UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO
FACULTAD DE INGENIERÍA

Documentación del Proyecto Final

REALIZADO POR
Gómez Luna, Alejandro
316034946

PROFESOR
ING. CARLOS ALDAIR ROMAN BALBUENA

ASIGNATURA
Laboratorio de Computación Gráfica

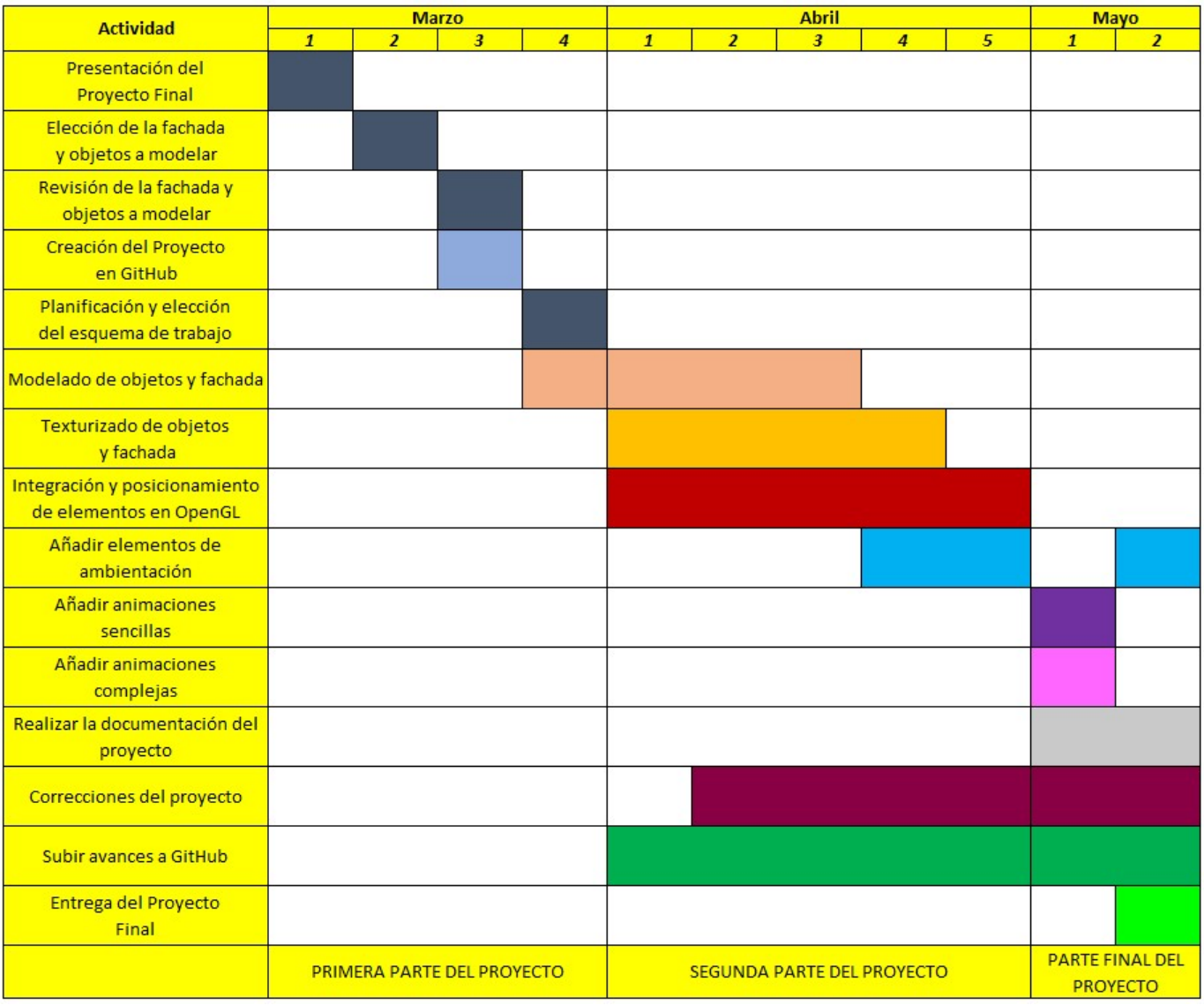
GRUPO DE TEORÍA **No. 04**
GRUPO DE LABORATORIO **No. 04**

FECHA DE ENTREGA
11/Mayo/2022

Objetivos

- El alumno deberá aplicar y demostrar los conocimientos adquiridos durante todo el curso.
- El alumno utilizará herramientas como Maya y GIMP para lograr el modelado y texturizado de sus objetos.
- El alumno aprenderá las bases de OpenGL, así como la utilidad y aplicación de GLFW y GLEW.
- El alumno analizará y comprenderá las partes más importantes del pipeline gráfico.

Diagrama de Gantt



La metodología utilizada para el desarrollo de este proyecto fue basada en Scrum, en donde se fijaron tres sprints, siendo el primero de una duración de 4 semanas, el segundo de una duración de 5 semanas, y el último de 2 semanas. Para el primer sprint se fijó como meta el establecer las bases y tiempos para poder realizar el proyecto, además de tener los fundamentos teóricos para satisfacer los requerimientos iniciales del proyecto.

Posteriormente, una vez que se había completado este sprint, se pasó a la segunda etapa del proyecto, en el cual se tuvo como objetivo modelar todos los objetos propuestos en conjunto con la fachada, además de texturizarlos correctamente, para finalmente integrarlos en el ambiente de OpenGL, tomando como base el proyecto proporcionado por el profesor. En este sprint también se buscó crear las componentes de ambientación adecuadas para el ambiente.
Finalmente se tuvo el último sprint, la última parte del proyecto, en donde la meta fue realizar las animaciones solicitadas, añadir la última componente de ambientación referente al skybox, generar la documentación del proyecto, y realizar correcciones generales.
Durante cada Sprint se tenía una fase de desarrollo y otra fase de prueba, en donde cada cambio que se hacía al proyecto se iba comprobando al ejecutar el código en Visual Studio, por lo que se tenía una retroalimentación constante, permitiendo modificar el código de manera flexible ante cualquier error que se pudiera llegara a presentar.

Alcance y Limitaciones del Proyecto

Se realizó la recreación lo más parecido posible de la fachada, sótano y siete objetos del hogar de Eren Jaeger de la serie “Attack on Titan", siguiendo un estilo de animación japonés, en donde para este caso se trata de asemejar a la realidad, por lo que se usaron texturas que se verían en un ambiente real.
Los objetos se modelaron y texturizaron haciendo uso del software de Maya LT 2020, además de que las imágenes ocupadas para las texturas se adaptaron haciendo uso de GIMP 2.10.30.
Todos los objetos se importaron en OpenGL 3.2, en donde, haciendo uso de transformaciones básicas como traslaciones y rotaciones, se posicionaron correctamente para lograr la recreación de la fachada y el sótano.
En conjunto con OpenGL se utilizaron las bibliotecas de GLEW y GLFW, con lo cual se pudo simplificar el trabajo de detectar entradas y el manejo de extensiones de OpenGL.
Todo el proyecto se trabajó haciendo uso del IDE Microsoft Visual Studio 2022. Para poder descargar Visual Studio 2022 se necesita acceder al siguiente enlace `https://visualstudio.microsoft.com/es/vs/` en donde se debe bajar Visual Studio Community 2022. Almacenamos el archivo en algún lugar de nuestra computadora, desde donde se realizará todo el proceso de instalación. Ya que está instalado entonces lo abrimos y seleccionamos la opción de Crear Nuevo Proyecto, luego Proyecto en Blanco, ponemos el nombre de nuestro proyecto y su localización, con lo cual finalmente creamos el proyecto.
Aunado a los objetos, se añadieron elementos de ambientación, entre los cuales se encuentran:

- Dos luces puntuales, las cuales proporcionaban la mayor fuente de iluminación en el cuarto principal de la fachada y en el sótano. Aunada a la iluminación se hizo uso de una luz direccional general que permitiera iluminar a todos los objetos en el ambiente.
- Un Skybox para representar los alrededores del hogar.
- El uso de transparencia en las ventanas de la casa y la lámpara colgante del techo con la finalidad de lograr el efecto de que eran cristales.

El proyecto no incluye:

- El uso de colisiones para delimitar físicamente los cuartos, objetos, etc.
- El uso de técnicas avanzadas de computación gráfica como Ray Tracing o Aliasing.
- Texturas complejas, pues las texturas aplicadas solamente se obtienen a partir de una imagen, por lo que no tienen ningún mapeo asociado.
- La interactividad con un personaje que se pueda mover a través del ambiente virtual.
- Elementos más inmersivos de ambientación, como pueden ser sonidos o físicas de los objetos.

Desarrollo

Este proyecto se basó en la siguiente lista de requerimientos funcionales

- Modelar y texturizar correctamente como mínimo 7 objetos mostrados en una imagen de referencia, la cual debe de ser avalada por el profesor.
- Modelar y texturizar correctamente cualquier otro objeto como ventanas y puertas que aparezcan mostrados en la imagen de referencia, la cual debe de ser avalada por el profesor.
- Modelar y texturizar correctamente una fachada que también se mostrara en una imagen de referencia, la cual debe de ser avalada por el profesor.
- Integrar todos los objetos y fachada en OpenGL, en donde se utilizaran transformaciones básicas para poder lograr la recreación en tercera dimensión del espacio escogido. Toda la integración se hará con base en un proyecto y código proporcionado por el profesor, el cual se ejecutará haciendo uso de la herramienta Visual Studio.
- Los objetos modelados deberán de estar en un archivo con extensión .obj. Estos pueden ser modelados totalmente o parcialmente, en donde se especifique de donde se obtuvieron para posteriormente ser adaptados.
- Haciendo uso de los objetos modelados e integrados en OpenGL, crear 5 animaciones, en donde 3 de ellas sean de tipo simple y 2 dos de ellas de tipo compleja. Las animaciones sencillas solamente hacen uso de traslaciones simples, como lo son movimientos lineales o rotaciones sobre un mismo eje. Las animaciones complejas pueden ser realizadas haciendo uso de técnicas de animación como KeyFrames o con movimientos más complejos, como tiros parabólicos, caída libre, trayectorias compuestas, trayectorias basadas en funciones trigonométricas, entre otros. Todas las animaciones deben de tener sentido en el contexto sobre el cual se están desarrollando, lo que significa que deben de tener sentido respecto al ambiente en el que se encuentran.
- Realizar la documentación técnica del proyecto, además de un manual de usuario en donde el usuario pueda saber cómo interactuar con el ambiente.
- Utilizar Git y GitHub para cargar todos los avances que se realicen durante el proyecto.

Respecto a los requerimientos no funcionales, los cuales proporcionaron mayor calidad al proyecto, se tienen

- Evitar el uso de escalamientos en OpenGL, pues esta es una transformación básica que consume muchos recursos computacionales en tiempo de ejecución.
- El uso de componentes de ambientación que logren una mejor recreación del ambiente. Dentro de estas componentes tenemos a la iluminación, Skybox, Bumping maps, transparencias, etc.
  Dentro de la iluminación se tiene el uso de luces direccionales, puntuales, y reflectoras.
- Lograr que el usuario pueda tener una interactividad fluida con el ambiente recreado, permitiéndole el manejo intuitivo de la cámara, así como el accionar sencillo de las animaciones.

Finalmente, respecto al código, se tiene el siguiente glosario de variables y de funciones

Glosario de variables

1. `WIDTH, HEIGHT, SCREEN_WIDTH Y SCREEN_HEIGHT`: Definen el ancho y alto de la ventana con la cual estamos trabajando.

2. `camera, lastX, lastY, firstMouse`: Se usan para la manipulación de la cámara con el mouse.

3. `keys`: Este arreglo de variables booleanas indica cuál tecla ha sido presionada.

4. `deltaTime y lastFrame`: Conocer los tiempos entre el frame actual y el último.

5. `anim_PP, regreso_PP y rot_PP`: Estas tres variables se ocupan para la animación sencilla de la puerta principal. La variable anim_PP es la encargada de conocer si la animación se encuentra activa o no, las otras dos variables se explican a detalle en la función de DoMovement().

6. `anim_PC, regreso_PC y rot_PC`: Estas tres variables se ocupan para la animación sencilla de la puerta del pasillo. La variable anim_PC es la encargada de conocer si la animación se encuentra activa o no, las otras dos variables se explican a detalle en la función de DoMovement().

7. `anim_PS, regreso_PS y rot_PS`: Estas tres variables se ocupan para la animación sencilla de la puerta del sótano. La variable anim_PS es la encargada de conocer si la animación se encuentra activa o no, las otras dos variables se explican a detalle en la función de DoMovement().

8. `anim_C, regreso_C, tras_caj1, tras_caj2, tras_caj3 y tras_caj4`: Estas seis variables se ocupan para la animación sencilla de los cajones del escritorio en el sótano. La variable anim_C es la encargada de conocer si la animación se encuentra activa o no, las otras variables se explican a detalle en la función de DoMovement().

9. `anim_L, espiral_L, regreso_C, posX, posY, posY2, posZ, rotX, rotX2, rotY, rotZ, theta`: Estas variables se ocupan para la animación compleja de la lámpara colgante en el techo del sótano. La variable anim_L es la encargada de conocer si la animación se encuentra activa o no, las otras variables se explican a detalle en la función de DoMovement().

10. `anim_T, regreso_T, puntoX, puntoY, giroX, giroZ, desA, t`: Estas variables se ocupan para la animación compleja de la copa que se encuentra posicionada sobre el escritorio en el sótano. La variable anim_T es la encargada de conocer si la animación se encuentra activa o no, las otras variables se explican a detalle en la función de DoMovement().

11. `lightPos`: Vector de 3 posiciones para definir de dónde se origina la luz respecto a los objetos de iluminación que se crean.

12. `pointLightPositions`: Arreglo de vectores de 3 posiciones que define las posiciones absolutas para colocar las dos luces puntuales.

13. `Light1 y Light2`: Definen el color de la luz para las anteriores luces puntuales creadas, en donde la primera tendrá un color anaranjado, mientras que la segunda será totalmente una luz blanca.

14. `vertices`: Arreglo de posiciones y normales, con las cuales se puede definir la figura geométrica de un cubo, lo cual se ocupará para dibujar las luces puntuales.
    De cada fila, los tres primeros elementos corresponden a coordenadas en donde se posicionarán los vértices necesarios para dibujar una cara del cubo, mientras que los tres siguientes elementos definen la normal de dicha cara del cubo.

15. `skyboxVertices`: Define las posiciones de los vértices que conformarán al cubo que se utilizará como el skybox.

Glosario de funciones

1. `main()`: Esta es la función principal del proyecto.
   Primero se empieza por configurar lo necesario para poder trabajar, por lo que se debe de inicializar la ventana y verificar que se cree correctamente. Esta ventana tendrá dimensiones de 800 x 600 píxeles.
   Ya con la ventana cargada la configuramos para que se asocien las funciones de llamado de tecla y de mouse, además de deshabilitar el cursor. Finalmente definimos nuestro viewport, el cual corresponde a las dimensiones de la ventana.
   Con la ventana cargada y configurada correctamente, se procede a localizar y compilar los shaders, que en este caso son el lightning, lamp y skybox, los cuales vienen en pares, pues se componen de un vertex y de un fragment shader, donde primero se recibe la información en el vertex para procesarla y posteriormente mandarla al fragment shader. Después se cargan y compilan los modelos definidos, especificando la ruta relativa de donde se encuentran los archivos .obj. Compilados y ubicados los archivos necesarios se hace uso del VBO y VAO, que son apuntadores para poder cargar los vértices definidos con anterioridad para poder dibujar un cubo y el sky box, en donde se debe especificar la cantidad de elementos que corresponden a posiciones. En el caso del sky box también se requiere especificar la dirección relativa de las caras que se estarán cargando en el sky box.
   Finalmente se define el tipo de proyección que se utilizará, que en este caso es en perspectiva, con lo cual se guarda la proporción de los objetos en tercera dimensión, pues las líneas de proyección convergen en puntos de fuga.
   Una vez que se tiene toda la configuración inicial se tiene un ciclo que se ejecutará siempre que no se cierre la ventana. En este ciclo se define todo lo referente a luces y carga de modelos. Las luces y modelos se verán afectados por el lightning shader, con el cual se logra el efecto de iluminación deseado, que está siendo provocado por una luz direccional (ambiental que es por igual en todos los objetos de la escena) y dos luces puntuales, con sus respectivas distancias de iluminación.
   Con toda la iluminación puesta en escena, se procede a cargar todos los modelos, a los cuales primero se les aplican transformaciones básicas como traslaciones o rotaciones para colocarlos correctamente en la escena según la imagen de referencia. En algunos objetos las traslaciones o rotaciones se realizan conforme a una variable, lo cual es porque realizarán una animación posteriormente.
   Después de todos los modelos que se han cargado entonces se usa el lamp shader para dibujar las luces puntuales que se utilizaron y el sky box shader para crear el sky box.
   Finalmente se limpian todos los recursos almacenados en memoria y termina la función.

2. `DoMovement()`: Esta es la función en donde se realizan todas las animaciones y controles de cámara.
   Además del control de cámara, que se realiza con las teclas W, S, A y D, se tienen todas las animaciones.

   - La animación sencilla de la puerta principal se acciona siempre que la variable anim_PP está activa.
     Esta animación consta de tres partes. La primera de ellas es cuando la puerta rota de 0 a 90 grados respecto al eje Y. Una vez que completa esta trayectoria entonces se pasa a la segunda parte, que es el regreso, en donde la puerta va desde 90 grados hasta -90 grados en el mismo eje Y. Ya que se completó se tiene la última parte, que es cuando regresa a su posición inicial.
     Todas estas rotaciones se controlan con la variable rot_PP, en donde se va almacenando la rotación de la puerta. Asimismo, la variable regreso_PP se usa para saber en qué parte de la animación se encuentra la puerta.
   - La animación sencilla de la puerta que da hacia el pasillo se acciona siempre que la variable anim_PC está activa.
     Esta animación consta de dos partes. La primera de ellas es cuando la puerta rota de 0 a -110 grados respecto al eje Z. Llegado este límite vuelve a su posición inicial.

Todas estas rotaciones se controlan con la variable rot_PC, en donde se va almacenando la rotación de la puerta. Asimismo, la variable regreso_PC se usa para saber en qué parte de la animación se encuentra la puerta.

- La animación sencilla de la puerta del sótano se acciona siempre que la variable anim_PS está activa. Esta animación es igual a la de la puerta principal de la fachada.
- La animación sencilla de los cajones del escritorio en el sótano se acciona siempre que la variable anim_C está activa.
  Los cajones se van a ir abriendo uno por uno, empezando por el cuarto de abajo hasta llegar al primero de arriba. Una vez que se han abierto todos, entonces se van a ir cerrando desde el primero de arriba hasta el cuarto de abajo.
  Los traslados de los cajones se controlan con las variables tras_caj 1 a 4, donde cada variable lleva el traslado en Z de cada uno de los cajones.
  Asimismo, la variable regreso_C se usa para saber en qué parte de la animación se encuentran los cajones.
- La animación compleja de la lámpara colgante en el techo del sótano se acciona siempre que la variable anim_L esá activa.
  La lámpara va a describir tres trayectorias. La primera de ellas es una inclinación de la lámpara respecto a su eje Z, en donde se posicionará hasta el punto en donde comienza la segunda trayectoria, que es una espiral en donde estará moviéndose la lámpara, para finalmente volverse a inclinar y volver a su posición inicial.
  En la primera trayectoria se tiene todo en función de la variable posZ, en donde se irá almacenando la posición en el eje Z de la lámpara de techo. Esta posición va desde 2 a 6 unidades, con incrementos de 0.05, teniendo un total de 80 incrementos. Con estos incrementos en mente tenemos el incremento de las otras dos variables, que son rotX y posY, en donde se busca que rotX llegue hasta 40 y posY hasta 1, por lo que $\frac{rotX}{80} = \frac{40}{80} = 0.5$, y $\frac{posY}{80} = \frac{1}{80} = 0.0125$. Con rotX rotamos la lámpara respecto al eje X para dar este efecto de inclinación. La variable posY sirve para modificar la posición de la lámpara en Y, pues al momento de aplicar la rotación esta sufre un desplazamiento en Y de aproximadamente una unidad, por lo que se debe de anular este movimiento para mejorar la animación.
  Para la segunda trayectoria se tiene una trayectoria en espiral de Arquímedes, la cual se describe con la siguiente ecuación en coordenadas polares

$$r = a + b\theta$$

En donde a vale 0 y b vale 0.1, por lo que tenemos $r = 0.1\theta$. Como se está trabajando con coordenadas rectangulares entonces se necesitan las ecuaciones de transformación

$$x = rcos(\theta)$$
$$y = rsen(\theta)$$

Sustituyendo todo en términos de θ

$$x = 0.1\theta cos(\theta)$$
$$y = 0.1\theta sen(\theta)$$

Para este caso el eje X corresponde al eje Z en OpenGL y el eje Y corresponde al eje X en OpenGL. En el eje Z de OpenGL se tiene un desplazamiento de 0.4 unidades, mientras que en el eje X de OpenGL se tiene un desplazamiento de 37.5 unidades.

Con estos cálculos realizados se puede ir trasladando la lámpara correspondiente para describir la trayectoria en elipse, pero a la par se necesita ir rotando respecto a un eje y, el cual corresponde a un pivote que está fuera de la lámpara y que no está inclinado, por lo que se necesitó crear otra lámpara inclinada con estas características necesarias, y la cual solamente es visible cuando se tiene esta parte de la animación.
La variable theta es la que corresponde al parámetro del que depende la posición de Z y X de la lámpara, por lo que este irá variando de -15.7 a 0.0, con un incremento de 0.02. A la par que se hace este cambio se requiere ir rotando la lámpara respecto al eje Y, lo cual se hace con la variable rotY. Esta necesita rotar 360 grados cada que se complete un ciclo de la espiral, lo cual sucede cada intervalo de 6.3 en theta, por lo que, si se tiene un incremento de 0.02 $\frac{rotY}{\frac{6.3}{0.02}} = \frac{360}{315} = 1.1428$
Finalmente se tiene la última trayectoria, en la cual la lámpara inclinada va a volver a la posición inicial. En este caso la rotación respecto al eje X se almacena en la variable rotX2, la posición en Z en posZ y la posición en Y en posY2. Al igual que para el primer recorrido, se van a tener 80 incrementos, pues rotX2 se irá realizando cada 0.5 unidades hasta llegar a 40.0, que es donde termina la animacion y todas las variables regresan a su estado inicial. En este caso posZ llega hasta 6 porque la lámpara inclinada tiene su pivote 4 unidades desplazado en Z. También se realiza la corrección de posición en el eje Y.
Aunado a todo este movimiento se tiene la luz puntual, la cual en el caso de la espiral solamente sigue un movimiento circular, pero en las otras dos trayectorias sí sigue la misma posición en Z que la lámpara.

- La animación compleja de la copa sobre el escritorio en el sótano se acciona siempre que la variable anim_T esá activa.
  En esta animación se tienen tres trayectorias. La primera de ellas solamente es un desplazamiento lineal de la copa en el eje X hacia la orilla del escritorio.
  La segunda trayectoria es un tiro parabólico definido por las siguientes ecuaciones en X y Y

$$x = x_0 + v * t$$
$$y = H - \frac{1}{2} * g * t^2$$

  Para este caso se utilizó una velocidad horizontal de 3.0 y una gravedad de 0.96, para tener un mayor desplazamiento en X y un desplazamiento más ligero en Y. La posición inicial en X es 2.0 y la altura es de 4.8. Con esto solamente se sustituyen los valores y se obtienen los desplazamientos correspondientes, los cuales se almacenan en puntoX y puntoY, correspondientes a los ejes X y Y, respectivamente. Cuando se llega a 0.4 en Y entonces la base de la copa ya toca el suelo. A la par se rota la copa respecto al eje Z, lo cual debe ser hasta 90 grados, para simular que está cayendo correctamente. Haciendo los despejes correspondientes se sabe que en $t = 3.027$ se llega al piso, y considerando un incremento de t de 0.05 se tienen $\frac{3.027}{0.05} = 60.54$ intervalos, con lo que el incremento en la rotación debe de ser de $\frac{90}{60.54} = 1.487$.
  En la última parte de la animación se necesita rotar la copa respecto a su eje X, para simular que rueda un poco por el piso debido a la energía cinética acumulada, sufriendo una ligera desaceleración. Esto se realiza hasta que se llega a 180 grados, y es cuando se vuelve a reinicializar toda la animación. La última trayectoria solamente es para regresar la copa a su estado inicial y volver a inicializar todas las variables ocupadas.
- `KeyCallback()`: Esta función se llama siempre que se presiona una tecla.
  Si la tecla corresponde a Esc entonces se cierra la ventana.
  Con la tecla C se cambia el estado de la variable anim_PP, permitiendo activar o desactivar la animación de la puerta principal.
  Con la tecla V se cambia el estado de la variable anim_PC, permitiendo activar o desactivar la animación de la puerta del pasillo.

Con la tecla B se cambia el estado de la variable anim_PS, permitiendo activar o desactivar la animación de la puerta del sótano.
Con la tecla N se cambia el estado de la variable anim_C, permitiendo activar o desactivar la animación de los cajones del escritorio en el sótano.
Con la tecla M se cambia el estado de la variable anim_L, permitiendo activar o desactivar la animación de la lámpara.
Con la tecla L se cambia el estado de la variable anim_T, permitiendo activar o desactivar la animación de la copa.

- `MouseCallBack()`: Esta función se llama siempre que se tiene movimiento del cursor del mouse, con lo cual se irá modificando la orientación de la cámara.

Se muestran los resultados obtenidos, en donde se comparan con las respectivas imágenes de referencia

Figura 1: Imagen de referencia para la fachada

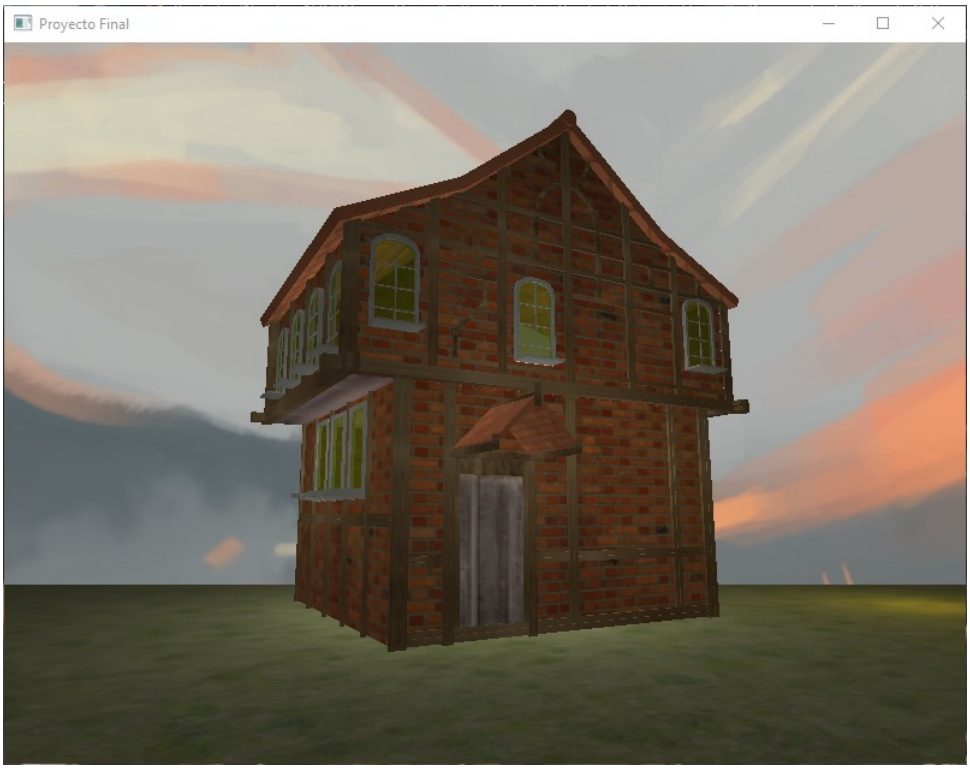


Figura 2: Fachada recreada e integrada en OpenGL

Figura 3: Imagen de referencia para el pasillo

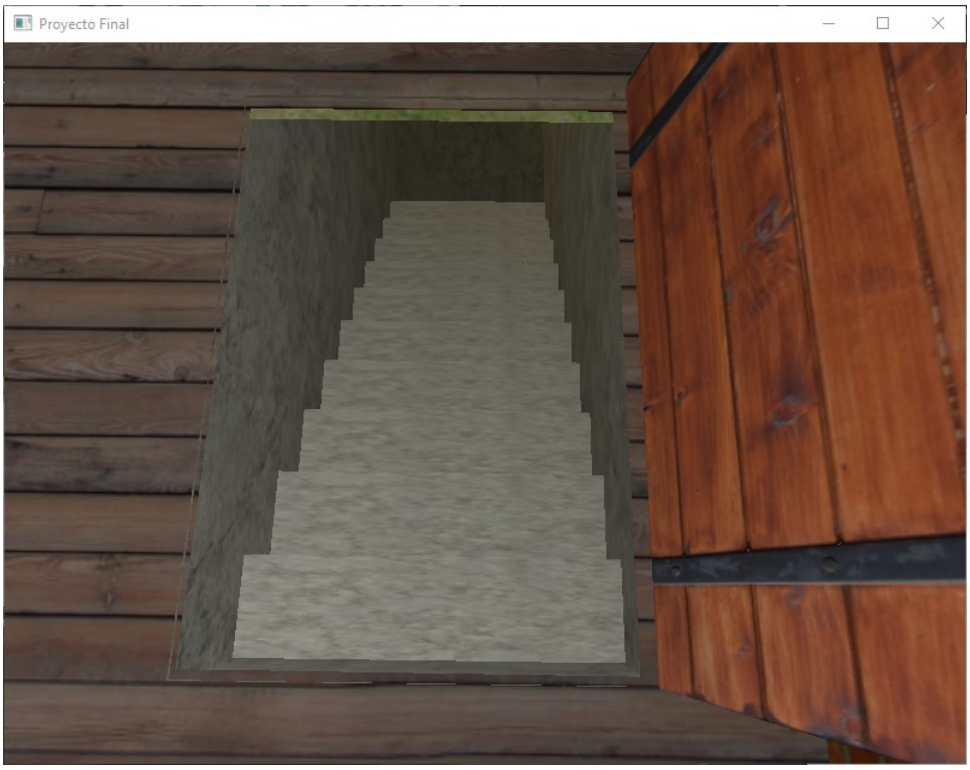


Figura 4: Pasillo recreado e integrada en OpenGL

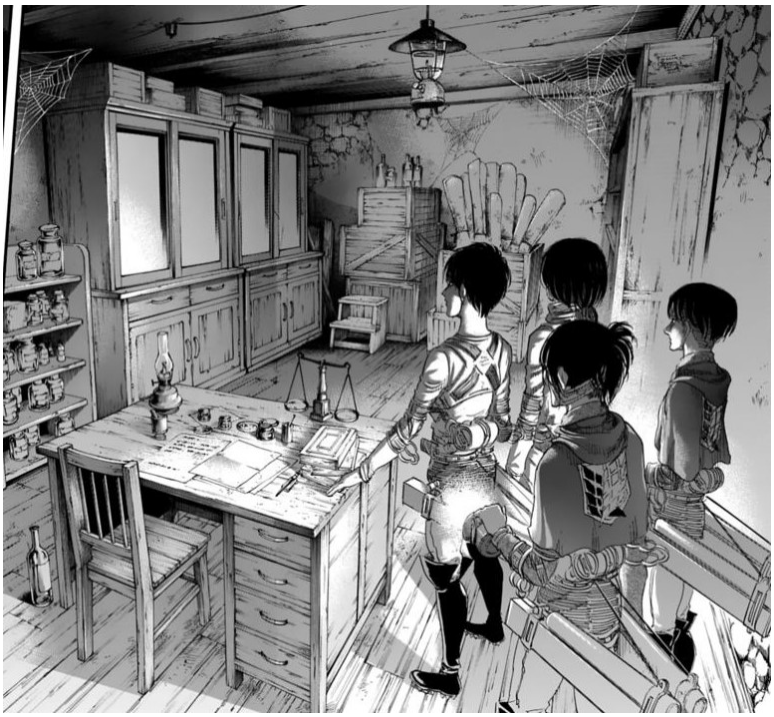


Figura 5: Imagen de referencia para el sótano

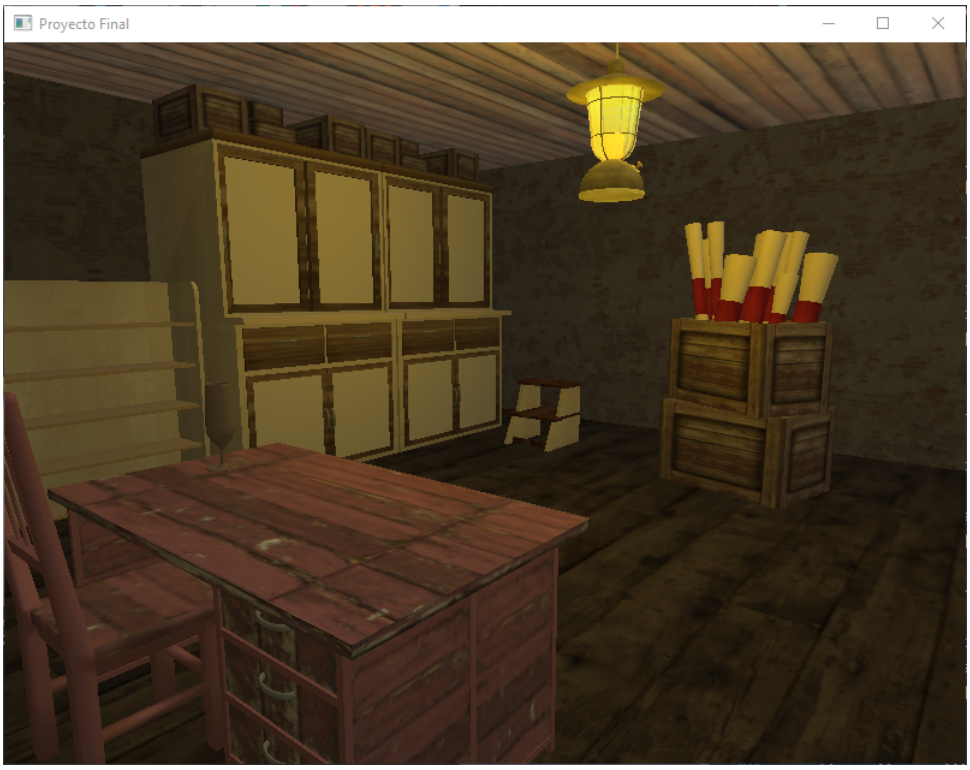


Figura 6: Sótano recreado e integrada en OpenGL

Conclusiones

Para este proyecto se cumplieron todos los objetivos propuestos, pues todo lo aprendido durante el curso se aplicó para la recreación el ambiente virtual propuesto, el cual si bien no es una representación totalmente fidedigna, sí muestra un buen grado de parentesco respecto a la imagen de referencia. A lo largo de este proyecto se fue aplicando lo que se aprendía de manera teórica, como lo fue en un inicio el aprender a modelar objetos desde OpenGL, únicamente definiendo toda la geometría a partir de vértices desde los cuales se iban dibujando líneas, ciclos, triángulos, etc. Si bien esta era una forma de modelar directamente los objetos en OpenGL, se volvía una tarea sumamente compleja y tediosa, por lo que se usó el software de modelado Maya, el cual también hace uso de transformaciones básicas pero le permite al usuario crear todos sus modelos de una manera mucho más intuitiva al contar con una interfaz gráfica. Asimismo, el proceso de texturizado de los modelos se realizó con ayuda de GIMP, para poder redimensionar imágenes que posteriormente se ocuparían como texturas, así como el poder crear colores puramente sólidos. Algo importante a destacar es que siempre que se texturizaba se necesitaba orientar correctamente la textura en Maya, ocupando la proyección en el eje correspondiente.
Después de que se tuvieran todos los objetos modelados y texturizados, estos se trajeron a OpenGL. Si bien no fue una tarea complicada, requería de entender el código para saber cómo colocar los objetos, trasladarlos, rotarlos, añadirles iluminación, transparencia, etc. Todo esto ya se había revisado previamente en clase, lo que facilitó bastante todo este proceso.
Finalmente se realizaron las animaciones sencillas y complejas, en donde las complejas fueron las que más requirieron de tiempo y entendimiento de ecuaciones matemáticas, pues los movimientos se basaban en funciones trigonométricas como el seno y coseno en el caso de la espiral, o basados en ecuaciones físicas como la del tiro parabólico. A pesar de las dificultades presentadas durante la realización de estas animaciones, se lograron animar los objetos correctamente.

Sin la realización de este proyecto final todo el conocimiento teórico y práctico no habría sido suficiente para lograr un mejor entendimiento de la computación gráfica, pues es hasta que todo lo aprendido se aplica cuando realmente se logra un aprendizaje más significativo. Si bien aún existen muchos otros temas de computación gráfica más avanzadas, con niveles de complejidad mayores, el poder entender las bases de modelado, texturizado e iluminación permite tener un amplio panorama general de el funcionamiento de la computación gráfica.
Con lo aprendido durante el curso y este proyecto se tiene una mejor noción de las aplicaciones de la computación gráfica, la cual en la actualidad ya está sumamente ligada a cualquier avance tecnológico, pues permite a los usuarios interactuar con un sistema digital de una manera mucho más sencilla y fluida, además de abrir paso a la creación de ambientes virtuales cada vez más realistas y detallados.

Referencias

Modelos en 3D

- La silla del sótano, adapatada de `https://www.turbosquid.com/3d-models/3d-set-16-medieval-wooden-model-1422733`, TurboSquid, realizado por leon017.
- La lámpara del sótano, adaptada de `https://www.turbosquid.com/3d-models/obj-ceiling-lamp/532153`, TurboSquid, realizado por odio.
- La copa encima del escritorio, obtenida de `https://www.turbosquid.com/3d-models/3d-glass-champagne-model-1465324`, TurboSquid, realizado por Lazaro Bustos.

Texturas

- Texturas de madera para el escritorio, obtenidas de `https://www.turbosquid.com/FullPreview/Index.cfm/ID/922096` y `https://www.turbosquid.com/FullPreview/Index.cfm/ID/922092`, TurboSquid, realizados por James Brody.
- Textura metálica usada en todas las manijas metálicas de los muebles, obtenidas de `https://www.turbosquid.com/FullPreview/Index.cfm/ID/342167`, TurboSquid, realizado por flumpe.
- Textura de madera clara, obtenida de `https://www.turbosquid.com/FullPreview/Index.cfm/ID/141341`,TurboSquid, realizado por Ace.
- Textura de tablones para las repisas, obtenida de `https://www.turbosquid.com/FullPreview/Index.cfm/ID/509041`,TurboSquid, realizado por Ghostman56.
- Textura de madera obscura para los muebles y tablones de la fachada, obtenida de `https://www.turbosquid.com/FullPreview/Index.cfm/ID/1143570`,TurboSquid, realizado por orange 3d.
- Textura de caja de madera, obtenida de `https://www.turbosquid.com/FullPreview/Index.cfm/ID/350667`, TurboSquid, realizado por zaphad1.
- Textura metálica obscura, obtenida de `https://www.turbosquid.com/FullPreview/Index.cfm/ID/162363`, TurboSquid, realizado por alexkemmier.
- Textura roja para enrollar los pergaminos, obtenida de `https://www.texturepalace.com/red-color-textile-texture-free-download/`, TexturePalace, realizado por Sab.

- Textura de concreto, obtenida de `https://www.turbosquid.com/FullPreview/Index.cfm/ID/595616`, TurboSquid, realizado por Game Design Planet.
- Textura de tablones de madera para techo y suelo de los cuartos, obtenida de `https://www.turbosquid.com/FullPreview/Index.cfm/ID/555186`, TurboSquid, realizado por ringhino.
- Textura de tablones de madera para la puerta del sótano, obtenida de `https://www.turbosquid.com/FullPreview/Index.cfm/ID/232531`, TurboSquid, realizado por bupaje.
- Textura de madera para la puerta del pasillo, obtenida de `https://www.textures.com/download/WindowsShutters0187/54583`, Textures.
- Textura de madera para el suelo del sótano, obtenida de `https://www.textures.com/download/WoodPlanksOld0244/121294`, Textures.
- Textura de tejas para el techo de la fachada, obtenida de `https://www.vecteezy.com/vector-art/171678-roof-tile-background`, Vecteezy, realizado por foxarthappy.
- Textura de ladrillos para la fachada, obtenida de `https://www.turbosquid.com/FullPreview/Index.cfm/ID/243350`, TurboSquid, realizado por InfinityStudio.
- Textura de madera para la puerta principal de la fachada, obtenida de `https://www.textures.com/download/WoodPlanksBare0354/110860`, Textures.
- Textura de pasto alrededor de la fachada, obtenida de `https://www.textures.com/download/Grass0169/51506`, Textures.
- Textura de ladrillos para el sótano, obtenida de `https://www.textures.com/download/BrickSmallPlaster0119/134802`, Textures.
- Textura de Sky Box, obtenida de `https://www.cleanpng.com/png-heat-sky-plc-skybox-3444567/download-png.html`, Cleanpng, realizada por Alishu.