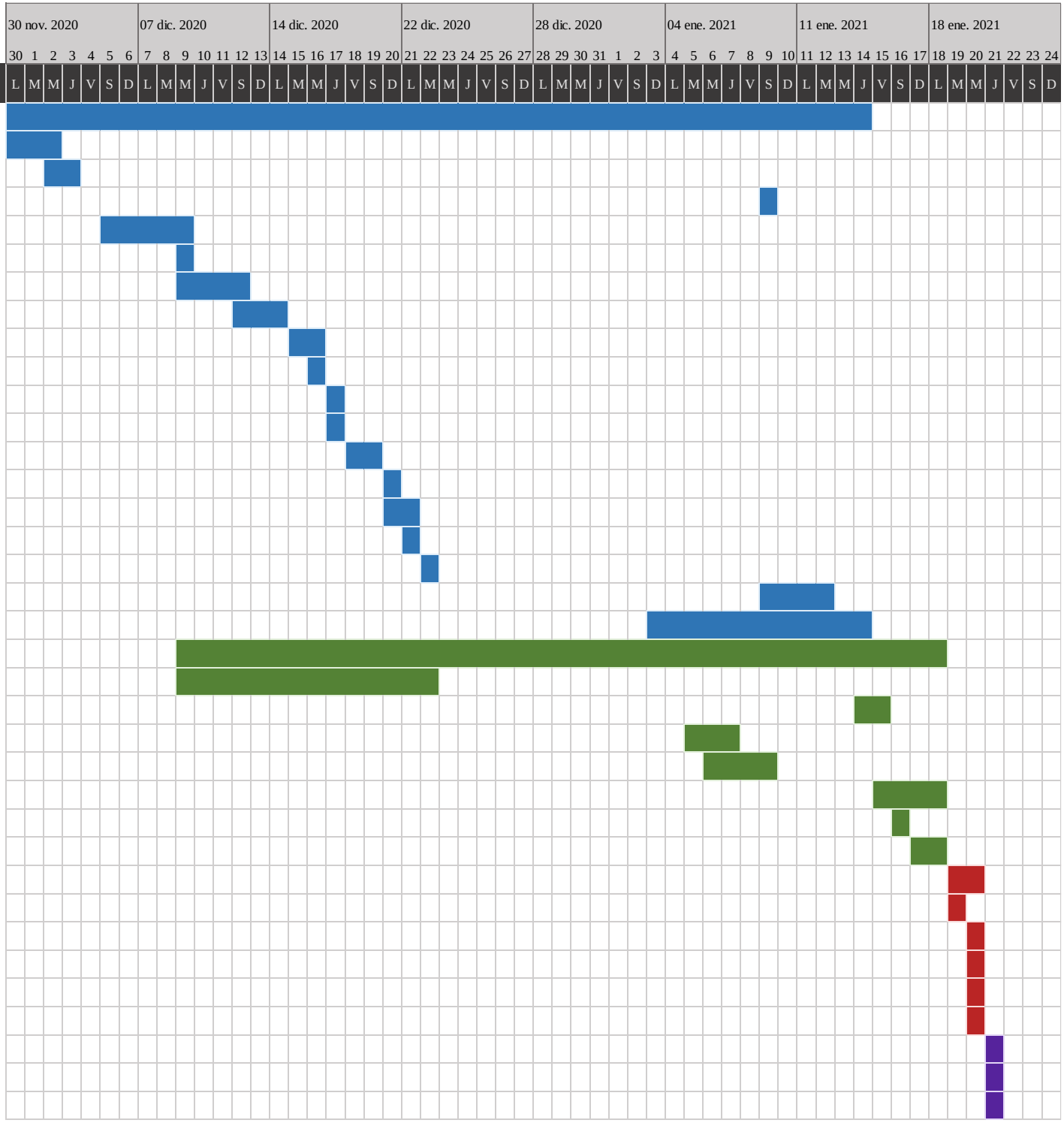
Objetivos

- El alumno deberá aplicar y demostrar los conocimientos adquiridos durante todo el curso.
- El alumno deberá elaborar un proyecto en el que se demuestren los conceptos aprendidos a lo largo del curso de Computación Gráfica e Interacción Humano-Computadora.
- El alumno deberá diseñar un espacio virtual a partir de una referencia, incluyendo una habitación interior con al menos siete objetos diferentes y una fachada, utilizando software de modelado y OpenGL.
- El alumno deberá implementar cinco animaciones diferentes, tres con transformaciones básicas y dos con procedimientos más complejos.

Diagrama de Gantt

Se muestra en la siguiente página.

ACTIVIDAD	INICIO	FINAL
Modelado en Blender	30/11/20	14/01/21
Sofá de salón	30/11/20	02/12/20
Mesa de salón	02/12/20	03/12/20
Lámpara de mesa (con primitivas de OpenGL)	09/01/21	09/01/21
Fuente central	05/12/20	09/12/20
Maceta de planta de interior	09/12/20	09/12/20
Lámpara de techo	09/12/20	12/12/20
Timbre	12/12/20	14/12/20
Botella de licor	15/12/20	16/12/20
Vela	16/12/20	16/12/20
Elevador	17/12/20	17/12/20
Vaso de café	17/12/20	17/12/20
Carretilla para maletas	18/12/20	19/12/20
Tableta	20/12/20	20/12/20
Cojín	20/12/20	21/12/20
Vela transparente	21/12/20	21/12/20
Bote de basura	22/12/20	22/12/20
Interior de la habitación	09/01/21	12/01/21
Fachada	03/01/21	14/01/21
Integración en OpenGL	09/12/20	18/01/21
Carga de modelos de objetos	09/12/20	22/12/20
Carga de fachada y organización de objetos	14/01/21	15/01/21
Manejo de semi-transparencia y texturas/mapas alfa	05/01/21	07/01/21
Implementación de texturas/mapas de normales	06/01/21	09/01/21
Correción de iluminación	15/01/21	18/01/21
Adición de skybox	16/01/21	16/01/21
Detalles menores (MSAA, orden de objetos con transparencia, etc)	17/01/21	18/01/21
Animaciones	19/01/21	20/01/21
Carretilla para maletas (recorrido mediante ecuación)	19/01/21	19/01/21
Vela de la fuente (caída mediante keyframes)	20/01/21	20/01/21
Lámparas de techo (rotación simple)	20/01/21	20/01/21
Sofás (rotación simple, con pivote)	20/01/21	20/01/21
Tableta (traslación y escalamiento)	20/01/21	20/01/21
Documentación	21/01/21	21/01/21
Manual de usuario	21/01/21	21/01/21
Manual técnico	21/01/21	21/01/21



Alcance del proyecto

Recrear un espacio virtual a partir de una referencia, formado por una habitación con al menos siete objetos distintos y una fachada. Las imágenes de referencia son las siguientes:

Modelar al menos siete objetos del interior del cuarto. La lista de objetos es la siguiente:

1. Fuente central.
2. Mesa de salón.
3. Planta de interior.
4. Lámpara de techo.
5. Sofá de salón.

6. Lámpara de mesa.
7. Carretilla para maletas.

Modelar uno de los objetos únicamente utilizando primitivas de OpenGL (en este caso, la lámpara de mesa).

Modelar el resto de los objetos y la fachada utilizando un software de modelado, como Blender o Maya.

Desarrollar el entorno utilizando OpenGL, mediante Visual Studio. Utilizar de base el código proporcionado por el profesor a lo largo de las prácticas de laboratorio, y modificarlo según sea necesario.

Importar, cargar y acomodar todos los modelos en OpenGL, utilizando las transformaciones que sean necesarias, en el orden correcto para obtener los resultados esperados.

Implementar cinco diferentes animaciones que el usuario pueda iniciar y detener. Dos de las animaciones deben ser complejas, es decir, ser descritas mediante una ecuación o utilizando *keyframes*; y el resto deben ser sencillas, es decir, transformaciones simples.

Documentación

Modelos

Los siete modelos principales son los siguientes:

1. Fuente central

2. Mesa de salón

3. Planta de interior

4. Lámpara de techo

5. Sofá de salón

6. Lámpara de mesa (creada con primitivas de OpenGL)

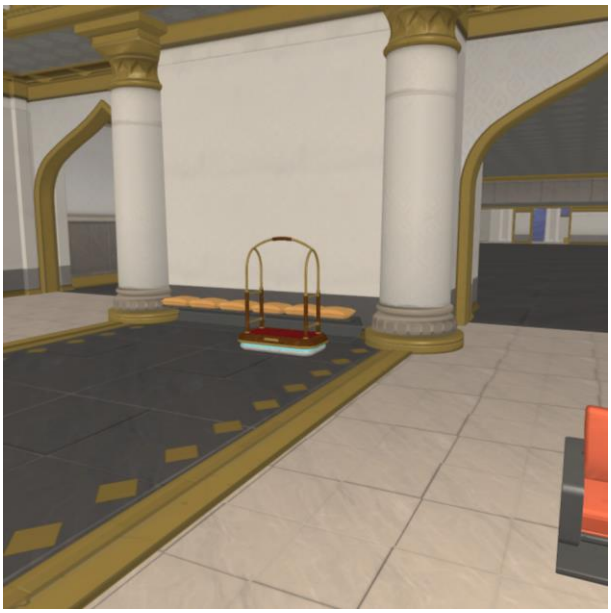
7. Carretilla para maletas

Todos los modelos listados en el diagrama de Gantt se modelaron en Blender, incluyendo la estructura de la habitación y de la fachada, con la excepción de las hojas de la planta, que se obtuvieron de Turbosquid[1]. Por otro lado, la gran mayoría de las texturas pertenecen a Blizzard, pues son las originales del juego de referencia (Overwatch). Las texturas de normales se modificaron, pues las originales estaban en un formato de compresión BC5, por lo que los colores no eran los adecuados. Las texturas de especular se obtuvieron a partir de modificaciones a las texturas de PBR originales, y algunas texturas de transparencia/alfa se modificaron también. Algunos modelos, al no ser fundamentales para el proyecto, como los detalles del interior, algunas columnas y buena parte del porche, son también los originales del juego, con modificaciones.

Animaciones

Las animaciones incluidas en el proyecto se detallan a continuación:

1. Recorrido del carrito para maletas. Se activa y pausa con la tecla 1, el carrito se devuelve a su posición inicial con la tecla 2. Se modeló mediante la ecuación de la recta, y consta de cuatro "estados", pues cambia de dirección para que la trayectoria tenga forma rectangular. Se realizó con base en la práctica 10.



2. Caída de la vela de la fuente. Se activa y detiene en su lugar con la tecla 3, la vela se devuelve a su posición inicial con la tecla 4. Se implementó mediante *keyframes*, y se compone de 4 *frames* precargados en el código. Se realizó con base en la práctica 11.

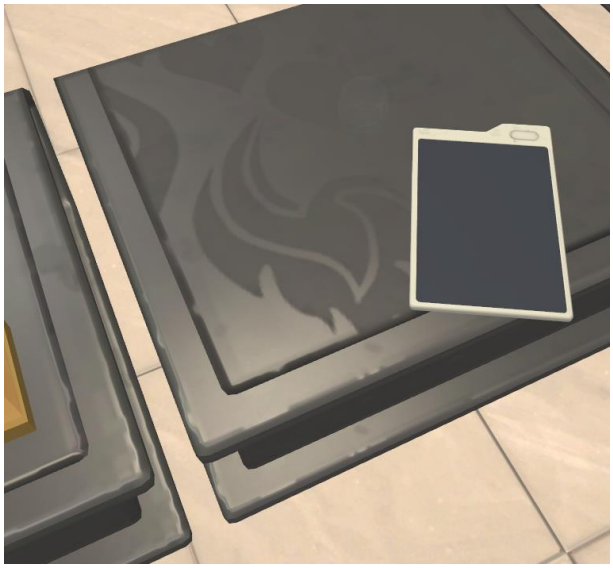
[1] https://www.turbosquid.com/3d-models/free-leaves-3d-model/580785

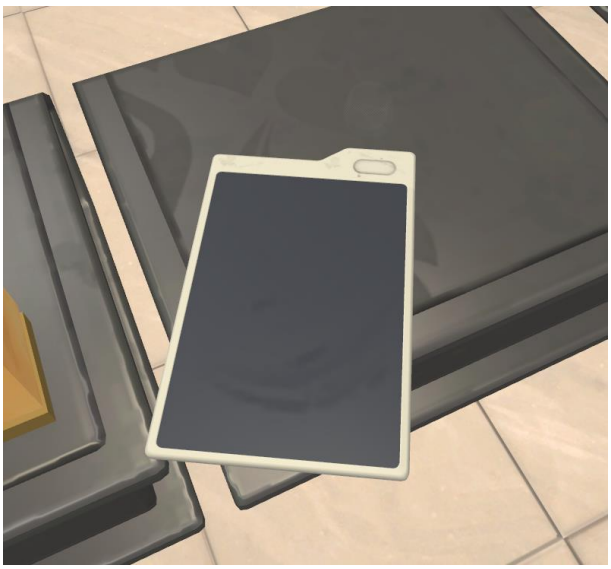

3. Rotación de las lámparas de techo. Se activa con la tecla 5 y las lámparas se devuelven a su rotación inicial con la tecla 6. La variación en la rotación está controlada a partir de la función *glfwGetTime()*, y sólo se devuelve al valor 0.0f al presionar la tecla 6. Se realizó con base en la práctica 9.


4. Rotación con pivote de los sofás cercanos a los arcos de entrada. Se activa (se puede cambiar de dirección, hacia afuera o devuelta a la posición inicial) con la tecla 7, y los sofás se devuelven a su posición inicial con la tecla 8. Se realizó sumando o restando valores pequeños y fijos a las variables que controlan la rotación, según si se cumplen las condiciones necesarias (el valor de dicha variable no supera o es menor a ciertos valores, y la bandera para determinar si la animación debe ejecutarse o no, que cambia según la tecla presionada). Además, como la rotación no es con respecto al centro del

objeto, se trasladó primero el objeto lo necesario para cambiar su origen a su extremo, se realizó la rotación y se trasladó de vuelta a su posición inicial.

5. Escalamiento y traslación de la tableta. Se activa (se puede cambiar de dirección, hacia la izquierda o devuelta a la posición inicial) con la tecla 9, y la tableta se devuelve a su posición y tamaño iniciales con la tecla 0. Se realizó de manera similar a la rotación de los sofás.





Código

Como se mencionó anteriormente, se tomó como base el código trabajado en las prácticas (más específicamente, el código de la práctica 10, al que se le agregaron fragmentos de otras prácticas).

Las funciones principales se muestran a continuación:

- *void initialiseKeyframes(void)*

  Se encarga de inicializar los *keyframes* para la animación de la caída de la vela. Todos los valores son fijos, calculados anteriormente para representar el movimiento de la vela; evidentemente, el primer *frame* contiene los valores correspondientes a la posición inicial de la vela, es decir, de su posición en el entorno. Cabe destacar que la estructura *Keyframe* contiene una variable por eje para la posición, rotación y sus incrementos respectivos, además de una variable para el número de pasos a utilizar en la interpolación.

- *void resetElements(void)*

  Se encarga de restablecer la posición y rotación de la vela a sus valores iniciales. Se utiliza cuando se inicializa la animación.

- *void interpolation(void)*

  Se encarga de realizar la interpolación necesaria para calcular los incrementos entre las posiciones de cada *frame* y obtener un movimiento suave. El número de incrementos define qué tan suave y lenta (o qué tan rápida y "dura") es la animación, a mayor cantidad de incrementos, más suave y lenta es. Estos incrementos se calculan mediante la división de la diferencia de los valores de posición y rotación del *frame* actual y el siguiente entre la cantidad de incrementos.

- *int main()*

  Se trata de la función principal que contiene al programa principal, valga la redundancia. Realiza las configuraciones pertinentes de OpenGL, crea la ventana de la aplicación, carga los modelos a utilizar, define variables importantes (como los vértices de las geometrías a utilizarse con primitivas y los vértices del *skybox*), carga las texturas independientes, define el vector con las coordenadas de los objetos transparentes (necesario para ordenarlos, proceso requerido para asegurar que las semi-transparencias se visualicen de manera correcta) y manda a llamar a la función *initialiseKeyframes* antes de entrar al ciclo principal.

  El ciclo principal, en el cual permanece mientras la ventana de la aplicación no se haya cerrado, se encarga de mandar a llamar a las funciones relacionadas con la animación, ordenar los elementos transparentes de acuerdo con su distancia con la cámara y la entrada de teclado y ratón y de dibujar los elementos en la escena y de trabajar con ellos. Se definen los valores de iluminación, se cargan y transforman los modelos y finalmente se carga el *skybox*.

  Cuando se termina el ciclo, se limpian los buffers.

- *void animation()*

Se encarga de realizar las operaciones correspondientes a las animaciones. Consiste en cinco *if* principales, cada uno perteneciente a una animación. Según cómo se modeló la animación, verifica más condiciones para cada una y realiza los cálculos y modificaciones de variables correspondientes. Estas variables son las correspondientes a la posición, rotación y/o escala de cada objeto relacionado, además de las banderas correspondientes a cada animación.

- *void DoMovement()*

  Se encarga de procesar la entrada continua por teclado, es decir, las teclas que se mantienen presionadas para varias continuamente un valor. En esta función sólo se encuentran las teclas relacionadas al movimiento de la cámara (WASDQE, flechas del teclado y SHIFT izquierdo).

- *void KeyCallback(GLFWwindow* window, int key, int scancode, int action, int mode)*

  Se encarga de procesar la entrada única por teclado, es decir, las teclas que se presionan una vez para realizar una única acción. Aquí se encuentran las teclas relacionadas a la activación y desactivación de las animaciones.

- *void MouseCallback(GLFWwindow* window, double xPos, double yPos)*

  Se encarga de procesar la entrada por ratón. Esta función actualiza la dirección de la cámara según el movimiento recibido del ratón.

Por otro lado, se modificaron los archivos *Modelo.h*, *Mesh.h*, *lighting.frag* y *lighting.vs* para implementar los cambios relacionados a las texturas de transparencia, texturas de normales y procesamiento de semi-transparencias. Estos cambios involucran la carga de las texturas, lectura del valor alfa del color de cada fragmento y modificaciones en los cálculos de la luz en los fragmentos para trabajar en espacio tangencial (necesario porque es el espacio en el que se trabajan las normales de un mapa de normales).