UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO
Facultad de Ingeniería

Materia:
Laboratorio de computación gráfica e interacción humano-computadora

Integrante:
Fernández Sánchez Lucía Victoria

Profesor:
Ing. Carlos Aldair Román Balbuena

Grupo:
09

Semestre:
2021-1

Fecha de entrega:
Martes, 19 de enero de 2020

Manual técnico:

Proyecto Final

Objetivos

- El alumno deberá aplicar y demostrar los conocimientos adquiridos durante todo el curso.
- El alumno deberá realizar un proyecto en donde se vean los diferentes conceptos aprendidos en el curso de laboratorio y teoría (si se requiere).
- El alumno deberá crear un espacio (Pinturas de Vincent Van Gogh) en un editor de preferencia (Blender) y en OpenGL con el fin de representar un espacio (cuarto) con mínimo 7 elementos.
- El alumno deberá realizar animaciones utilizando mínimo dos conceptos utilizados en las prácticas.

Alcance del proyecto

Recrear mediante software especializado un ambiente, mínimo un cuarto, utilizando diferentes figuras geométricas; utilizando materiales o texturas para darle más parecido a las imágenes de referencia previamente elegidas y aceptadas por el profesor.

Representar mínimo 7 figuras dentro del cuarto. En este caso, fueron una cama, sillas, mesa, plato, perchero, un cuadro y un florero.

Adjuntar y editar un archivo proyecto de C++ en el entorno de desarrollo de MS Visual Studio utilizando OpenGL. Dicho código deberá de ser basado en los anteriores dado por el profesor de laboratorio, con modificaciones del alumno dado que éste se debe de acomodar a las necesidades que el proyecto requiera.

Crear desde cero una figura primitiva en Visual Studio, esto es, aplicando el conocimiento del curso, ser capaz de aplicarlo y poder visualizar de manera correcta una figura, en este caso, fue un cuadro de la noche estrellada.

Implementar todas las figuras creadas en el software, en este caso, se utilizó Blender. Realizando los cambios necesarios con el fin de tener cada una de ellas en la posición y tamaño correctos.

Codificar las diferentes animaciones en el proyecto de Visual Studio. La cantidad mínima de animaciones son cinco, donde tres son sencillas y 2 son complejas. Esto con el objetivo de que el usuario con presionar una tecla, se puedan visualizar de manera correcta.

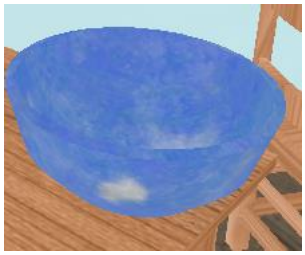
Modelos

La mayoría fueron creados por su servidora. Sin embargo, algunos otros no, por lo que se listarán y se colocarán sus respectivas fuentes:

- ***Florero***

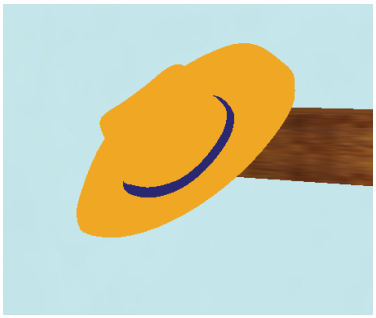
(Oropeza, 2020)

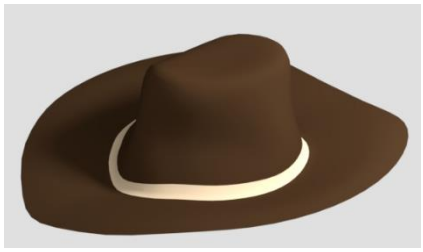
- ***Plato***



(DIBUHO 3D, 2018)

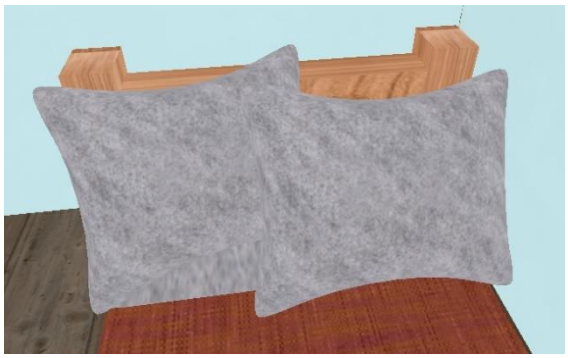
- ***Sombrero***





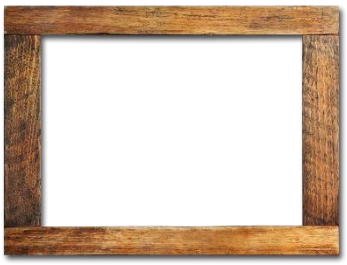
(Low Poly Cartoon Cowboy Hat Free, 2019)

- ***Almohada***



(DIBUHO 3D, 2018)

- ***Cuadro***



(MARCO DE MADERA GIF, s. f.)

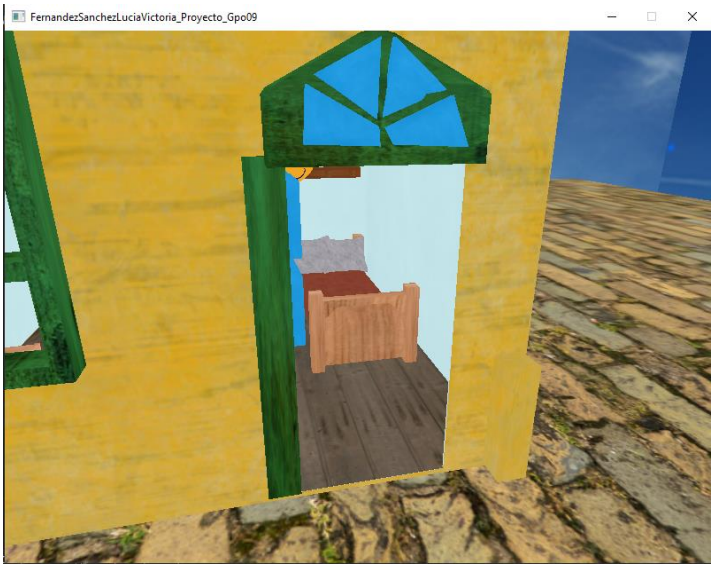
(Van Gogh, s. f.)

Animaciones

Las animaciones se dividen en dos, las sencillas y las complejas. Para los dos tipos de animaciones, se tuvieron que requerir **variables auxiliares** para que solamente el usuario pudiera ver la animación correspondiente.

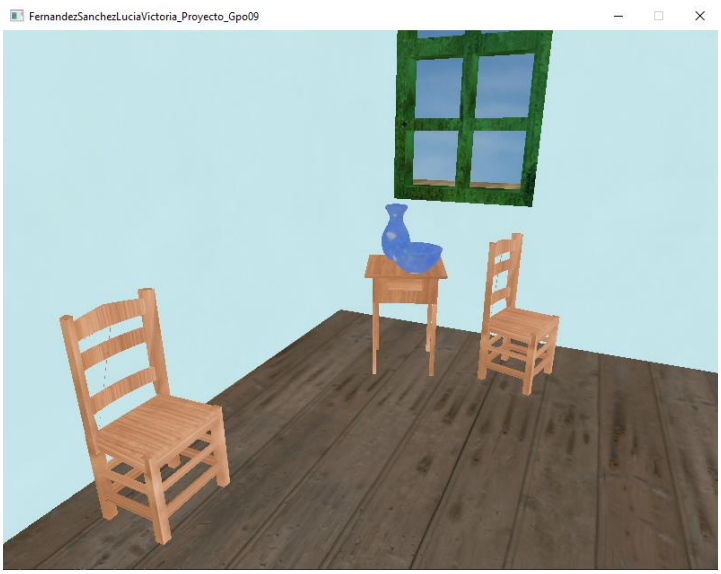
- ***Sencillas***
  El abrir y cerrar de una puerta; el movimiento de una de dos almohadas; cambio de posición de las sillas con el fin de que coincidan con la mesa.

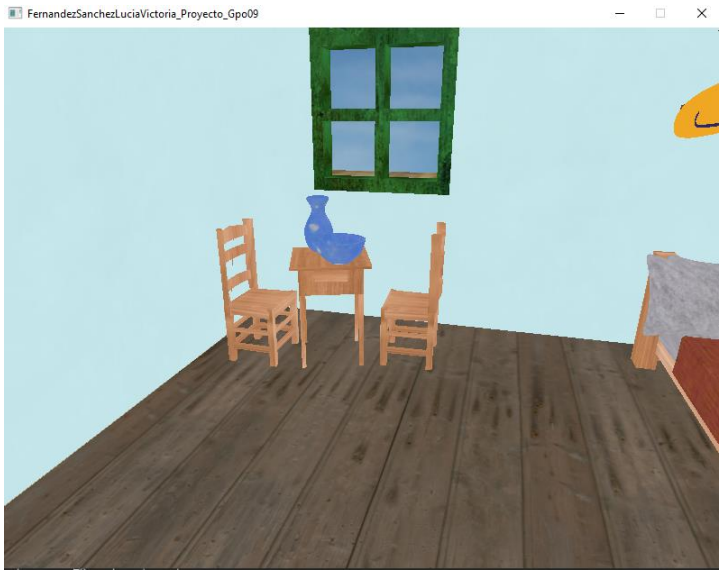
1. Abrir y cerrar la puerta



2. Movimiento de almohada

3. Cambio de posición de sillas





- ***Complejas***

  Caída del sombrero amarillo (se utilizaron dos teclas, en donde la segunda devuelve la posición del sombrero); Movimiento del florero al plato (solo una tecla, pues solito se devuelve a la posición inicial).

  1. Sombrero

2. Florero (No se pudo realizar la evidencia, pero si funciona)

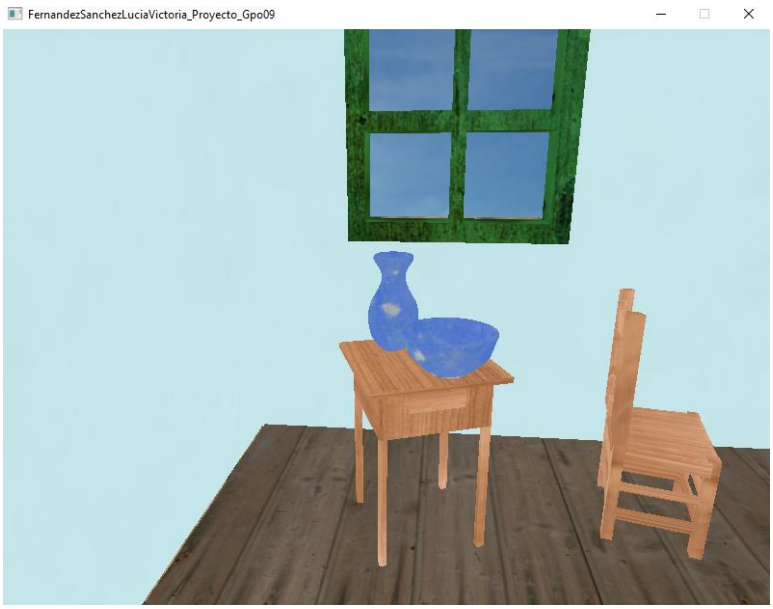
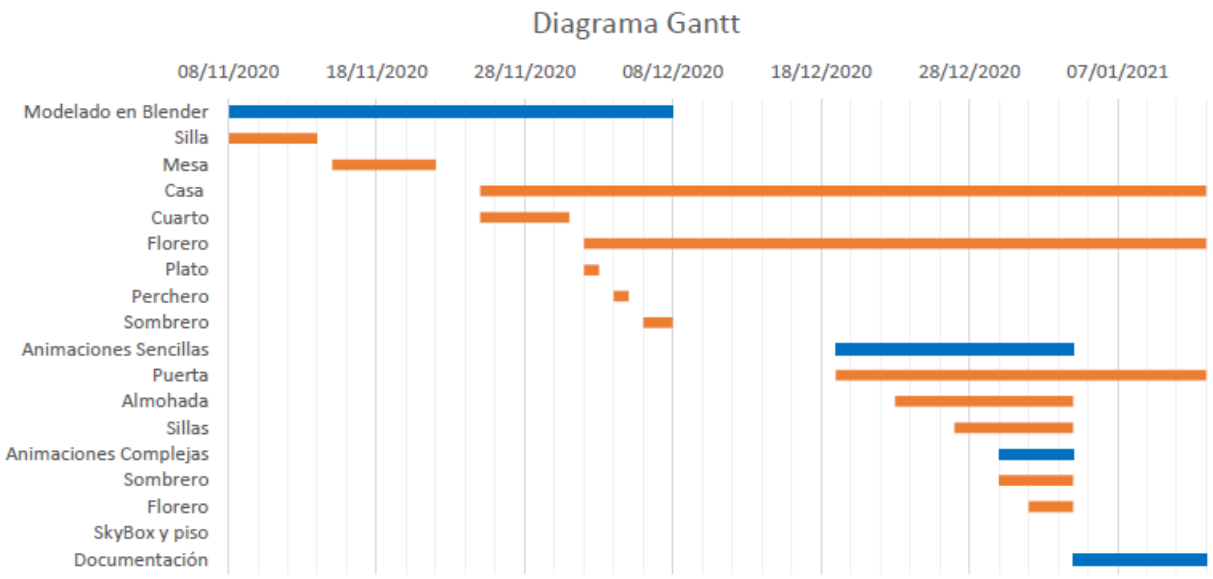


Diagrama de Gantt



Existen algunos elementos que tienen mayor duración a pesar de no realizarse en toda esa cantidad de días, sin embargo, se menciona hasta el último día en el que hubo modificación.

Documentación

Funciones principales del código

- ***void saveFrame(void)***

  Función que se encarga de guardar las posiciones, por eso que el arreglo de los keyframes se encuentra en 0 y en 1, en 0 únicamente se guardan las posiciones iniciales mientras que en 1 se guardan las posiciones modificadas**.** ***Por motivos de codificación, en algunos caso se requirió de variables y en otros fueron directamente los valores flotantes.***

  Variable:
  KeyFrame: Dicho arreglo se encarga de guardar las modificaciones realizadas a los objetos, dichas se estarán visualizando en otra función.

Variable	***Valor***	***Descripción***
KeyFrame[0].posSY	posSomY	Posición inicial del sombrero en el eje Y, es en la cual, dicho objeto se encuentra al apretar la tecla O.
KeyFrame[0].posSZ	posSomZ	Posición inicial del sombrero en el eje Z, es en la cual, dicho objeto se encuentra al apretar la tecla O.
KeyFrame[0].rotSX	rotSomX	Rotación inicial del sombrero en el eje X, es en la cual, dicho objeto se encuentra al apretar la tecla O. En este caso, se cargan los objetos sin modificaciones, por lo que las posiciones y las rotaciones son realizadas en OpenGL.
KeyFrame[0].rotSY	rotSomY	Rotación inicial del sombrero en el eje Y, es en la cual, dicho objeto se encuentra al apretar la tecla O.
KeyFrame[1].posSY	3.8	Posición final del sombrero en el eje Y, es en la cual, dicho objeto se encuentra después de apretar la tecla L.
KeyFrame[1].posSZ	-4.4	Posición final del sombrero en el eje Z, es en la cual, dicho objeto se encuentra después de apretar la tecla L.
KeyFrame[1].rotSX	45.0	Rotación final del sombrero en el eje X, es en la cual, dicho objeto se encuentra después de apretar la tecla L.
KeyFrame[1].rotSY	-180.0	Rotación final del sombrero en el eje Y, es en la cual, dicho objeto se encuentra después de apretar la tecla L.Se tuvo que modificar en dos ejes debido a la posición que se requiere.

KeyFrame[0].posFY	posFloY	Posición inicial del florero en el eje Y, es en la cual, dicho objeto se encuentra después apretar la tecla K.
KeyFrame[0].rotFY	rotFloY	Rotación inicial del florero en el eje Y, es en la cual, dicho objeto se encuentra después apretar la tecla K.
KeyFrame[0].rotFXZ	rotFloXZ	Rotación inicial del florero en el eje X y Z, es en la cual, dicho objeto se encuentra después apretar la tecla K.
KeyFrame[1].posFY	6.0	Posición final del florero en el eje Y, es en la cual, dicho objeto se encuentra después apretar la tecla K.
KeyFrame[1].rotFY	-90.0	Rotación final del florero en el eje Y, es en la cual, dicho objeto se encuentra después de apretar la tecla K.
KeyFrame[1].rotFXZ	-40.0	Rotación final del florero en el eje X y Z, es en la cual, dicho objeto se encuentra después de apretar la tecla K.

- ***void interpolation(void)***

  Función que se realizar la interpolación de las animaciones con Keyframes. Lo que realiza es aritmética, ya que de la estructura KeyFrame se obtiene la posición siguiente menos la actual sobre la variable i_max_steps. Dicho valor cambia dependiendo del objeto que se trate, si es el sombrero o el florero.

- ***void resetElements(void)***

  Función que se encarga de reestablecer los valores iniciales a los diferentes objetos con el fin de que su posición o rotación no se encuentren editadas. Se encuentra un *if* para verificar si lo que se quiere restablecer de valores es el sombrero o el florero, se controla con una variable booleana. Los valores descritos a continuación son los dos únicos valores que pueden tomar, puesto que la estructura KeyFrame es de tamaño 2.

Variable	***Valor Anterior***	***Nuevo Valor***
posSomY	KeyFrame[1].posSY	KeyFrame[0].posSY
posSomZ	KeyFrame[1].posSZ	KeyFrame[0].posSZ
rotSomX	KeyFrame[1].rotSX	KeyFrame[0].rotSX
rotSomY	KeyFrame[1].rotSY	KeyFrame[0].rotSY
posFloY	KeyFrame[1].posFY	KeyFrame[0].posFY
rotFloY	KeyFrame[1].rotFY	KeyFrame[0].rotFY
rotFloXZ	KeyFrame[1].rotFXZ	KeyFrame[0].rotFXZ
som	true	false
	false	true

- ***int main( )***

  Función principal del proyecto. Se encarga de realizar las configuraciones de GLFW, modifica la cámara para tener mejor visualización, configuraciones de las ventanas, establece las funciones callbacks, establece el GLEW, define las dimensiones del puerto de vista. Además de que modifica las opciones de OpenGl. Después, modifica los shaders, también se realizan la carga de modelos y texturas para que el SkyBox se pueda visualizar de la manera correcta; carga de los modelos realizados previamente, cada uno de ellos llevan el nombre en minúsculas de lo que representan, así como su respectiva ubicación. Se cargan los vértices del SkyBox y después se manda a llamar la función saveFrame() debido a que se cargan con ella las diferentes animaciones.

  Se realizan las configuraciones necesarias para el VBO, VAO y EBO. Se carga la imagen que se utilizará un poco después para la figura primitiva, se verifica que fue cargada correctamente para después modificar el shader de la luz, texturas, etc.

  Se ingresa a un ciclo en el cual verifica mientras no esté cerrada la ventana del ambiente, realiza las siguientes instrucciones:
  Configura el currentFrame con ayuda de la función glfwGetTime. Se realizan las llamadas necesarias para verificar si ya se presionaron teclas con animación, como ejemplo. Se colocan los objetos necesarios, los que no tienen animación alguna solamente se resetean en la posición origen (dicha posición previamente modificada en Blender), además de dibujarlos con el Shader. Sin embargo, las animaciones con animaciones, se colocaron primero con una translación y después con la rotación.

  Finalmente, se dibuja el SkyBox y después se vacían los buffers. Retorna un 0.

- ***void DoMovement( )***

  Modifica posición de la cámara, también se encarga de recibir los cambios en las teclas con el fin de realizar las animaciones. Dichos modificadores dependen de las teclas apretadas.

Teclas	***Valor***	***Descripción***
W || UP		Movimiento de la cámara hacia arriba.
S || DOWN		Movimiento de la cámara hacia abajo.
A || LEFT		Movimiento de la cámara hacia la izquierda.
D || RIGHT		Movimiento de la cámara hacia la derecha.

1	movPuy = 90	Realiza el movimiento de apertura de la puerta con un ángulo de 90°
2	movPuy = 0	Realiza el movimiento de cerrar de la puerta con un ángulo de 0°.

- ***void animacion()***

  Función que se encarga de reproducir las animaciones complejas, es decir, con keyframes.

  Variables:
  - **playS:** Reproduce la animación correspondiente al sombrero.
  - **playF:** Reproduce la animación correspondiente al florero.

Variable	***Valor***	***Descripción***
playS	true	Verifica al contador, si es así entonces se termina la animación, en el caso del sombrero se queda en la posición nueva.
	false	Verifica si se tienen animaciones guardadas en la estructura keyFrame, si es así, entonces con un if reproduce las animaciones del sombrero.
playF	true	Verifica al contador, si es así entonces se termina la animación, en el caso del florero se manda a llamar a la función resetElements, para cambiar la posición final a la inicial.
	false	Verifica si se tienen animaciones guardadas en la estructura

		keyFrame, si es así, entonces con un if reproduce las animaciones del florero.

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

  Se manda a llamar a esta función cada vez que se presiona una tecla. Se verificará si la tecla ESC fue presionada, si lo fue, entonces se sale de la ventana. Verificará si determinadas teclas fueron presionadas, si es así, entonces se realizan las animaciones correspondientes (pueden ser tanto complejas como sencillas).

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

  Se obtiene el movimiento del mouse y actualiza la posición de la cámara gracias al mouse.

Lógica de las animaciones

Para que estas fueran de manera automática, se realizaron ciclos utilizando variables auxiliares, esto con el fin de ir variando los valores de las translaciones o bien, variando las rotaciones. Para el ciclo, se utilizó solamente while.

- *Sencillas*

Fueron creadas muy fáciles, ya que cuando el usuario presiona una tecla, se cambian los valores de mínimo.

- *Complejas*

Se tuvo de guía la práctica 11 del curso, ya que siguiendo la lógica de ésta, es como se realizaron las dos animaciones. Además, se utilizaron diferentes variables para poder diferenciar cada una de ellas. Sin embargo, cuando se decidió la lógica de la segunda, se tuvo en mente siempre la lógica del mundo real, es decir, mientras que el florero puede volver a su lugar por la gravedad, el sombrero no puede subir a su lugar de origen, es por eso que solo en esa animación se utilizan los teclas. En el código, se tiene una variable denominada **som** que es la que se encarga de verificar que, cuando se aprieta la tecla del florero o la tecla para regresar a su lugar el sombrero, se reinicien los valores del objeto que se desea y no los dos.

Referencias

- *DIBUHO 3D. (2018, 12 abril). Floral Pillo. TurboSquid. https://www.turbosquid.com/3d-models/3d-floral-pillow-1275999*
- *D.I.B.U.H.O.D. (2018, 17 abril). Plate Bowl var 1. TurboSquid. https://www.turbosquid.com/3d-models/3d-pbr-plates-bowls-1277572*
- *Low Poly Cartoon Cowboy Hat Free. (2019, 16 agosto). TurboSquid. https://www.turbosquid.com/3d-models/3d-cowboy-hat-model-1437165*
- *MARCO DE MADERA GIF. (s. f.). [Ilustración]. Pinterest. https://www.pinterest.com/pin/833799318485380521/*
- *Oropeza, B. (2020, 12 febrero). Vase. TurboSquid. https://www.turbosquid.com/3d-models/vase-games-3d-1509284*
- *Van Gogh, V. (s. f.). La Noche Estrellada [Fotografía]. https://historia-arte.com/obras/noche-estrellada-van-gogh. https://historia-arte.com/obras/noche-estrellada-van-gogh*