UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE INGENIERÍA

COMPUTACIÓN GRÁFICA E INTERACCIÓN HUMANO-COMPUTADORA

Proyecto Final

Integrantes:

ESLAVA RICO JOSÉ CRUZ
HERNÁNDEZ GUTIÉRREZ LESLIE VIVIAN
RIVERA LIMA CARLOS EDUARDO

Fecha de entrega: 26 de enero de 2021

RESUMEN

Ya que la computación gráfica es ahora una parte importante de las nuevas tecnologías en desarrollo, tener esta materia incluida en el plan de estudios de los ingenieros es fundamental.

A lo largo de este proyecto, encontramos diferentes retos que fueron resueltos utilizando el conocimiento adquirido durante el semestre y obteniendo información de varios sitios web y videos disponibles en internet.

Aplicamos las estrategias y modelados vistos en clase, además de otras que no conocíamos sino hasta que comenzamos nuestra investigación, los resultados pueden presentarse de una manera ligeramente diferente a la planeada, pero la funcionalidad del proyecto está presente.

INTRODUCCIÓN

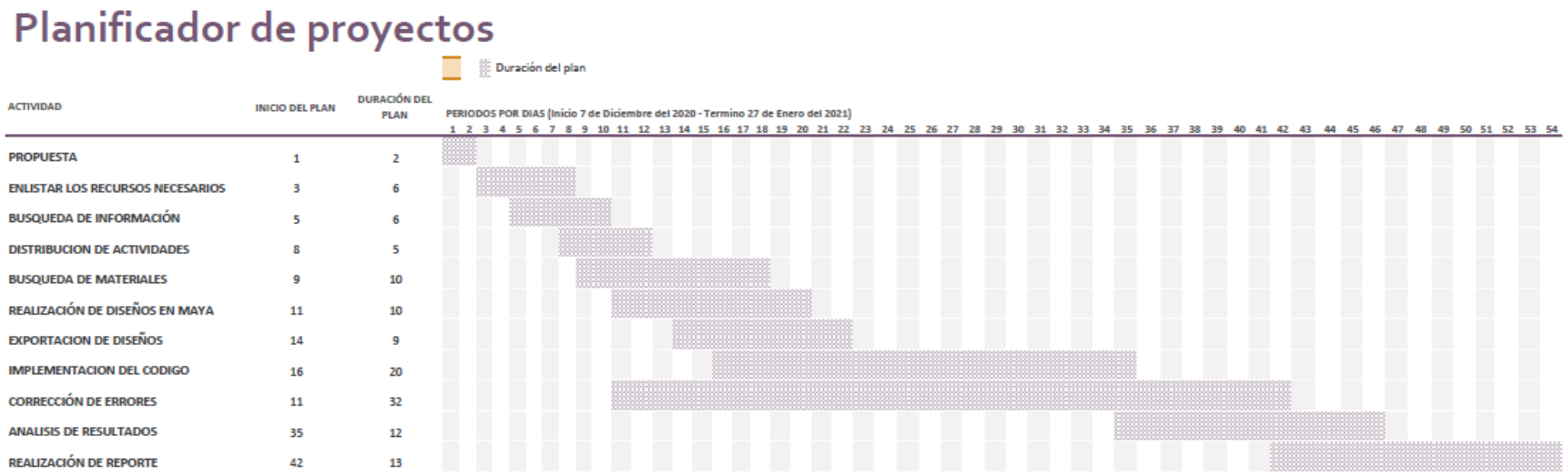
El término “Computación gráfica” hace referencia a cualquier estrategia o recurso utilizado en la creación o manipulación de objetos o imágenes a través de la computadora, incluyendo aquellos que son animados. Este campo ha crecido bastante durante los últimos años, y se ha vuelto un campo muy grande. La computación gráfica puede ser utilizada en la fotografía digital, en el entretenimiento, el diseño UI, renderizado, animaciones, dispositivos electrónicos y en otras tecnologías en donde sea requerido. Hay diferentes herramientas utilizadas para implementar la computación gráfica.

El principal centro de atención de la computación gráfica es la creación de imágenes en tres dimensiones, pero en casi todos los casos, el resultado de un proyecto 3D es una imagen de dos dimensiones y su correcta manipulación. Una imagen presentada en una computadora se hace a través de los pixeles. Una imagen digital se especifica utilizando un sistema de coordenadas, que establece una relación entre sus números y puntos geométricos, ya sean dos o tres dimensiones, las coordenadas se asignan en alguno de los tres planos, ya sea “x”, “y” o “z”. La manipulación de estas imágenes, junto con la creación de formas básicas, nos ayudará a formar lo que llamamos “objetos”.

Cuando hablamos de gráficos 3D, nos referimos a la técnica llamada “modelado geométrico”, cuyo punto de inicio comienza al intentar construir un “mundo artificial” como una colección de formas geométricas básicas, ubicadas correctamente en un plano tridimensional. Los objetos pueden tener atributos que, combinados con otras propiedades, determinan la apariencia de los objetos. Los bloques más pequeños con los cuales debemos trabajar, como líneas o triángulos, son llamados “primitivas geométricas”. Una escena compleja puede contener un gran número de primitivas y sería muy difícil de crear si sólo se tienen las coordenadas explícitas para cada primitiva.

Para modificar estos objetos, utilizamos algo llamado “Transformaciones geométricas”, lo cual se utiliza para ajustar el tamaño, la orientación y la posición de un objeto geométricos. Existen tres tipos de transformaciones básicas, como escalamiento, rotación y traslación. El escalamiento se utiliza para cambiar el tamaño de un objeto, la rotación es utilizada para cambiar la orientación de un objeto y la rotación es utilizada para rotar el objeto utilizando cierto ángulo y lo hace sobre un eje específico.

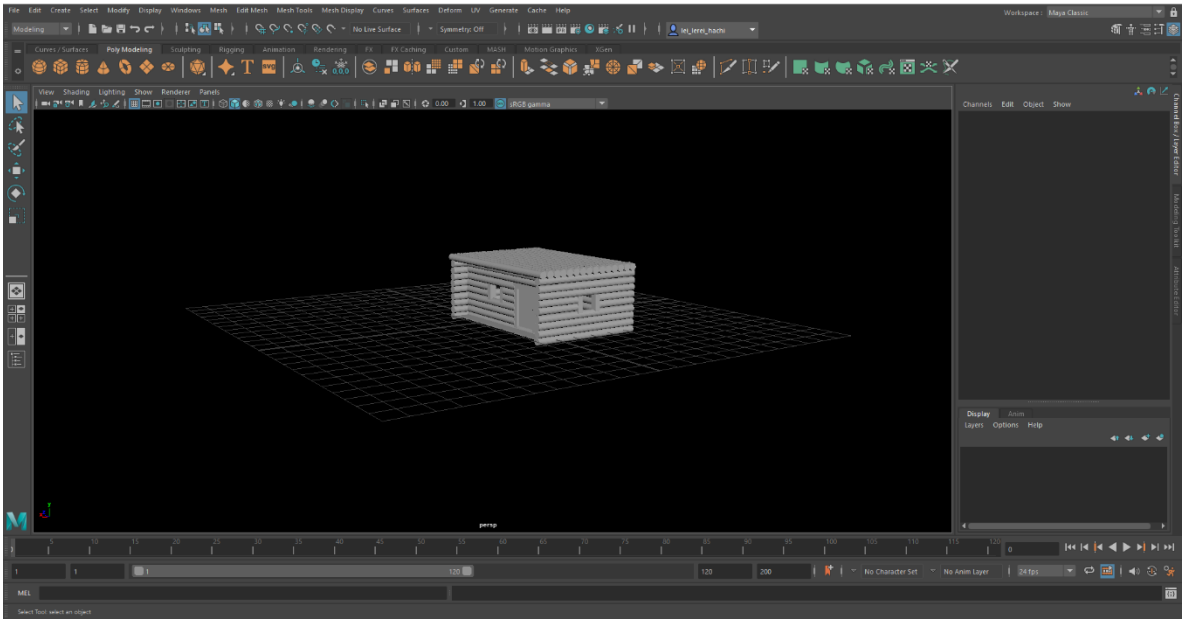
CRONOGRAMA DE ACTIVIDADES



MATERIALES Y MÉTODOS

Para el desarrollo de este proyecto, utilizamos el software Autodesk Maya, el cual es un programa dedicado al desarrollo de gráficos 3D, efectos especiales y animaciones. Se caracteriza por su potencia y las posibilidades de expansión y personalización de su interfaz y herramientas.

Para todos los objetos que incluimos en nuestro proyecto, utilizamos Maya para poder manipularlos y transformarlos, escalarlos y rotarlos, después, los exportamos como objetos para poder añadirlos a nuestro código.

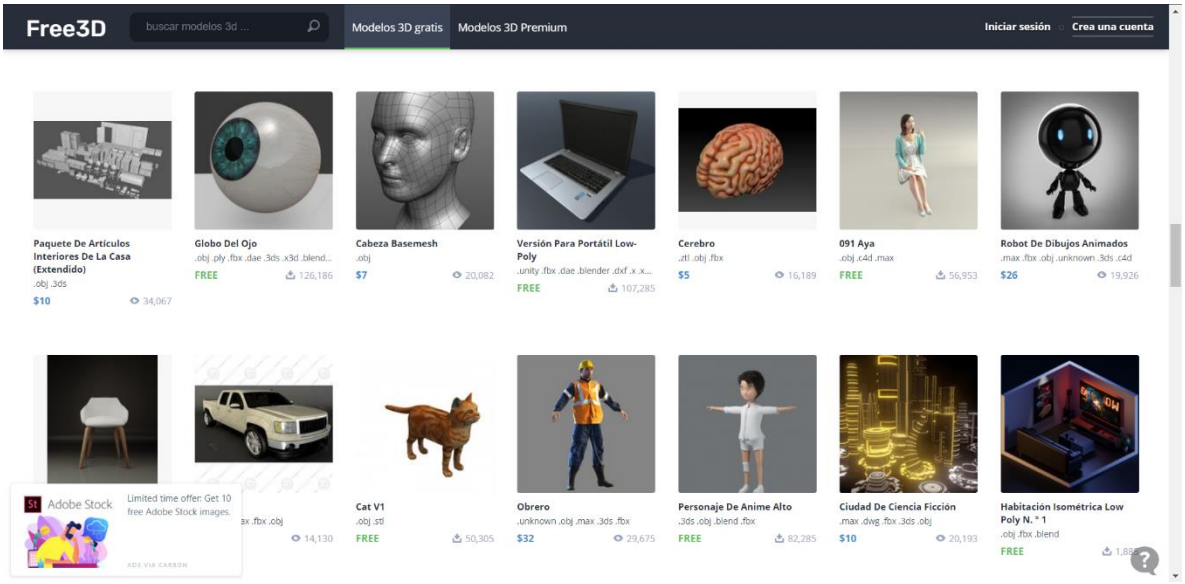


1 - Sofware Maya manipulación de un objeto

El uso de este software nos permitió manipular los objetos de una manera más rápida a través de su interfaz, encontramos fácil la manera de colocar nuestros objetos en nuestro código de Visual Studio, esta herramienta también le permite a los usuarios animar objetos de acuerdo a sus necesidades.

LOS ENLACES DEL REPOSITORIO, LOS EJECUTABLES Y OTROS RECURSOS SE ENCUENTRAN AL FINAL DEL DOCUMENTO.

También utilizamos diferentes recursos de internet, algunos de paga, otros gratis, que incluían texturas o objetos ya diseñados para nuestro proyecto. Una de las páginas en las que nos apoyamos fue https://free3d.com/es/modelos-3d/obj donde pudimos encontrar una variedad de objetos que otros usuarios comparten a través de ese sitio web.



2.- Pagina de modelos en 3D (Free3D)

Además de las dos herramientas mencionadas anteriormente, utilizamos las transformaciones básicas vistas en laboratorio para poder manipular nuestro objeto, tales como traslación, rotación y escala, utilizando los atributos de los objetos en nuestro código.

```
606        //Carga de modelo
607        // prueba CABAÑA 2
608        view = camera.GetViewMatrix();
609        model = glm::mat4(1);
610        //model = glm::translate(model, PosIni + glm::vec3(movKitX, 0, movKitZ));
611        model = glm::translate(model, glm::vec3(-56.0f, 0.0f, 35.0f));;///-56se pasa,//bien,//
612        model = glm::rotate(model, glm::radians(rotKit), glm::vec3(0.0f, 0.0f, 90.0f));
613        model = glm::scale(model, glm::vec3(1.2f, 1.2f, 1.2f));
614        glUniformMatrix4fv(modelLoc, 1, GL_FALSE, glm::value_ptr(model));
```

3.- Transformaciones básicas

EXPERIMENTOS

Para nuestras animaciones, utilizamos valores booleanos con primitivas sencillas dentro de nuestro código. Lo realizamos para los recorridos tanto del bote como de la mariposa.

```
float movKitX1 = 0.0;//mariposa
float movKitZ1 = 0.0;//mariposa
float rotKit1 = 0.0;//mariposa

bool circuito = false;
bool recorrido1 = true;
bool recorrido2 = false;
bool recorrido3 = false;
bool recorrido4 = false;
bool recorrido5 = false;


bool circuito1 = false;//mariposa
bool recorrido11 = true;//mariposa
bool recorrido21 = false;//mariposa
bool recorrido31 = false;//mariposa
bool recorrido41 = false;//mariposa
bool recorrido51 = false;//mariposa
```

4.-Valores Booleanos

Para la carga de nuestros modelos, utilizamos las siguientes funciones y referenciamos correctamente en dónde estaban almacenados nuestros objetos. Les dimos un nombre a cada uno y su localización en nuestro folder, así no tendríamos problemas al correr el programa.

```
Model Carroseria((char*)"Models/Carro/12150_Christmas_Tree_V2_L2.obj");
//Model LLanta((char*)"Models/Carro/Wheel.obj");
Model Piso((char*)"Models/Carro/Piso.obj");
Model Cabana((char*)"Models/Carro/WoodenCabinObj.obj");
Model CaparazonCabeza_Tortuga((char*)"Models/Carro/cabeza_caparazon.obj");
Model BrazoDerecho_Tortuga((char*)"Models/Carro/tortuga_brazoderecho.obj");
Model BrazoIzquierdo_Tortuga((char*)"Models/Carro/tortuga_brazoizquierdo.obj");
Model PataDerecha_Tortuga((char*)"Models/Carro/tortuga_pataderecha.obj");
Model Muneco((char*)"Models/Carro/LowResOBjNightMare.obj");
Model Boat((char*)"Models/Carro/boat.obj");
Model Sofa((char*)"Models/Carro/sofa.obj");
Model Comedor((char*)"Models/Carro/3dstylish-fdb001.obj");
Model Estufa((char*)"Models/Carro/fkc.obj");
Model Refri((char*)"Models/Carro/B_Daily_R_St_N_Electric_0002.obj");
Model Arbusto((char*)"Models/Carro/Low Grass.obj");
Model Camioneta((char*)"Models/Carro/Truck_obj.obj");
Model Bocho((char*)"Models/Carro/bocho.obj");
Model Bano((char*)"Models/Carro/Toilet.obj");
Model CuartoBano((char*)"Models/Carro/banocomplete.obj");
Model Banera((char*)"Models/Carro/bath_obj.obj");
Model Radio((char*)"Models/Carro/radio_high.obj");
```

5.- Funciones para carga de modelos

Para la carga de modelos, utilizamos las funciones con transformaciones para posicionr correctamente nuestros modelos. Este proceso lo repetimos para cada uno de los objetos en nuestro mapa.

```
//Carga de modelo
//tortuga Cabeza y Caparazón
view = camera.GetViewMatrix();
model = glm::mat4(1);
//model = glm::translate(model, PosIni + glm::vec3(movKitX, 0, movKitZ));
model = glm::translate(model, glm::vec3(-56.0f, 0.4f, 11.5f));;
model = glm::rotate(model, glm::radians(rotKit), glm::vec3(0.0f, 1.0f, 0.0));
model = glm::scale(model, glm::vec3(0.02f, 0.02f, 0.02f));
glUniformMatrix4fv(modelLoc, 1, GL_FALSE, glm::value_ptr(model));

CaparazonCabeza_Tortuga.Draw(lightingShader);
```

6.- Funciones con transformaciones

Para el recorrido de nuestra mariposa y nuestro bote, utilizamos estructuras básicas condicionales, en las que la mariposa respondía a una de las dos teclas asignadas para este propósito, y logramos hacer que su movimiento fuera alrededor del mapa

```
void animacion1()
{

    //Movimiento
    if (circuito1)
    {
        if (recorrido11)
        {
            movKitZ1 += 0.1f;
            if (movKitZ1 > 90)
            {
                recorrido11 = false;
                recorrido21 = true;
            }
        }
        if (recorrido21)
        {
            rotKit1 = 90;
            movKitX1 += 0.1f;
            if (movKitX1 > 90)
            {
                recorrido21 = false;
```

7.- Estructuras básicas condicionales para recorrido

Cámara del Proyecto

La cámara nos permite desplazarlos a lo largo y ancho del terreno gráfico.

Teclas:

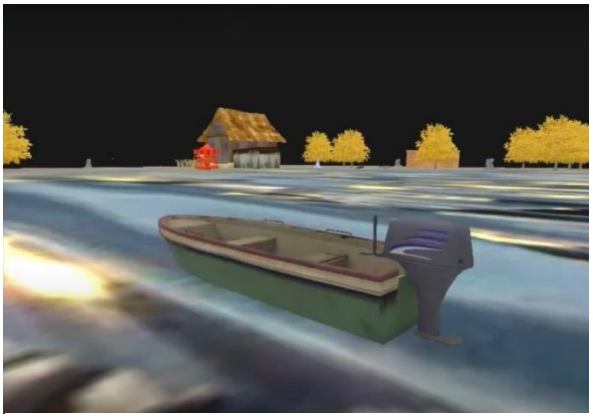
A (Izquierda)

W (Frente)

D (Derecha)

S (Atrás)

RESULTADOS

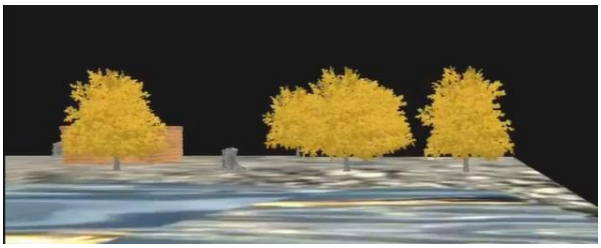


8.- Bote

Para nuestro primer objeto, utilizamos un bote en el lago que se puede observar en la imagen, este responde a dos teclas (I) y (O) con las cuales podemos hacer que avance en línea recta y se detenga, respectivamente.

9.- Mariposa

Para la mariposa, utilizamos una animación que hace que haga un recorrido alrededor del mapa, las alas no se mueven, sino que son estáticas.



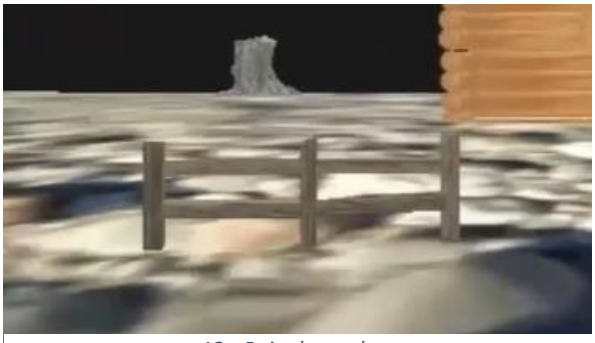
10.- Árboles

Los árboles alrededor de nuestro mundo virtual no siguen ningún patrón específico, no están animados, se encuentran estáticos.

11.- Tortuga

La tortuga se encuentra junto al lago, y se trata de una animación incompleta, ya que no logramos hacer que se movieran hacia el agua utilizando las patas.

.



12.- Reja de madera

Añadimos también una reja de madera que no terminamos de colocar adecuadamente alrededor de la cabaña.

13.- Cabaña de madera

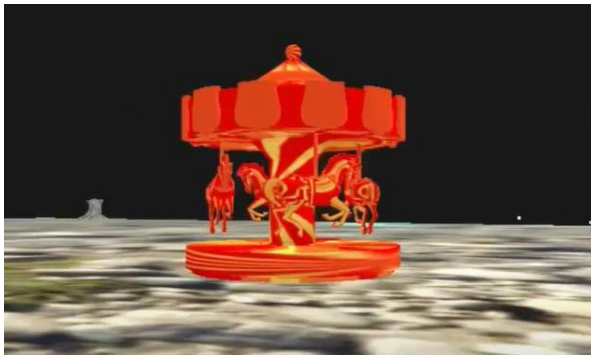
Añadimos también una cabaña de manera, en la que planeamos colocar muebles adentro para mayor realismo.

14.- Muñeco de nieve

El muñeco de nieve es acorde a la temporada navideña que tuvo lugar en Diciembre, ya que deseábamos añadir un poco de este tema.

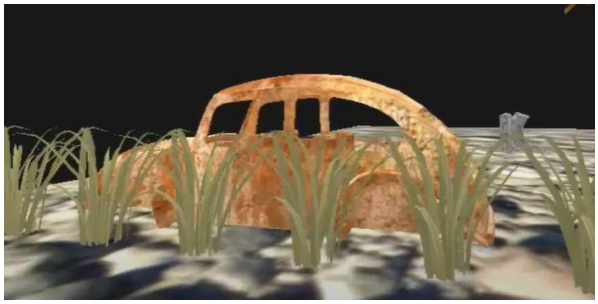
15.- Baño

La habitación aledaña de la primera cabaña incluía un baño pero también está incompleto ya que le falta una puerta, con la cual se planeaba incluir otra animación.



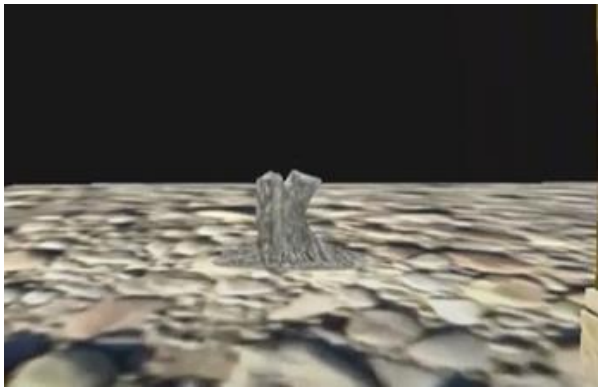
El carrusel cuenta con una animación sencilla que no se activa con el teclado sino que comienza desde el inicio del proyecto.

16.- Carrusel



El carro es estático también y no cuenta con ninguna animación, aunque al inicio se trataba de mover también.

17.- Carro



Estos troncos son también parte de nuestra idea, ya que tiene relación con el entorno natural.

18.- Tronco

Para la habitación contamos con diferentes objetos que representan los muebles de la cabaña, tales como la cama, una mesa, una cocina y un árbol de navidad.

19.- Muebles

VIDEOS DE LAS ANIMACIONES

Animación de la puerta: **https://www.youtube.com/watch?v=V-LFFMtP3jw**

Para la rotación de la puerta, establecimos un valor en grados para la posición inicial, cuenta con un contador que irá creciendo en un número determinado y cuando este alcance los grados de la posición final, se detendrá. La animación se activa con dos teclas, L para que inicie la animación y K para que se detenga.

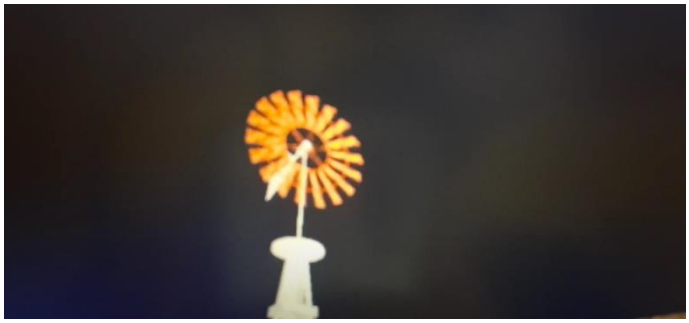
```
void porton() {
	if (puerta) {
		if (rotPuerta < 416)//336
			for (int i = 0; i <= 198; i++)//335
				rotPuerta += 5.0f;
	}
	else
		rotPuerta = 0;
}
```

Animación del molino: **https://www.youtube.com/watch?v=CjR_Nwrzmo8**

Para la rotación del molino, utilizamos la función rotación perteneciente a OpenGL, y establecimos que girara sobre un eje para obtener el movimiento de un molino real.

```
model = glm::rotate(model, (float)glfwGetTime(), glm::vec3(0.0f, 0.0f, 1.0f));
```

Le indicamos al código que debe rotar nuestro modelo **model** en un ciclo y utilizamos la función **glwGetTime()** para obtener un ángulo respecto al tiempo, la cual llama el valor del timer de GLFW.



Carrusel: **https://www.youtube.com/watch?v=Dn7lowy0VvU**

Este objeto está compuesto de dos partes, la primera parte es la base y la segunda parte es la torre que gira sobre la base como primer objeto. Lo descompusimos en dos partes para poder realizar esta animación. Para esta animación usamos también la función de **rotate** con la que cuenta OpenGL.

```
model = glm::rotate(model, (float)glfwGetTime(), glm::vec3(0.0f, 1.0f, 0.0f));
```

Mariposa: **https://www.youtube.com/watch?v=yTpYKH8RHa8**

Para la mariposa creamos un recorrido predeterminado mediante funciones, y empleamos las teclas B y N para comenzar el recorrido y detenerlo respectivamente, la mariposa realiza un recorrido de manera cuadrada a lo largo de todo el plano.

En este extracto del código podemos observar que iniciamos el recorrido con un circuito y condicionales anidadas, establecemos primero una posición y un ángulo para la mariposa, y la vamos incrementando, una vez que llega a un límite, saltamos a un ciclo siguiente que va a mover su posición en otro de los ejes, el primero será el eje Z y después cambiará al eje X.

```
void animacion1()
{
	//Movimiento de la mariposa
	if (circuito1)
	{
		if (recorrido11)
		{
			movKitZ1 += 0.1f;
			if (movKitZ1 > 90)
			{
				recorrido11 = false;
				recorrido21 = true;
			}
		}
		if (recorrido21)
		{
			rotKit1 = 90;
			movKitX1 += 0.1f;
			if (movKitX1 > 90)
			{
				recorrido21 = false;
				recorrido31 = true;
			}
		}
...
```

Bote: https://www.youtube.com/watch?v=dpaB2fOkYc4

Para nuestro objeto bote, realizaremos el mismo procedimiento, lo activamos con la tecla I y lo detenemos con la tecla O, el recorrido se hace también con una posición inicial y una posición final, utilizando geometría analítica, utilizamos la recta de la pendiente mara modificar el traslado del bote.

Este personaje es capaz de moverse en todo nuestro plano, además de poder mover sus dos extremidades. Se ha empleado Keyframe para su animación, sus movimientos son dependiente de teclas de la computadora así como su posición en la cámara es dependiente tanto de teclas como del mouse.

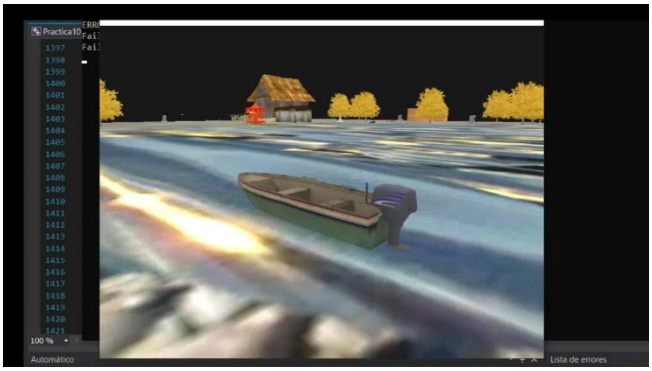
Teclas:
1 (Rotación en su propio eje)
6 (Mueve brazo izquierdo al frente)
7 (Mueve brazo izquierdo hacia atrás)
8 (Mueve brazo derecho al frente)
9 (Mueve brazo derecho hacia atrás)

Movimientos de desplazamiento del personaje.

El personaje puede interactuar con todo el terreno.
Teclas:
G (al frente)
J (para atrás)
Y (a la Izquierda)
H (hacia la derecha)

En este vídeo observamos un recorrido general de una versión previa de nuestro proyecto.

https://youtu.be/XXKFHVVM1F8



También aquí podemos ver algunas animaciones previas al resultado final.

https://youtu.be/5q39ymKkKBY

Este video muestra el manejo de las luces en el entorno gráfico, así como el skicube que nos muestra un entorno de montañas.

https://youtu.be/AZfMRHqCdF4

En la siguiente liga mostramos un recorrido por todo el entorno, dando así una completa visualización de todo el proyecto en su versión final, las animaciones mostradas anteriormente no son afectadas en esta última versión

https://youtu.be/yhZ7QV5y_UI

CONCLUSIONES

Retomando lo mencionado en nuestro documento y la visualización del video que se entregó se puede observar todo el trabajo que se requirió para poder realizar este proyecto que engloba todos los conocimientos adquiridos no solo en el laboratorio, sino en teoría que a pesar de no ver mucha práctica, las investigaciones realizadas nos ayudaron a comprender varios aspectos que se nos presentaron a lo largo de esta implementación. El objetivo de este proyecto que era hacernos usar las herramientas para poder crear un entorno grafico se logró cumplir, incluso utilizamos una herramienta diferente a la vista en clase, porque nos dimos cuenta de que esta nueva herramienta nos facilitaba mejor las cosas y tenía más opciones de trabajo las cuales pudimos usar para una mejor implementación y definición de este proyecto.

ENLACES

- Ejecutable del Proyecto

https://github.com/CarlosE101/ProyectoCGIHC2021-1_dinamita

- Archivos CPP y Filtros

https://github.com/CarlosE101/-ProyectoCGIHC2021-1_DinamitaCPP-

- Drive con el proyecto completo

https://drive.google.com/drive/folders/1xKEzJHmqUUN-VmvNCjKMV-5e_1jIlQE3

CREDITOS CORRESPONDIENTES

Gran parte de las imágenes de textura son una mezcla entre nuestra edición y capturas de múltiples imágenes de la web.

A continuación se muestran las paginas de las cuales se tomaron los objetos y texturas, teniendo en cuenta las licencias correspondientes que hay para el uso de estos. Debido a la gran cantidad de objetos utilizados se muestran solo las ligas generales de las paginas utilizadas.

www.turbosquid.com

www.free3d.com

www.artec3d.com

www.gimp.org

www.autodesk.mx

REFERENCIAS

Introducción a la Computación gráfica – GeeksforGeeks, (2020). Recuperado el 18 de enero de 2021 de, https://www.geeksforgeeks.org/introduction-to-computer-graphics/

Elementos de los gráficos 3D. Recuperado el 18 de enero de 2021 de, http://math.hws.edu/graphicsbook/c1/s2.html

Introducción a la computación gráfica: introducción. Recuperado el 18 de enero de 2021, de https://www.cs.uic.edu/~jbell/CourseNotes/ComputerGraphics/IntroToCG.html

Cómo moverte por Maya y crear geometrías. (2019) Recuperado el 18 de enero de 2021, de https://www.ilerna.es/blog/aprende-con-ilerna-online/imagen-sonido/maya-como-moverte-por-el-viewport-y-como-crear-geometrias/

Maya. Soporte y Documentación. Recuperado el 18 de enero de 2021, de https://knowledge.autodesk.com/es/support/maya/getting-started/caas/simplecontent/content/maya-documentation.html

Transformaciones. Recuperado el 18 de enero de 2021, de https://learnopengl.com/Getting-started/Transformations