



UNIVERSIDAD NACIONAL AUTÓNOMA DE
MÉXICO

FACULTAD DE INGENIERÍA
DIVISIÓN DE INGENIERÍA ELÉCTRICA
INGENIERÍA EN COMPUTACIÓN



LABORATORIO DE COMPUTACIÓN GRÁFICA e
INTERACCIÓN HUMANO COMPUTADORA

Manual y documentación del proyecto final

PROFESOR: Ing. Luis Sergio Valencia Castro

GRUPO: _____ 03

Manual y Documentación: Centro Turístico Recreativo

ALUMNO: Cervantes García Eduardo

N° DE LISTA: _____ -

SEMESTRE: 2025-1

FECHA DE ENTREGA: 14 de noviembre de 2024

OBSERVACIONES:

CALIFICACIÓN: _____

HOJA DE EVIDENCIAS

PROYECTO FINAL: [CENTRO TURÍSTICO RECREATIVO]

318123602

Tabla de contenido

1. Manual de uso.....	4
Instalación de la aplicación	4
Uso del software	8
2. Cronograma de actividades realizadas	17
3. Herramienta colaborativa de software – repositorio	20
4. Desarrollo (descripción de actividades)	21
Modelado.....	21
Texturizado	27
Iluminación	28
Animación	30
Sonido	35
5. Estudio técnico para la venta de la aplicación.....	35
6. Conclusiones individuales	38
7. Referencias bibliográficas	39
Modelos (4 modelos humanoids de animaciones por KeyFrames)	40
Texturas.....	40
Sonido	41

1. Manual de uso

Este manual explicará la instalación del software mediante el uso de un instalador, con un paso a paso y dando sugerencias para evadir problemas con la instalación. También, se explicará cómo usar y sacar el máximo provecho al software, mencionando los controles de los periféricos básicos que se pueden usar en el mismo y como usarlos, es decir, su uso esperado y como se les puede sacar el provecho mencionado.

Instalación de la aplicación



Ilustración 1

El primer paso para realizar una instalación correcta es dirigirse a la carpeta donde decidió almacenar el archivo dentro de su sistema de archivos local.

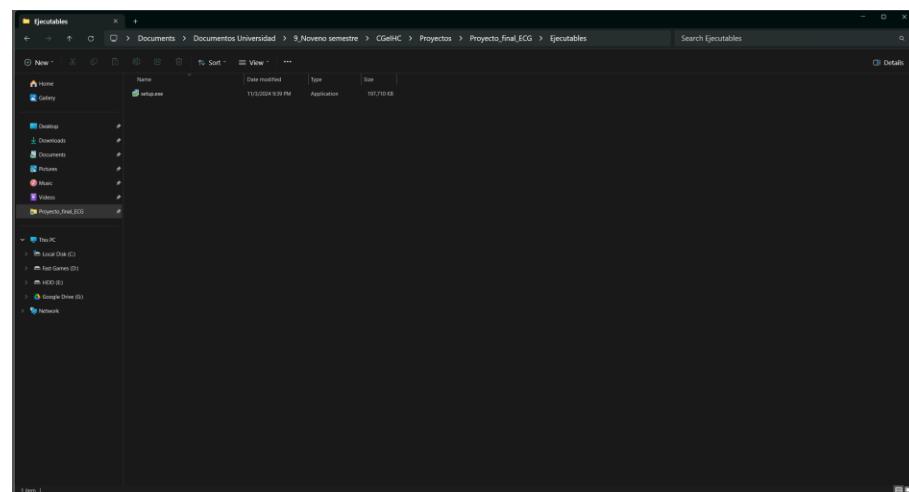


Ilustración 2

A pesar de que en las pruebas realizadas no fue necesario, se sugiere antes de seguir con los demás pasos, desactivar su antivirus, EDR o la protección que proporciona Windows mediante el software de “Defender”. Para ello de clic en la tecla de Windows y escriba “Security”.

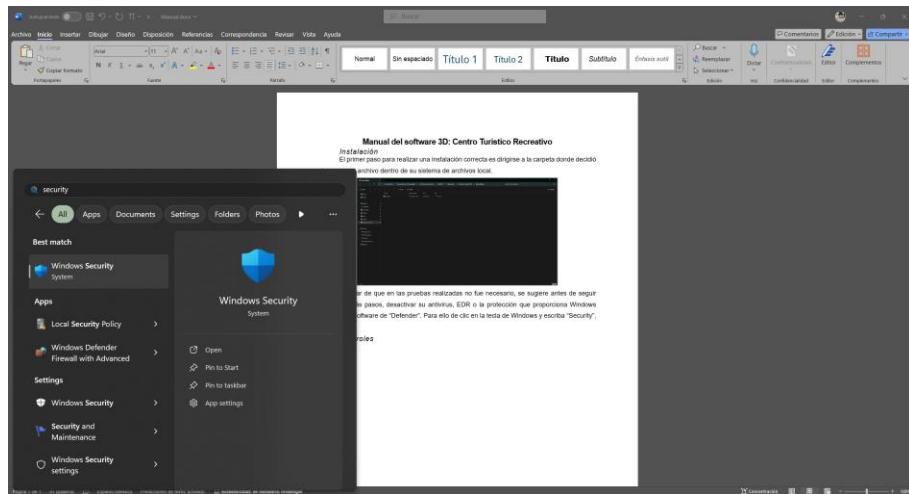


Ilustración 3

Posteriormente, de clic sobre la primera búsqueda que arroja la indexación y en la pantalla desplegada de clic en el apartado de “Virus and threat” y dentro de ese modulo del programa de clic en la slider de “Real-time protection”. Lo anterior, evitará problemas al instalar el software ya que no contiene una firma reconocida.

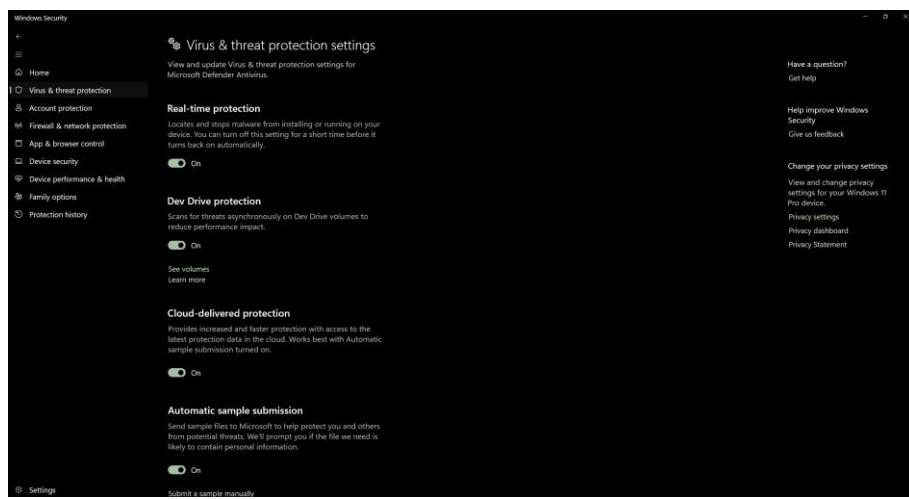


Ilustración 4

Ahora, regrese a la carpeta donde se encuentra el “setup.exe” y de clic derecho sobre el mismo para seleccionar “Ejecutar como administrador”.

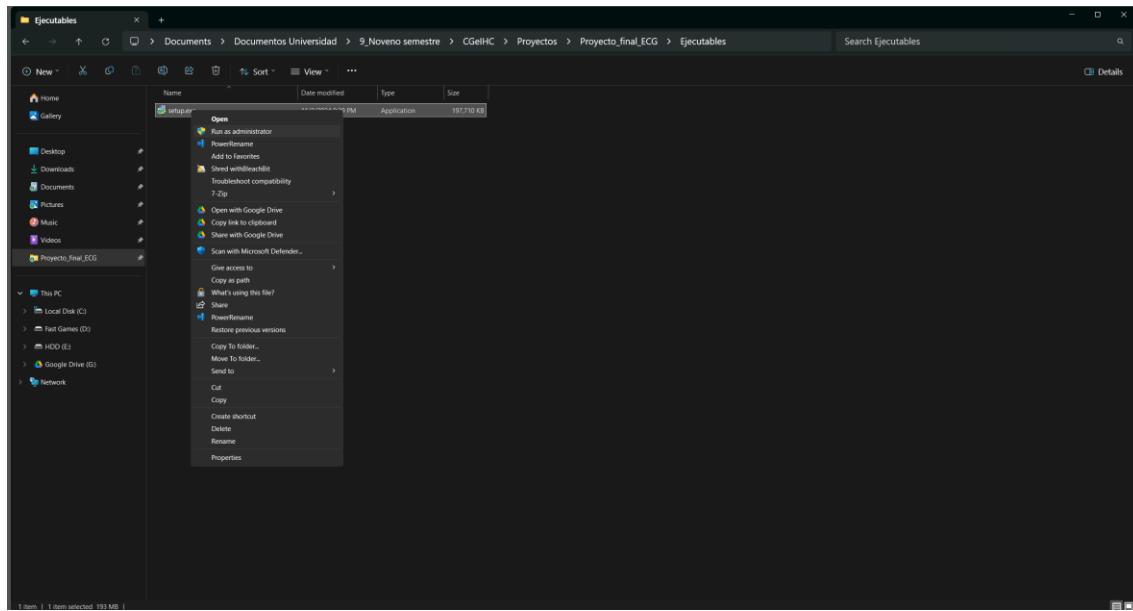


Ilustración 5

En la siguiente pantalla escoja el idioma de su preferencia.

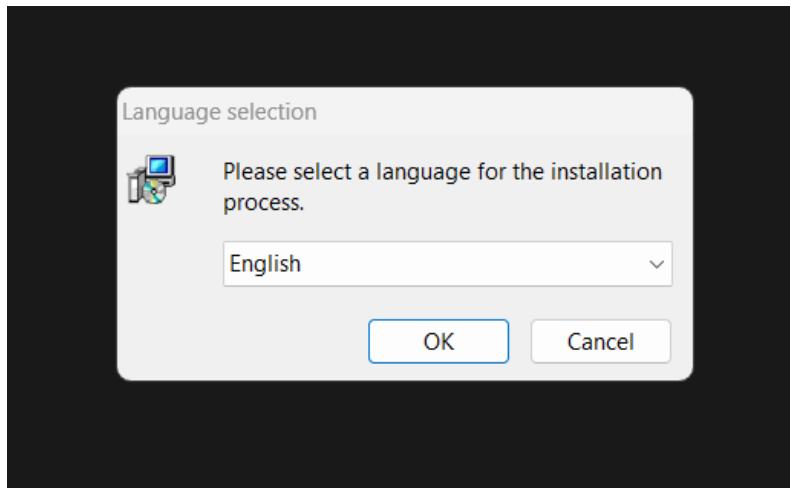


Ilustración 6

Posteriormente de clic a “Next” y escoja la ruta en la que desea instalar el software o si no es necesario vuelva a dar clic a “Next”, entonces la ruta default será una de las comunes en Windows para softwares de arquitectura x86.

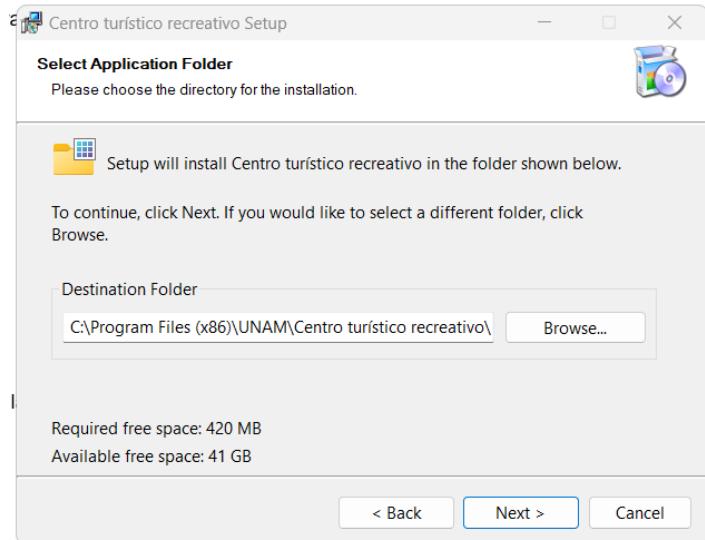


Ilustración 7

Espere a que termine la instalación y de clic en “Finish”.

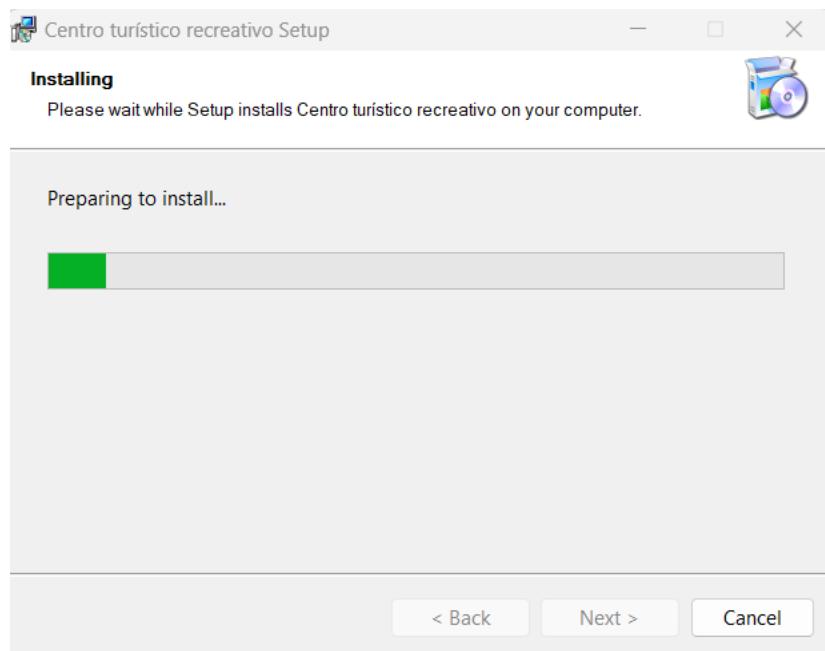


Ilustración 8

El software ya está instalado, diríjase a la ruta de instalación y navegue en las carpetas hasta llegar al directorio “Bin”.

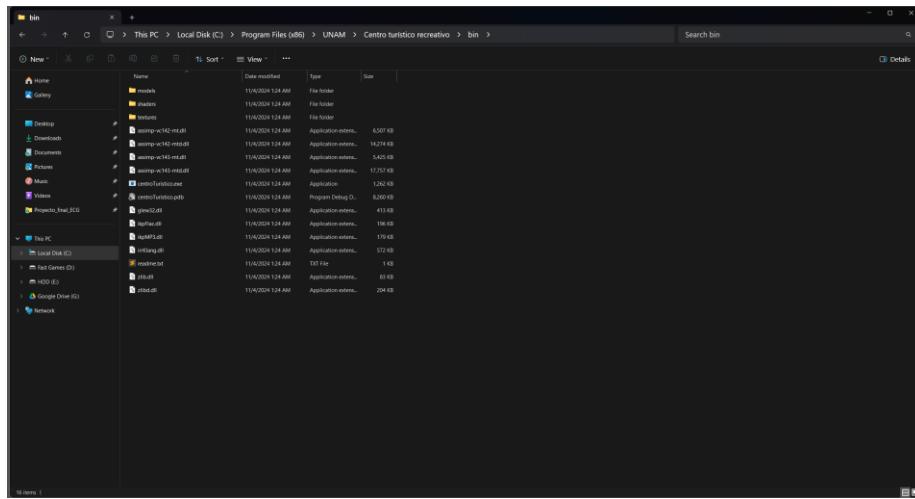


Ilustración 9

Finalmente, dentro del directorio “Bin” de doble clic sobre el archivo ejecutable “centroTuristico.exe”.

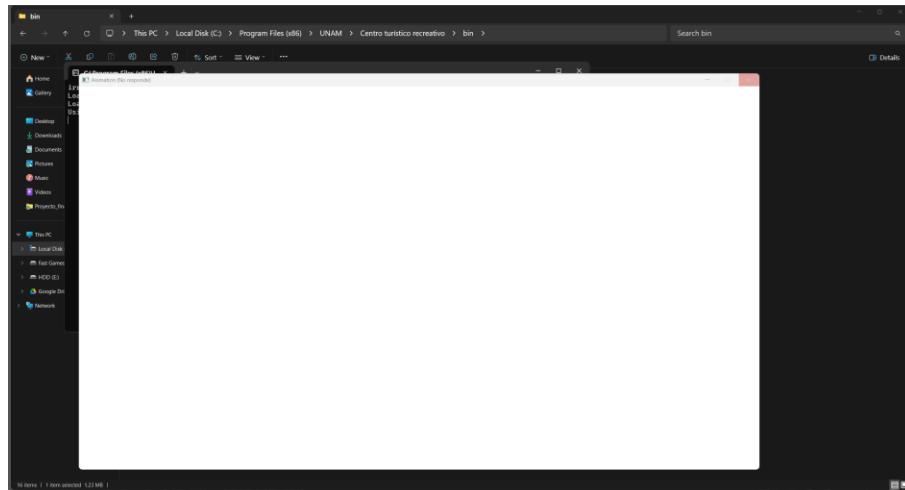


Ilustración 10

Nota: el software no esta correctamente optimizado y en ocasiones le llega a tomar 5 minutos en cargar toda la escena, debe tenerse en cuenta, ya que no esta trabado o es un error, solo toma bastante tiempo en cargar.

Uso del software

Para el cambio de cámara presione F1 o F2, la cámara predeterminada es un cámara en primera persona libre asociada a la tecla F2; mientras que la cámara que se encuentra en F1 es una cámara aérea con proyección ortogonal, la cual permite ver la escena como si fuera un plano.

Cámara predeterminada

La cámara predeterminada comienza en el origen de la escena, tiene una proyección en perspectiva y es una cámara en primera persona que puede moverse por toda la escena, para ello tiene asignadas distintas teclas que consiguen este movimiento, lo primero es saber que el vector “look at” o la dirección en la que mira la cámara se controla con el movimiento del mouse.



Ilustración 11

Para mover la cámara hacia delante, se usa la tecla “W” en el teclado.



Ilustración 12

Para mover la cámara hacia atrás, se usa la tecla “S” en el teclado.



Ilustración 13

Para mover la cámara hacia derecha, se usa la tecla “D” en el teclado.



Ilustración 14

Para mover la cámara hacia izquierda, se usa la tecla “A” en el teclado.

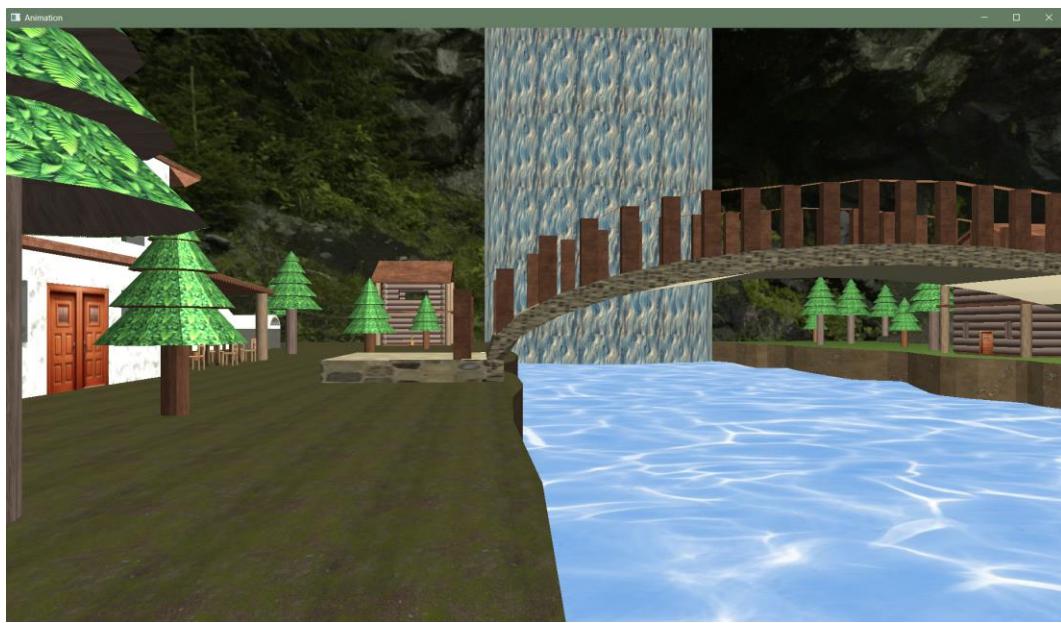


Ilustración 15

Para mover la cámara hacia arriba o en la altura, se usa la tecla o flecha de "UP" en el teclado.



Ilustración 16

Para mover la cámara hacia abajo o en la altura, se usa la tecla o flecha de "DOWN" en el teclado.



Ilustración 17

Cámara aérea

Esta cámara tiene una proyección ortogonal como se mencionó, esta fija a una altura y permite obtener una vista de la escena parecida a la de un mapa, donde puede verse todo lo que hay en ella. Sin embargo, la escena es grande y para explorarla toda se puede mover la cámara como si se moviera en un plano.

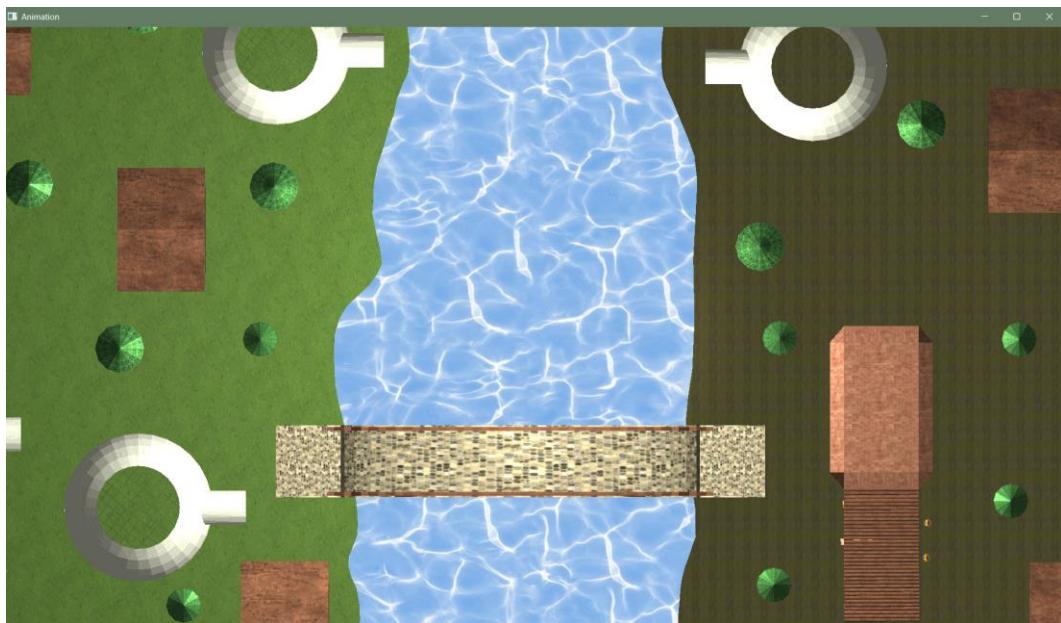


Ilustración 18

La cámara puede moverse hacia delante o en un sentido positivo respecto al eje "Z" con la tecla "W" del teclado.

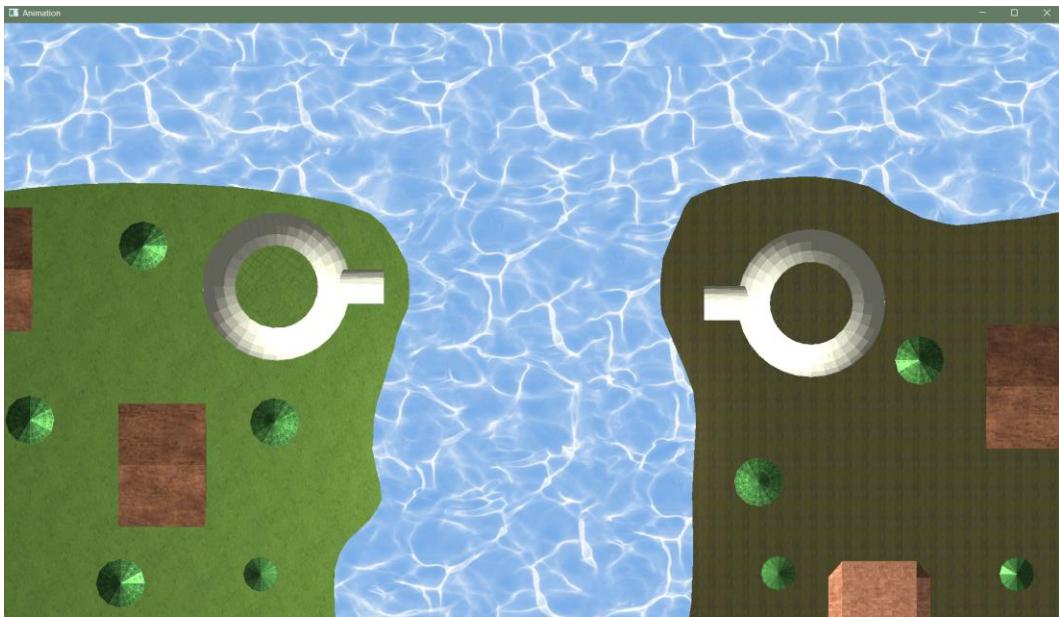


Ilustración 19

La cámara puede moverse hacia atrás o en un sentido negativo respecto al eje "Z" con la tecla "S" del teclado.

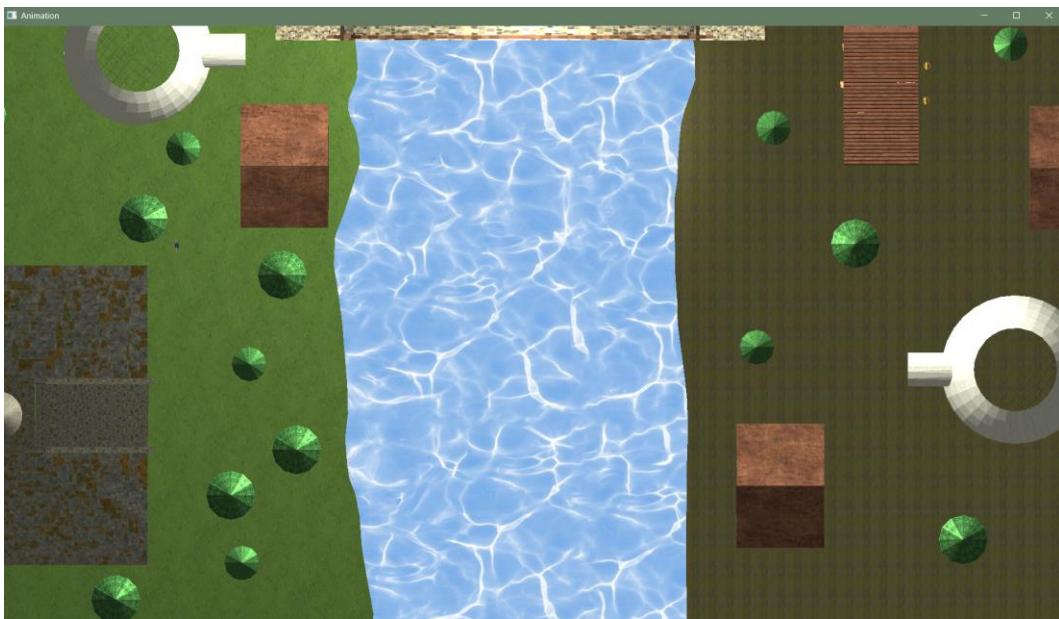


Ilustración 20

La cámara puede moverse hacia la derecha o en un sentido negativo respecto al eje "X" con la tecla "D" del teclado.



Ilustración 21

La cámara puede moverse hacia la derecha o en un sentido positivo respecto al eje “X” con la tecla “A” del teclado.

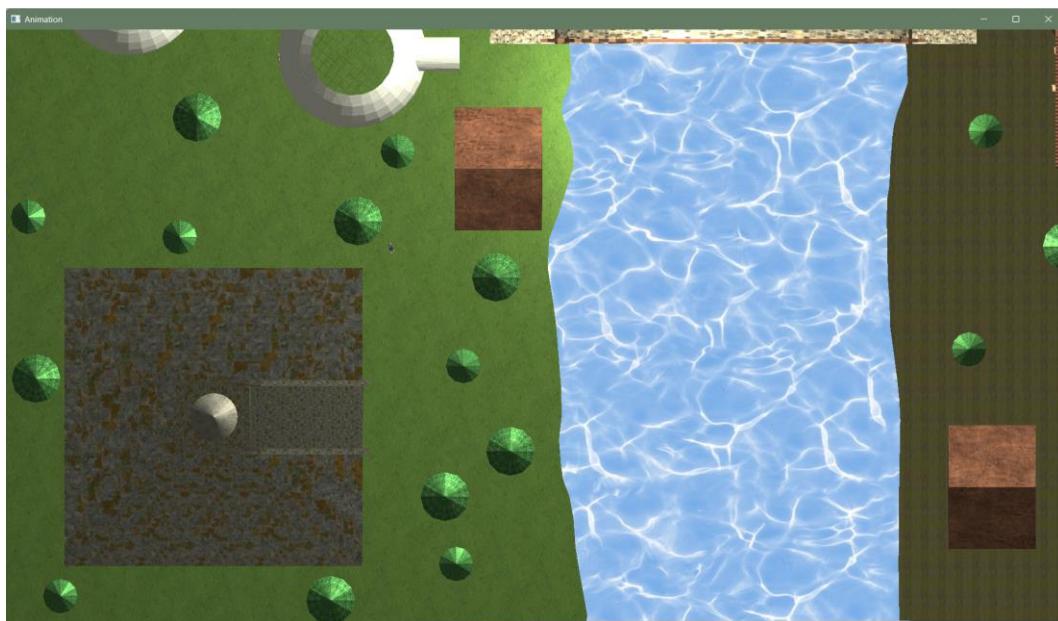


Ilustración 22

Animación por especificación directa

Las puertas pueden abrirse dejando presionada la tecla “Z” del teclado por un corto periodo de tiempo.



Ilustración 23

Las puertas pueden cerrarse dejando presionada la tecla “X” del teclado por un corto periodo de tiempo



Ilustración 24

Animación por KeyFrames programada

Esta animación es controlada mediante la tecla “P” del teclado, para iniciar la animación por primera vez habrá que presionar “P” dos veces seguidas. La primera

pulsación lleva el objeto a la posición donde se realiza la animación, mientras que la segunda es para la ejecución.



Ilustración 25

Después de la primera ejecución de la animación, basta con solo pulsar una vez “P” si la animación se desea repetir. En las imágenes puede observarse el objeto animado y el lugar en la escena donde sucede esta animación por KeyFrames programada en C++ y tomando apoyo de la matriz de transformaciones.



Ilustración 26

2. Cronograma de actividades realizadas

Como se mencionó en la propuesta del proyecto se usó la metodología Agile junto a un Kanban para la gestión y avance del proyecto, por lo tanto, el cronograma de actividades se fue realizando y adecuando mientras iba avanzando el proyecto.

La primera captura mandada es referente a cuando se entregó la propuesta y se comenzó el proyecto.

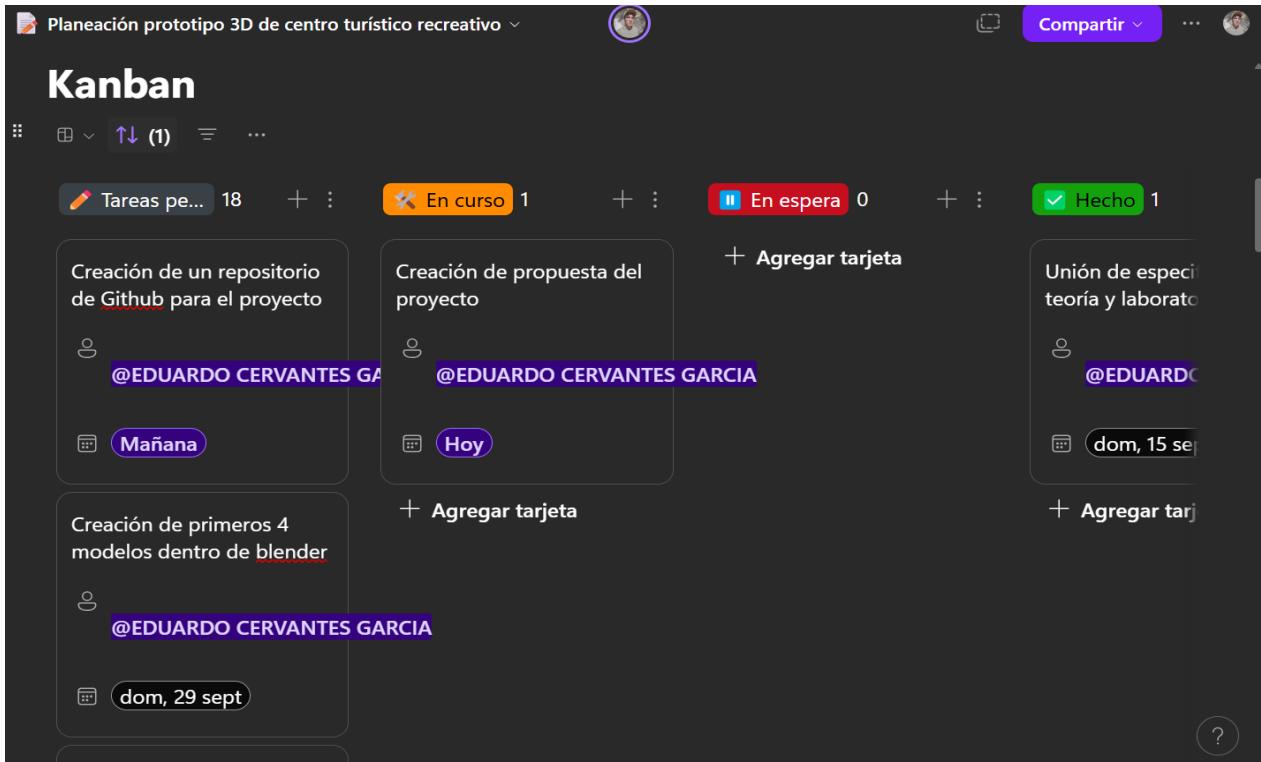


Ilustración 27

La siguiente captura del Kanban, también cronograma, es de cuando se entregó el avance o el proyecto para laboratorio. Puede notarse que algunas tareas no se terminaron y otras se dejaron pendientes para la entrega de teoría, así como que algunas fechas se fueron cambiando conforme lo que realmente se estaba realizando, a pesar de no haber logrado cumplir satisfactoriamente con los tiempos estimados.

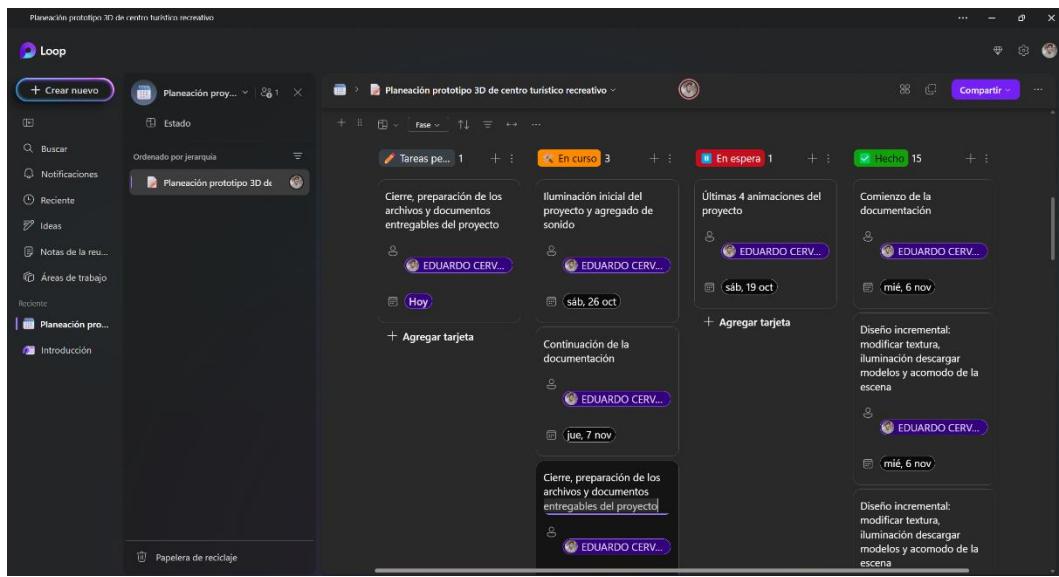


Ilustración 28

Las siguientes imágenes contienen las actividades planeadas ya todas con el estado “Hecho”, junto a la fecha en la que se terminaron, logrando realizar de esta manera una trazabilidad y gestión completa del proyecto. Puede observarse todo el Kanban en el siguiente enlace, accediendo con una cuenta de comunidad UNAM.: [Planeación prototipo 3D de centro turístico recreativo.loop](#)

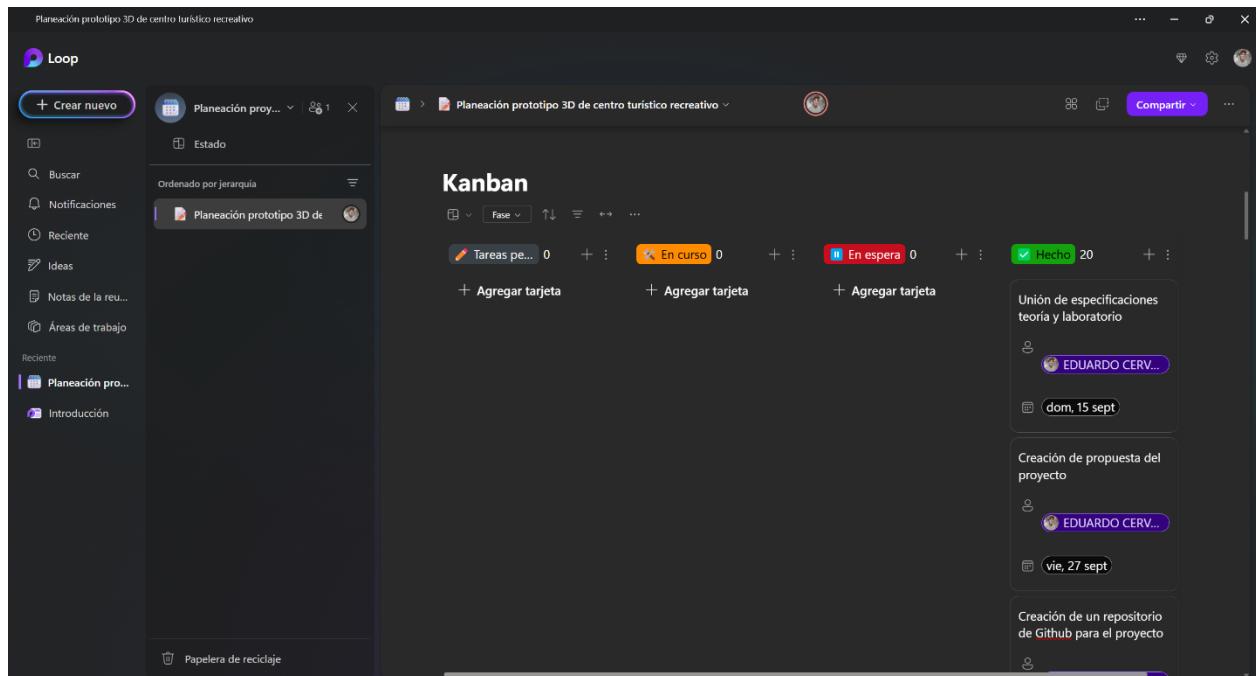


Ilustración 29

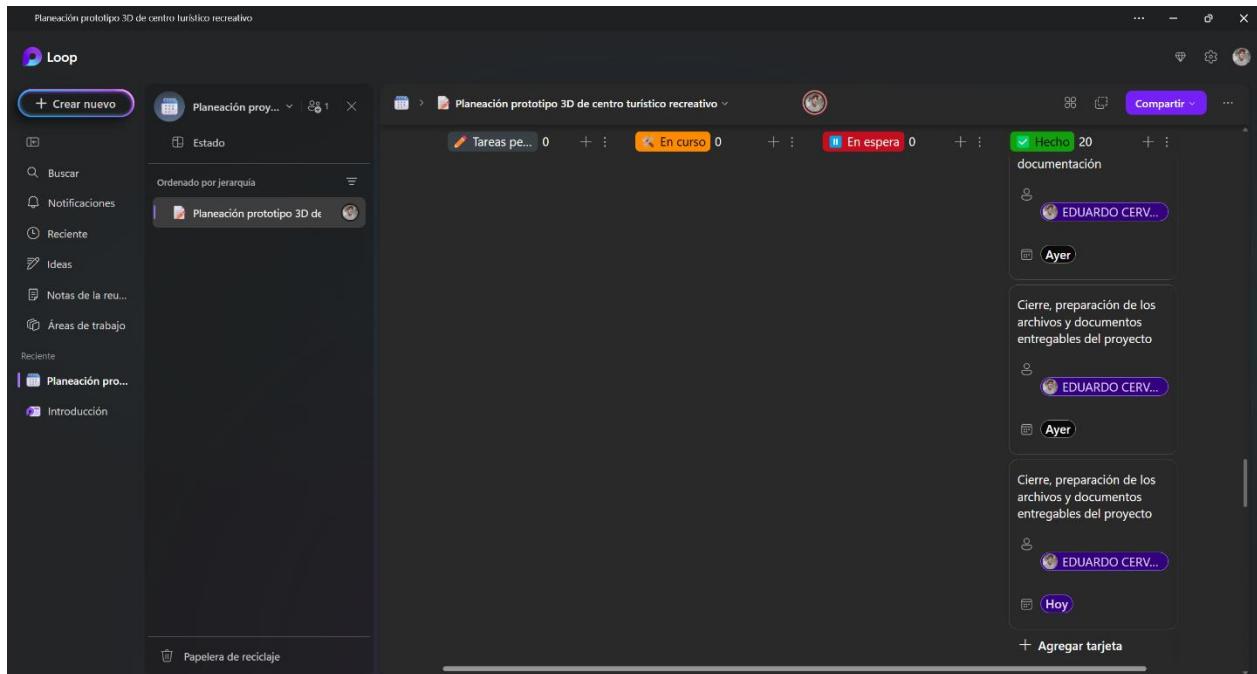


Ilustración 30

Al mismo tiempo, dentro de la herramienta loop, se realizó la gestión para la creación de todos los modelos que se encuentran en la escena. Ya que todos los modelos son propios, excepto algunos personajes animados que se obtuvieron de Mixamo.

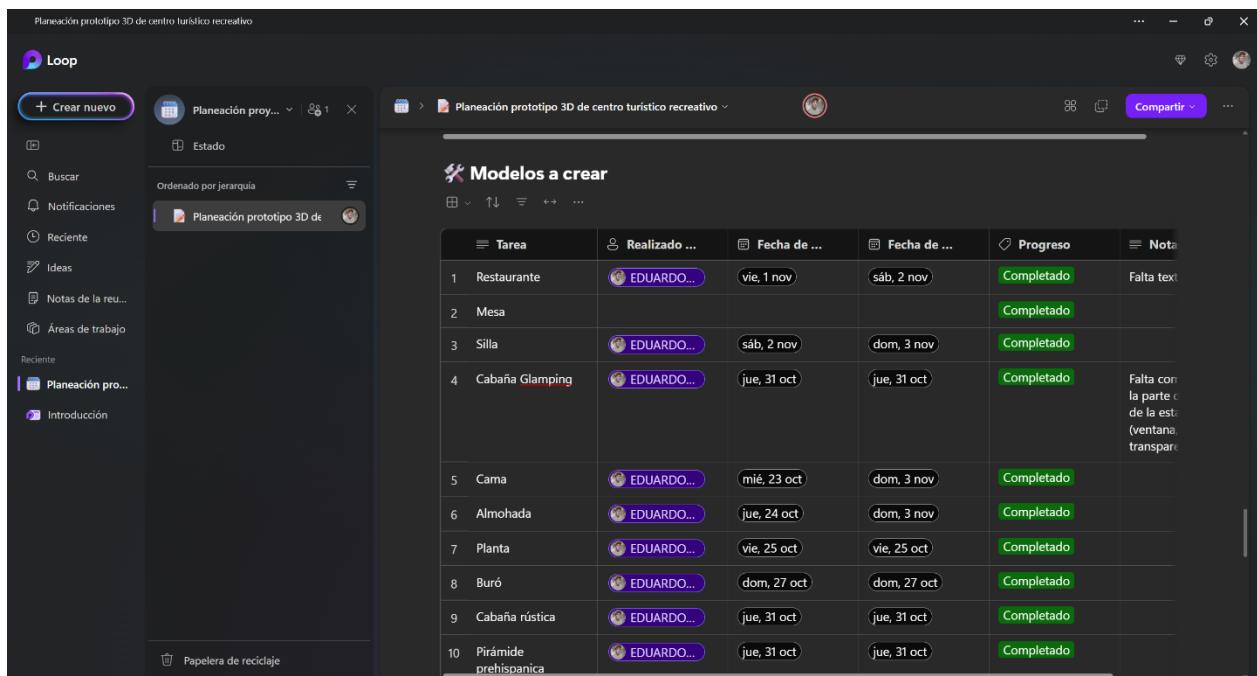


Ilustración 31

Tarea	Realizado ...	Fecha de ...	Fecha de ...	Progreso	Nota
rustica					
12 Puerta rustica	EDUARDO...	sáb, 2 nov	sáb, 2 nov	Completado	
13 Fogata	EDUARDO...	jue, 31 oct	vie, 1 nov	Sin iniciar	
14 Cascada	EDUARDO...	dom, 27 oct	sáb, 2 nov	Completado	
15 Puente	EDUARDO...	jue, 31 oct	jue, 31 oct	Completado	Se cambió el modelo
16 Rio	EDUARDO...	jue, 31 oct	jue, 31 oct	Completado	
17 Árbol esponjoso				Descartado	
18 Árbol pino	EDUARDO...	sáb, 2 nov	sáb, 2 nov	Completado	
19 Estructura cafetería				Descartado	
20 Pez				Descartado	
21 Taza de café	EDUARDO...	sáb, 19 oct	lun, 21 oct	Completado	
22 Barra de la cafetería				Descartado	
23 Condimentos (sal, pimienta, etc)	EDUARDO...	vie, 25 oct		En curso	
24 Servilleteros	EDUARDO...	mié, 23 oct	mié, 23 oct	Completado	

Ilustración 32

3. Herramienta colaborativa de software – repositorio

La herramienta colaborativa además de loop fue Github, el cual más que como herramienta colaborativa se usa como software de control de cambio y repositorio para el proyecto de Visual Studio. Esto se complementó con el uso de Google Drive donde se almacenaron archivos pesados como texturas o los “.blend”.

En las siguientes capturas pueden observarse los commits realizados:

Prueba en grande de la escena y limpieza de archivos del proyecto	Eddspicy committed 2 weeks ago
El escenario base está listo	Eddspicy committed 2 weeks ago
Se mejoró el modelo del terreno y el río-lago, se adaptaron las dimensiones al cubemap en blender lo que requiere menos transformaciones, se usa el shader estático en el terreno	Eddspicy committed 2 weeks ago
Escenario básico con cubemap, terreno del lugar y cuerpo de agua	Eddspicy committed 2 weeks ago
análisis inicial del código de laboratorio usado como template	Eddspicy committed 3 weeks ago
Merge branch 'main' of https://github.com/Eddspicy/Project_CGelHC	Eddspicy committed 3 weeks ago
Inicialización del proyecto en el repositorio	Eddspicy committed 3 weeks ago
Initial commit	Eddspicy authored 3 weeks ago

Ilustración 33

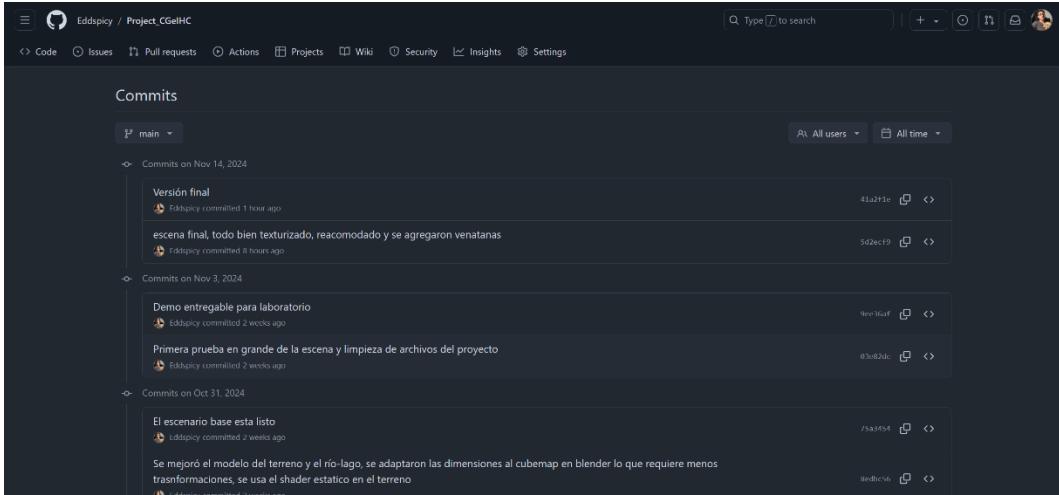


Ilustración 34

El enlace al repositorio en Github es el siguiente:
https://github.com/Eddspicy/Project_CGeIHC y puede ser clonado usando git, desde bash, con este enlace: https://github.com/Eddspicy/Project_CGeIHC.git

4. Desarrollo (descripción de actividades)

Modelado

El modelado se hizo dentro del software de modelado “Blender”, haciendo uso de primitivas geométricas, transformaciones, cambios a primitivas gráficas cuando era necesario, modelado jerárquico y texturizado. Además, se fue aprendiendo con el uso más del software blender, por lo cual más de sus herramientas o simplificaciones pudieron ser aprovechadas, en contraste con haber tenido que modelar mediante primitivas y código en OpenGL.

Muchas de las herramientas o técnicas usadas para modelar se aprendieron en la clase de laboratorio, puede verse el uso de una de esas técnicas en la primera imagen de esta sección. Se tomó el plano realizado en la propuesta, se creó un plano y modificaron sus vértices para adaptarse a la forma requerida o “calcarlo”, finalmente se añadió volumen mediante extrusiones y de esta manera se obtuvo el terreno donde se sostienen los modelos de la escena. Esto implico muchas transformaciones, ya que el terreno determino la escala de la escena y la distribución final de todos los modelos.

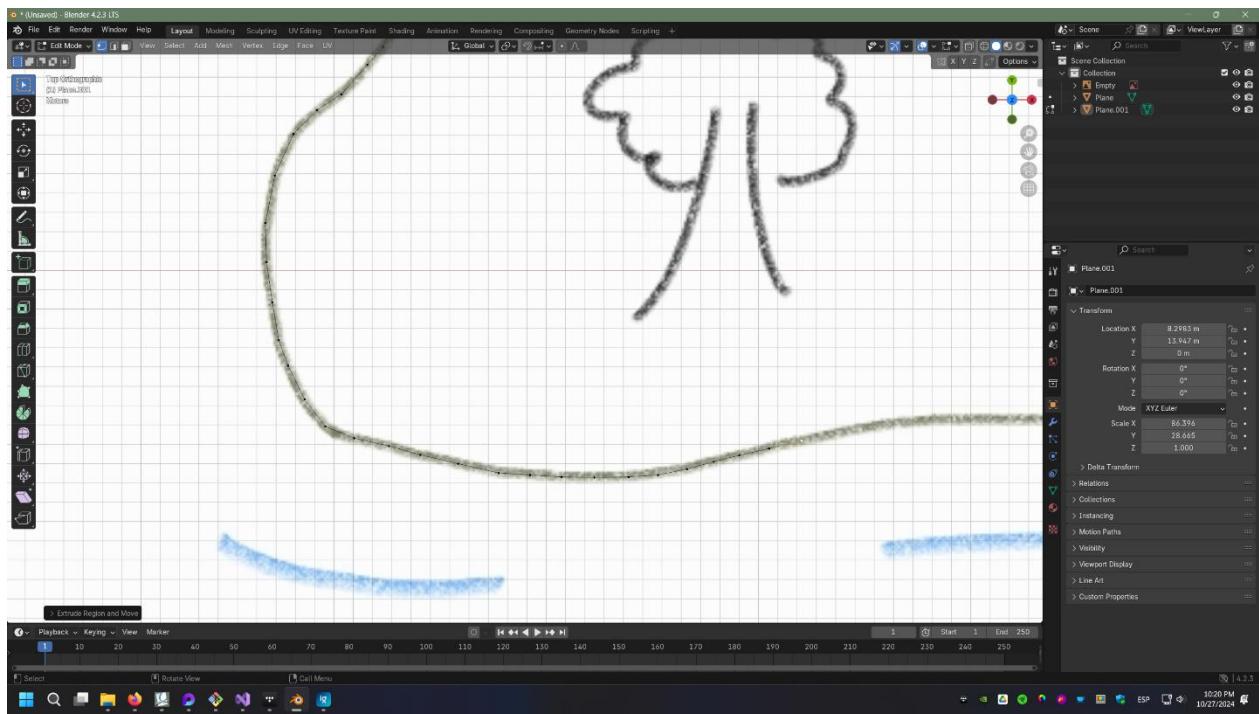


Ilustración 35

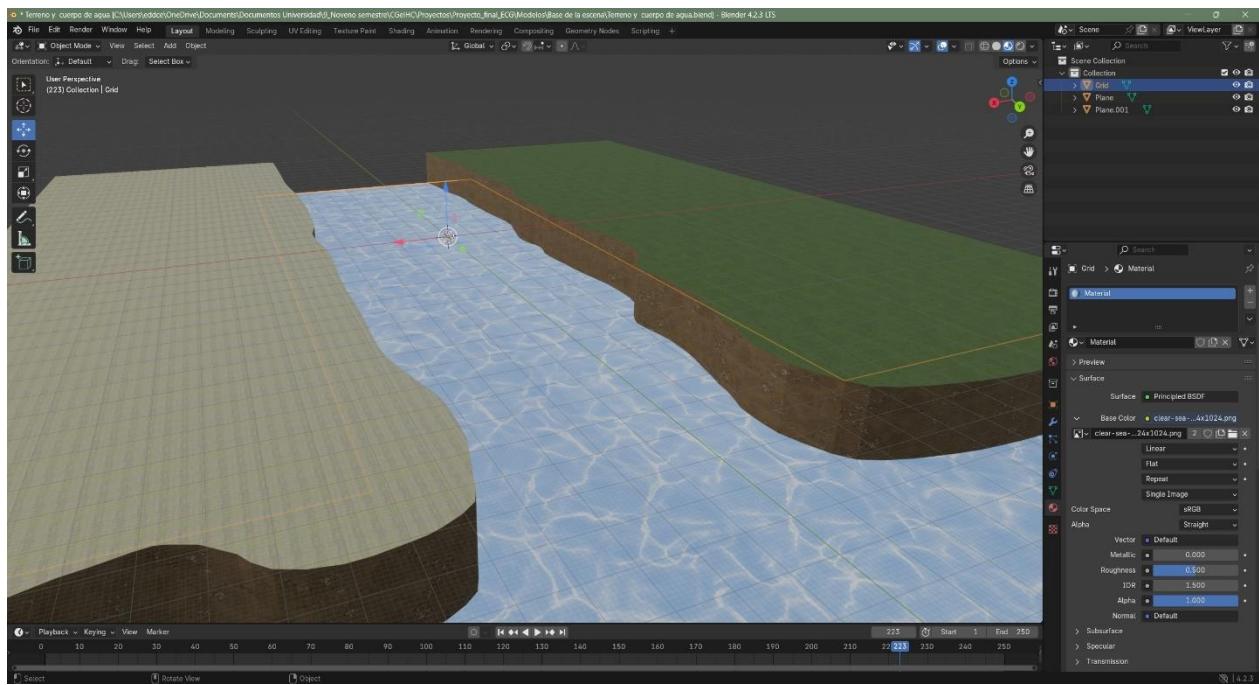


Ilustración 36

En las siguientes imágenes puede observarse como se usó una primitiva geométrico cilindro, la cual se trató como un conjunto de troncos, lo que permitió una construcción más ágil de este tipo de cabaña. Al mismo tiempo, se usaron algunas otras primitivas transformadas como el cubo y un gran uso de modificadores para conformar todas las figuras en un objeto. Los modificadores resultaron una pieza clave durante todo el modelo,

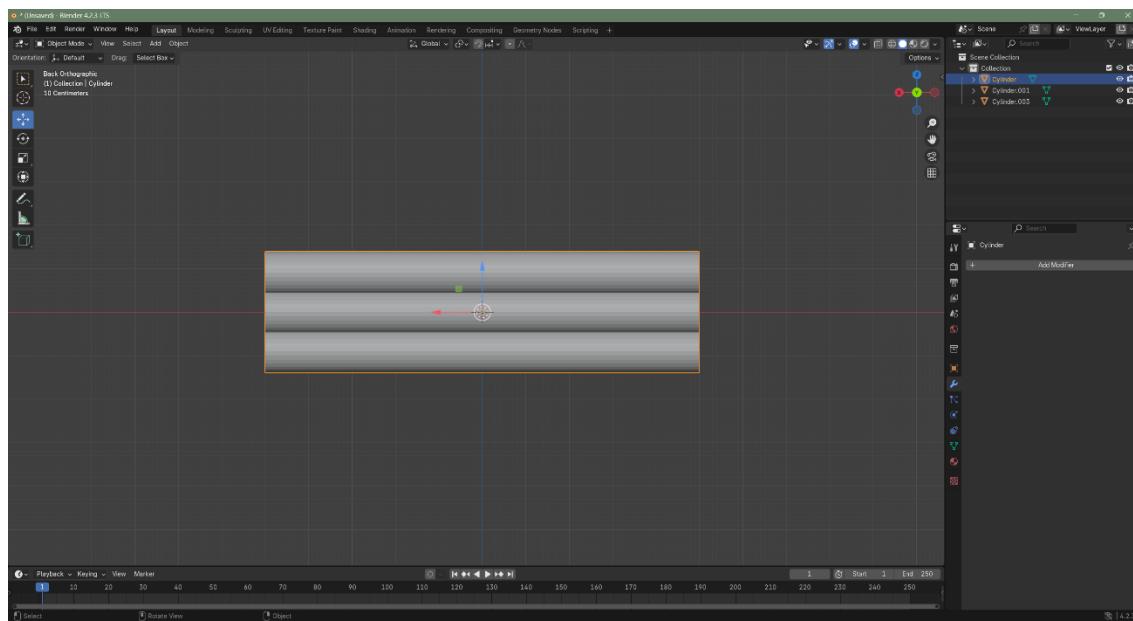


Ilustración 37

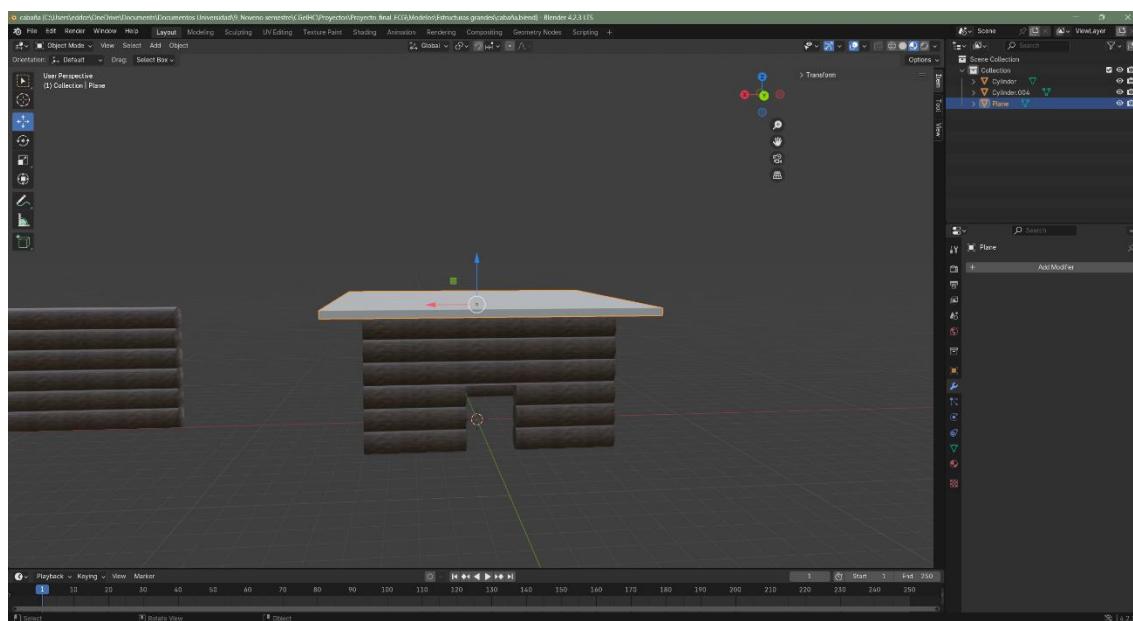


Ilustración 38

La pirámide, por ejemplo, fue construida casi exclusivamente mediante cubos, los cuales se transformaban o ajustaban al tamaño de cada etapa y se les aplicaban modificadores booleanos para que se ejecutaría la unión al modelo o en algunos casos la intersección como en el barandal de la escalinata de la pirámide.

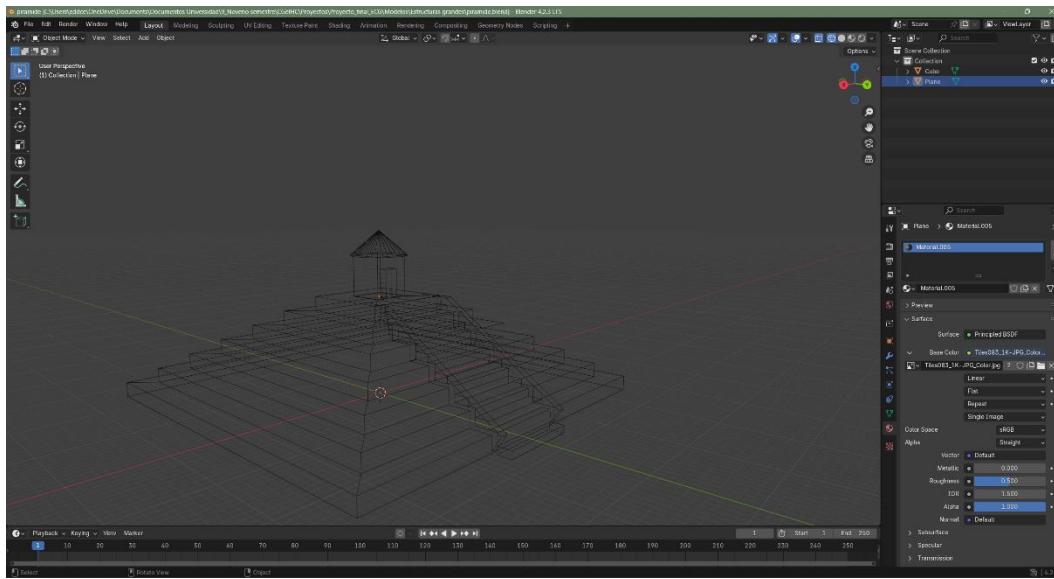


Ilustración 39

En modelos más sencillos o planos, se hizo uso de planos, así como de modificaciones a nivel de vértices y caras para recrear las formas esperadas, junto a modificadores booleanos. Ese fue el caso del restaurante el cual contiene paredes casi totalmente planas, a excepción de sus tejados o relieves.



Ilustración 40

El modelo del puente fue el más complejo de todos ya que, aunque no lo parezca partió de un cilindro, el cual contiene planos mediante modificadores booleanos. Este modelo implico realizar muchas formas vértice por vértice, o modificarlos para obtener lo deseado, es por eso por lo que se consideró el más difícil. Además, se puede decir que este modelo es el culmen de todo, o la mayoría de lo aprendido en blender en laboratorio.

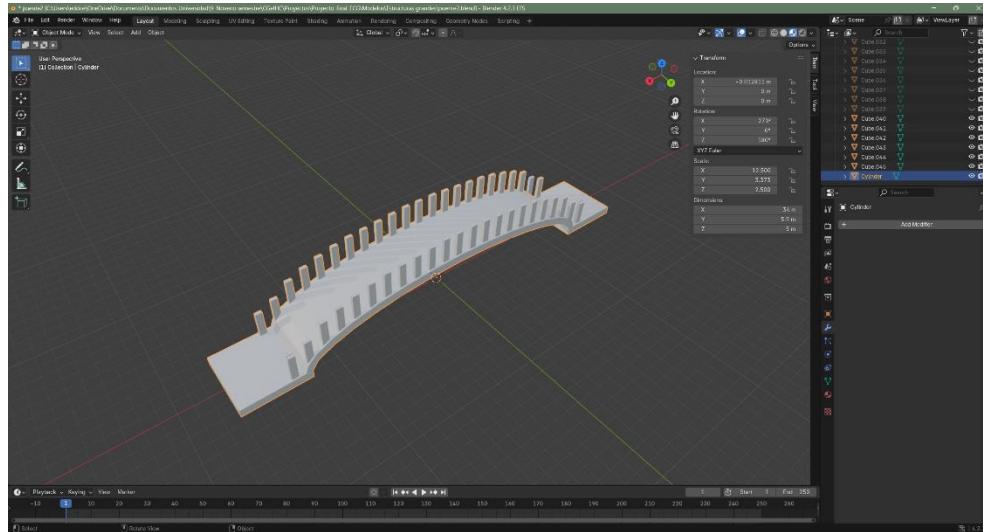


Ilustración 41

La fase de modelado no se limitó a blender, para uno de los personajes animados se utilizó el software de modelado de humanos “MakeHuman”. El cual maneja el modelado a muy alto nivel, permitiendo crear y texturizar personajes muy rápido, los cuales después mediante distintos formatos de intercambio grafico pueden ser llevados a cualquier otro software para su uso o modificación.

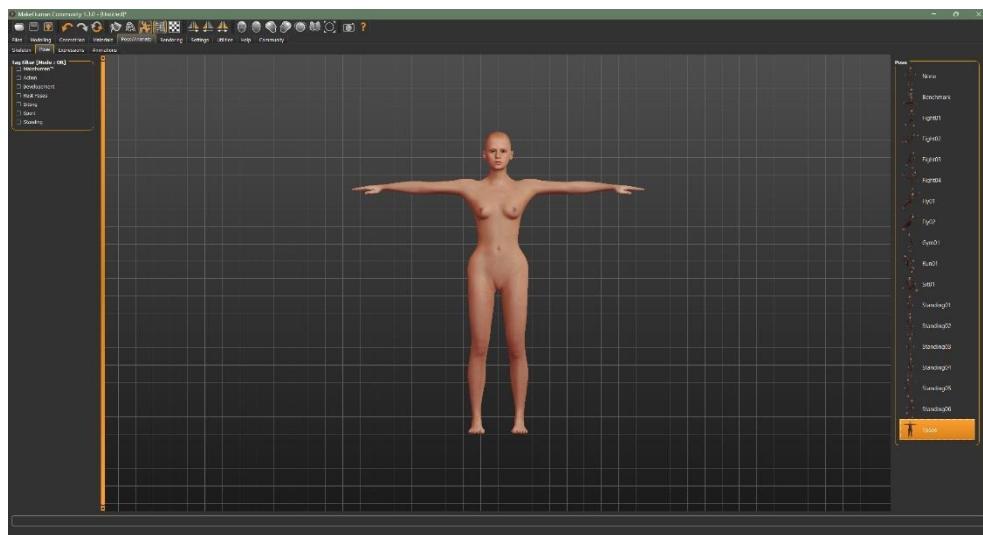


Ilustración 42

En la imagen siguiente puede observarse el resultado final del modelado dentro de MakeHuman, donde a la vez se introdujo un esqueleto o estructura jerárquica al modelo, la cual se usará después en la fase de animación.

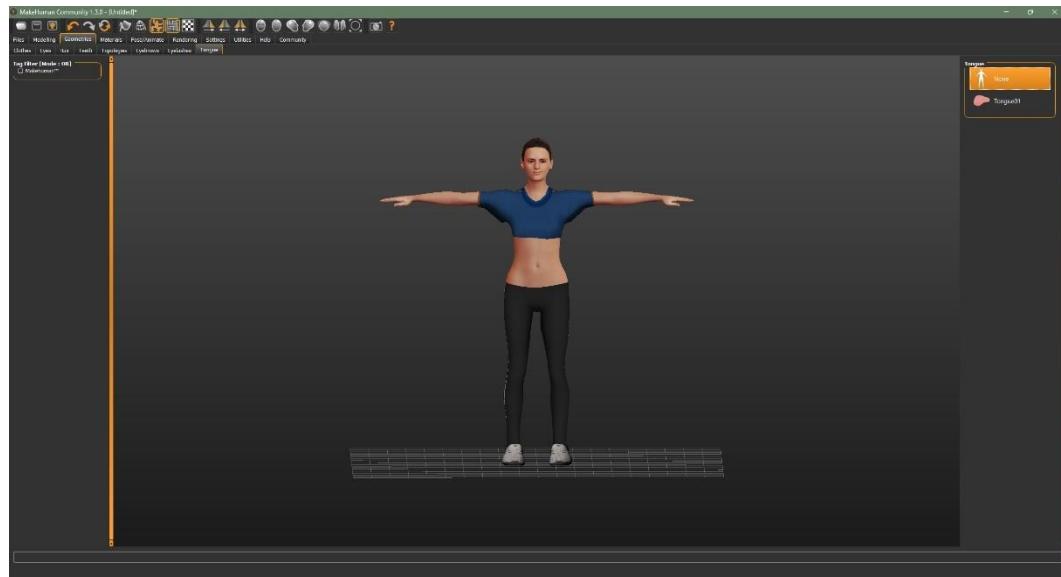


Ilustración 43

Los descrito fueron algunos ejemplos destacados de todos los modelos realizados, pero hubo mucho más trabajo y aprendizaje dentro de todos los modelos que pueden verse en la escena, en la última imagen puede verse parte del proceso de creación de una lámpara o modelo “dummy” para una fuente de iluminación.

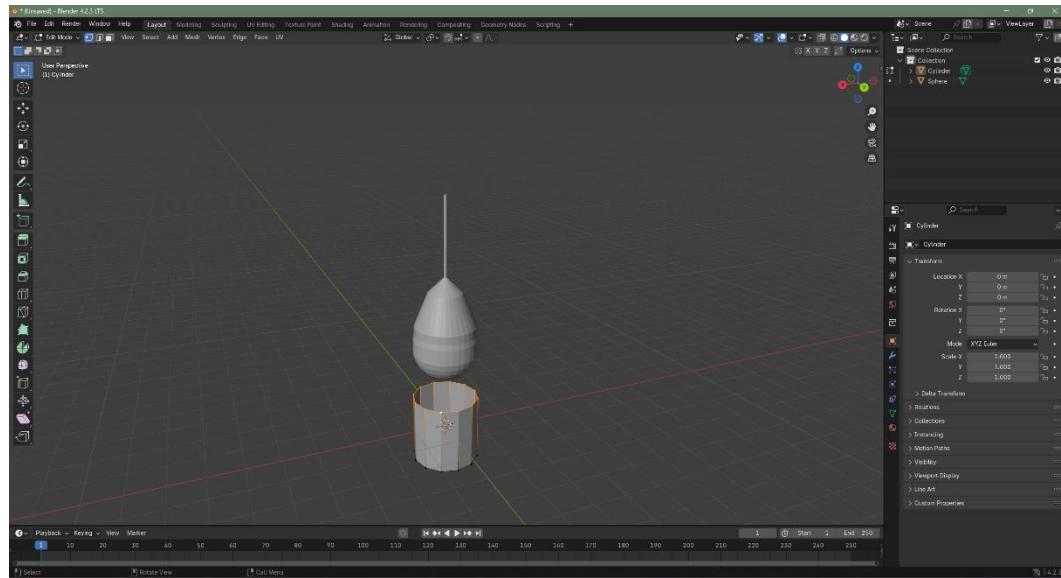


Ilustración 44

Texturizado

Todo el texturizado fue realizado dentro de blender usando “UV maping”, este proceso se realizó en simultaneo a la creación de los modelos en muchos casos, ya que como se mencionó se trabajó con una metodología Agile y no con un ciclo de desarrollo tradicional o con etapas muy marcadas.



Ilustración 45

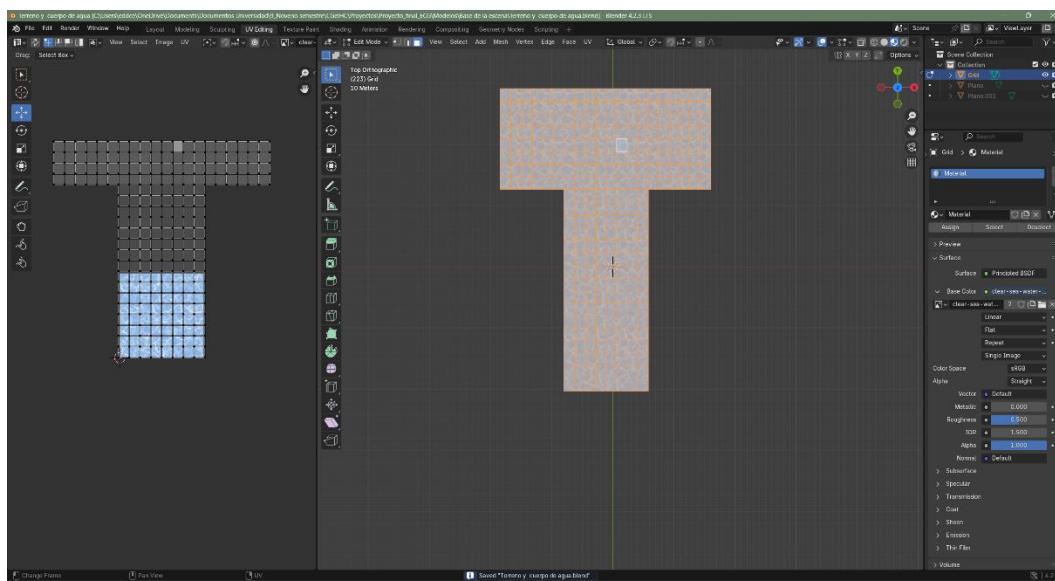


Ilustración 46

Es importante destacar que, aunque la mayoría de texturas se tomaron de páginas de internet que permiten descargarlas gratuitamente, no todas pudieron ser obtenidas de esa forma. Para las texturas más difíciles de encontrar, se utilizó generación de imágenes

por inteligencia artificial y el editor de imágenes GIMP, estas herramientas y el trabajo en ello permitieron que todas las texturas se hicieran realidad, como las ventanas, la textura de la cascada o las hojas de los árboles, etc.



Ilustración 47

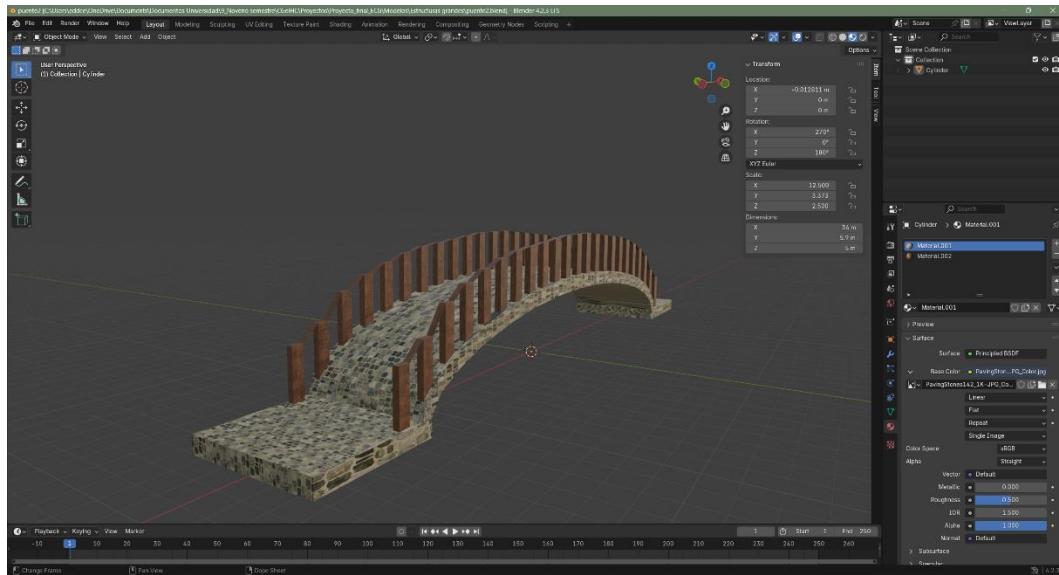


Ilustración 48

Iluminación

En el caso de la iluminación a pesar de tener varias opciones de shaders revisadas en laboratorio, se decidió utilizar el método de iluminación de Phong ya que se pensó como una idea artística que haría lucir más los modelos creados y el uso de materiales.

En las líneas de código presentadas puede observarse la instantiación a la clase del shader de Phong.

```
202 // Compilación y enlace de shaders
203 mLightsShader = new Shader("shaders/11_BasicPhongShader.vs", "shaders/11_BasicPhongShader.fs");
204 proceduralShader = new Shader("shaders/12_ProceduralAnimation.vs", "shaders/12_ProceduralAnimation.fs");
205 wavesShader = new Shader("shaders/13_wavesAnimation.vs", "shaders/13_wavesAnimation.fs");
206 cubemapShader = new Shader("shaders/10_vertex_cubemap.vs", "shaders/10_fragment_cubemap.fs");
207 dynamicShader = new Shader("shaders/10_vertex_skinning-IT.vs", "shaders/10_fragment_skinning-IT.fs");
208 staticShader = new Shader("shaders/10_vertex_simple.vs", "shaders/10_fragment_simple.fs");
209
```

Ilustración 49

También dentro del código y la parte más complicada de la iluminación, fueron las líneas de su configuración, donde se establecen los vectores de color que darán tonalidad a la luz, la potencia, la ubicación, etc. En sí no se sintió difícil esta parte del proyecto, pero llevó muchas pruebas encontrar los parámetros adecuados que permitieran proyectar como se tenían en mente los modelos, ya que las alteraciones a la luz pueden cambiar radicalmente la vista.

```
303 // Configuración para la luz direccional (sol)
304 light.Position = glm::vec3(0.0f, 300.0f, 0.0f);
305 light.Color = glm::vec4(1.0f, 1.0f, 1.0f, 1.0f);
306 light.Power = glm::vec4(60.0f, 60.0f, 60.0f, 1.0f);
307 light.alphaIndex = 10;
308 light.distance = 5.0f;
309
310 // Configuración de lampara
311 artificial.Position = glm::vec3(0.0f, 3.5f, -80.0f);
312 artificial.Color = glm::vec4(1.75f, 1.25f, 1.0f, 1.0f);
313 artificial.Power = glm::vec4(12.0f, 12.0f, 12.0f, 1.0f);
314 artificial.alphaIndex = 10;
315 artificial.distance = 3.0f;
```

Ilustración 50

Como añadido para que el mundo tuviera congruencia a cada fuente se le agregó un modelo que representará la emisión de la luz y permitiera ubicar de dónde venía. Para el mundo se agregó al sol, y en el restaurante se agregó una lampara o foco modelado.

```
572 //DUMMY PARA LA FUENTE DE ILUMINACIÓN PRINCIPAL DE LA ESCENA
573 {
574     staticShader->use();
575
576     staticShader->setMat4("projection", projection);
577     staticShader->setMat4("view", view);
578
579     glm::mat4 model = glm::mat4(1.0f);
580     model = glm::translate(model, light.Position);
581     model = glm::rotate(model, glm::radians(-90.0f), glm::vec3(1.0f, 0.0f, 0.0f));
582     model = glm::scale(model, glm::vec3(1.0f, 1.0f, 1.0f));
583     staticShader->setMat4("model", model);
584
585     sun->Draw(*staticShader);
586 }
587
588 glUseProgram(0);
```

Ilustración 51

En las siguientes imágenes pueden observarse cuales y donde están los modelos o dummies de iluminación para cada fuente de iluminación.

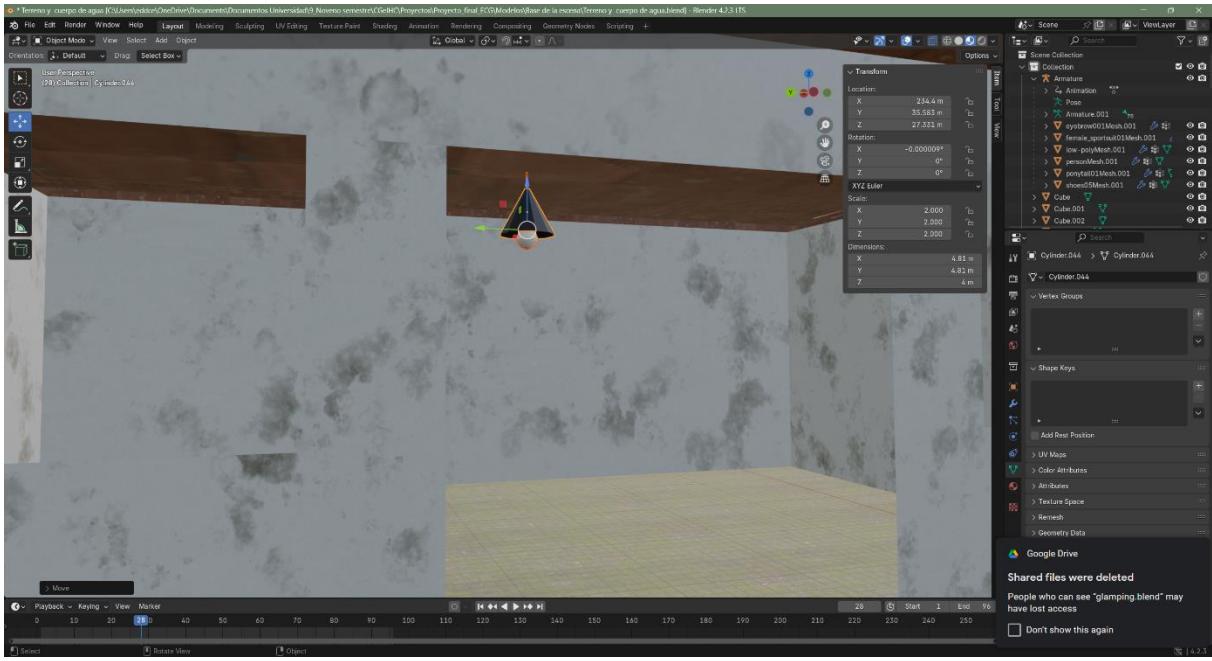


Ilustración 52



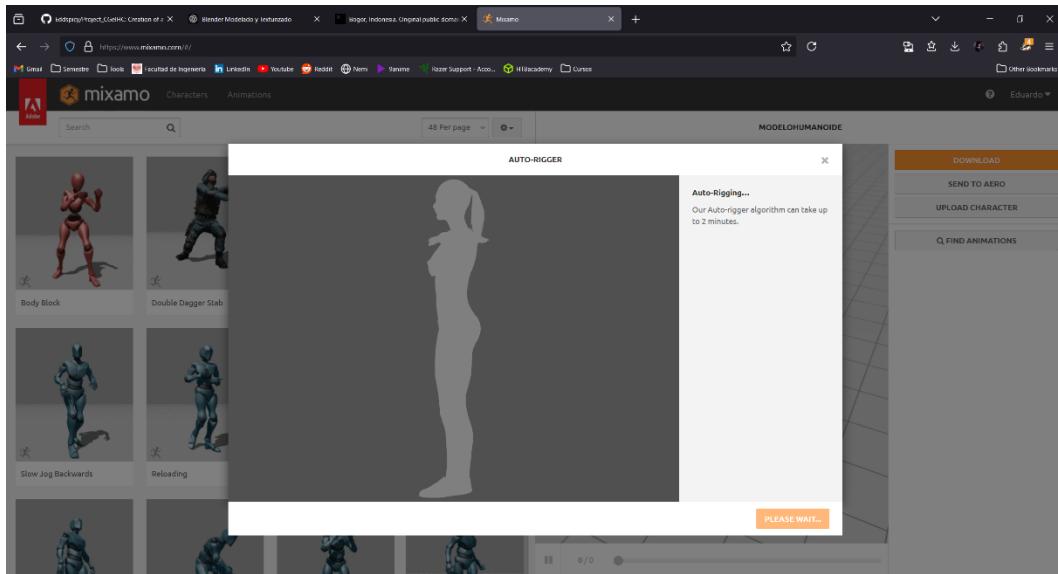
Ilustración 53

Animación

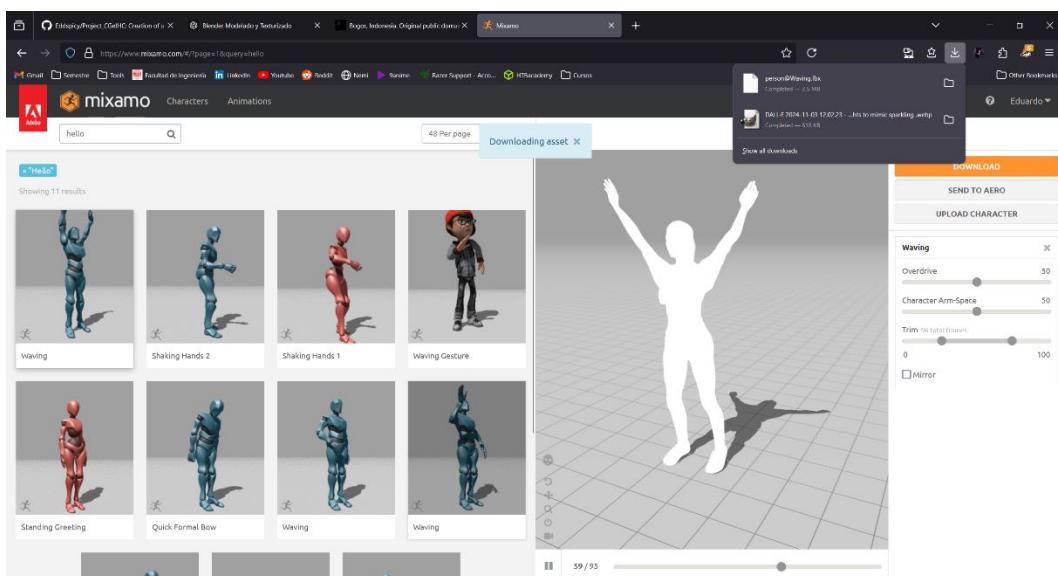
Se realizaron las ocho animaciones solicitadas, seis por keyframes; una de estas animaciones fue programada con el método visto en teoría, para las otras dos

animaciones restantes a las ocho, se usó las técnicas de especificación directa y también animación procedural.

En esta primera imagen del apartado, puede observarse que el modelo humano modelado en “MakeHuman” se llevó a Mixamo, donde se volvió a hacer el proceso de rigging para poder aprovechar en este modelo todas las animaciones por keyframes que la plataforma proporciona de manera sencilla.



Para esto, simplemente se importó el archivo. fbx del modelo a la plataforma, se le aplicó el proceso de rigging a alto nivel que da la plataforma y se seleccionó la animación deseada, permitiendo escoger cualquier animación para un modelo propio.



El modelo ya animado en Mixamo, se llevó también a Blender para ajustar detalles con la normalización de caras del modelo. Un detalle que nunca pudo ser resuelto y sigue sin comprenderse porque sucedió, es un error en la orientación de las caras del modelo, así como tampoco se entendió porque no pudo ser resuelto o no sufría ningún cambio el modelo a pesar de realizarle muchos procesos para intentar repararlo.

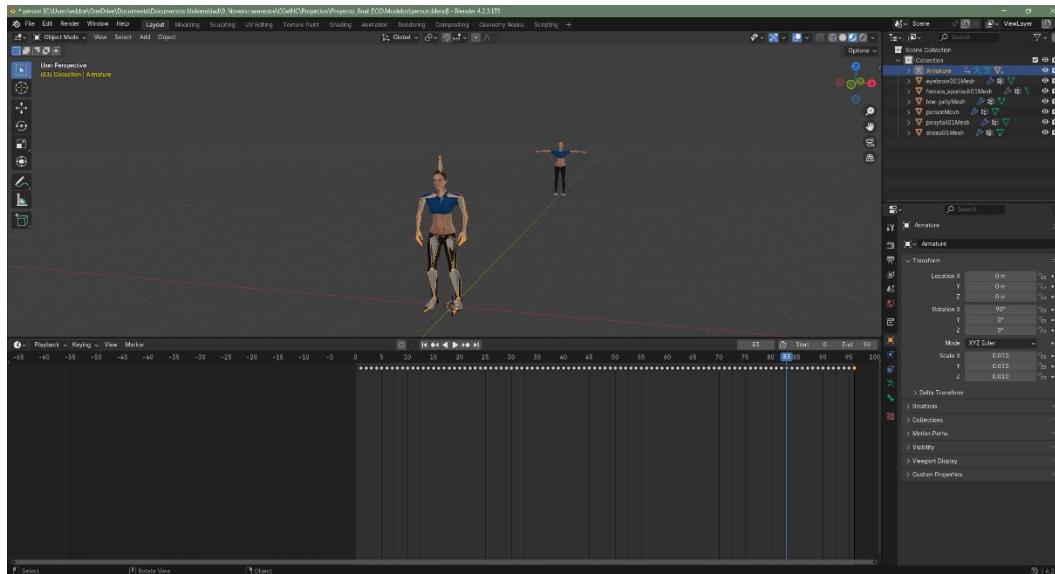


Ilustración 54

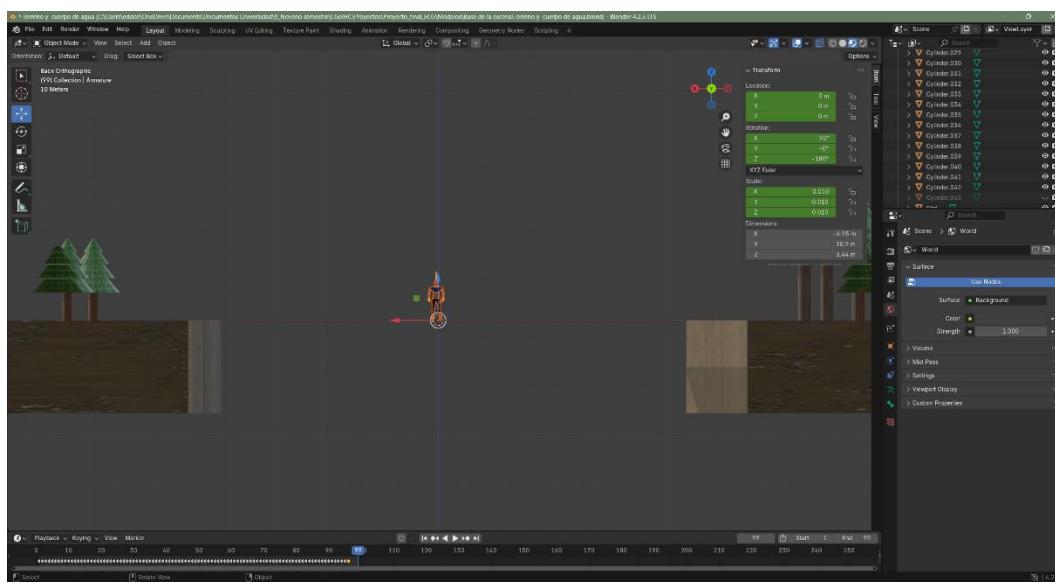


Ilustración 55

En la siguiente imagen puede observarse la animación por KeyFrames realizada a bajo nivel o en código. El proceso fue el mismo al realizado en clase, solo que se adaptó al código que ya se tenía, a los parámetros que necesitaba el modelo; y siguiendo la

recomendación del profesor, luego de definir la animación se codeo cada keyFrame para guardar la animación y que el usuario pueda observar de manera natural este suceso en la escena.

```
character01 = new AnimatedModel("models/empanim.fbx");
character02 = new AnimatedModel("models/personSit.fbx");
character03 = new AnimatedModel("models/personCook.fbx");
character04 = new AnimatedModel("models/personSup.fbx");
character05 = new AnimatedModel("models/personWalking.fbx");

//INICIALIZACION DE KEYFRAMES
//Inicialización de KeyFrames, quitar esto y codear los frames para tener una animación guardada con keyFrames.
KeyFrame[0].movTazaX = -245;
KeyFrame[0].movTazaY= 14;
KeyFrame[0].movTazaZ = -91;
KeyFrame[0].rotTazaY = 0;

KeyFrame[1].movTazaX = -241;
KeyFrame[1].movTazaY = 14;
KeyFrame[1].movTazaZ = -91;
KeyFrame[1].rotTazaY = 0;

KeyFrame[2].movTazaX = -241;
KeyFrame[2].movTazaY = 14;
KeyFrame[2].movTazaZ = -91;
KeyFrame[2].rotTazaY = 0;

KeyFrame[3].movTazaX = -239;
KeyFrame[3].movTazaY = 14;
KeyFrame[3].movTazaZ = -91;
KeyFrame[3].rotTazaY = 0;

KeyFrame[4].movTazaX = -239;
KeyFrame[4].movTazaY = 14;
KeyFrame[4].movTazaZ = -92;
KeyFrame[4].rotTazaY = -6;

KeyFrame[5].movTazaX = -239;
KeyFrame[5].movTazaY = 14;
KeyFrame[5].movTazaZ = -94;
KeyFrame[5].rotTazaY = -15;

KeyFrame[6].movTazaX = -238;
KeyFrame[6].movTazaY = 14;
KeyFrame[6].movTazaZ = -95;
KeyFrame[6].rotTazaY = -15;

KeyFrame[7].movTazaX = -238;
KeyFrame[7].movTazaY = 14;
KeyFrame[7].movTazaZ = -95;
KeyFrame[7].rotTazaY = -20;

KeyFrame[8].movTazaX = -236;
KeyFrame[8].movTazaY = 14;
KeyFrame[8].movTazaZ = -95;
KeyFrame[8].rotTazaY = -45;

KeyFrame[9].movTazaX = -236;
KeyFrame[9].movTazaY = 0;
KeyFrame[9].movTazaZ = -93;
KeyFrame[9].rotTazaY = -180;
```

Ilustración 56

En las siguientes imágenes de código, puede observarse las declaraciones para la animación por especificación directa y su estructura de control. A la vez, se presentan las ecuaciones usadas para el movimiento del agua en la escena, siendo que se usó animación procedural tanto en los vértices de las mallas como en las texturas, brindando un efecto más creíble. Esto último se hizo mediante un shader de vértices y uno de fragmentos, ya que de esa forma es como se realizó este tipo de animación en laboratorio.

```

637 //DIBUJADO DE PUERTAS ANIMADAS POR ESPECIFICACION DIRECTA
638
639     model = glm::mat4(1.0f);
640     model = glm::translate(model, glm::vec3(174.0f, 0.0f, -90.0f + door_rotation)); // translate it down so it's at the center of the scene
641     model = glm::rotate(model, glm::radians(180.0f), glm::vec3(0.0f, 1.0f, 0.0f));
642     model = glm::rotate(model, glm::radians(-90.0f), glm::vec3(1.0f, 0.0f, 0.0f));
643     model = glm::scale(model, glm::vec3(1.8f, 1.3f, 1.3f)); // it's a bit too big for our scene, so scale it down
644     mLightsShader->setMat4("model", model);
645
646     puerta1->Draw(*mLightsShader);
647
648     model = glm::mat4(1.0f);
649     model = glm::translate(model, glm::vec3(173.5f, 0.0f, -90.0f + door_rotation2)); // translate it down so it's at the center of the scene
650     model = glm::rotate(model, glm::radians(180.0f), glm::vec3(0.0f, 1.0f, 0.0f));
651     model = glm::rotate(model, glm::radians(-90.0f), glm::vec3(1.0f, 0.0f, 0.0f));
652     model = glm::scale(model, glm::vec3(1.8f, 1.3f, 1.3f)); // it's a bit too big for our scene, so scale it down
653     mLightsShader->setMat4("model", model);
654
655     puerta2->Draw(*mLightsShader);

```

Ilustración 57

```

842 //Abrir y cerrar puertas restaurante
843 if (glfwGetKey(window, GLFW_KEY_Z) == GLFW_PRESS)
844     if (door_rotation < 10.0f and door_rotation2 > -10.0f) {
845         door_rotation += 1.f;
846         door_rotation2 -= 1.f;
847     }
848     else {
849         door_rotation = door_rotation;
850         door_rotation2 = door_rotation2;
851     }
852 if (glfwGetKey(window, GLFW_KEY_X) == GLFW_PRESS)
853
854     if (door_rotation >= 0.1f and door_rotation2 <= 0.1f) {
855         door_rotation -= 1.f;
856         door_rotation2 += 1.f;
857     }
858     else {
859         door_rotation = door_rotation;
860         door_rotation2 = door_rotation2;
861     }

```

Ilustración 58

```

12_ProceduralAnimation.vs ▾ X
1 #version 330 core
2 layout (location = 0) in vec3 aPos;
3 layout (location = 1) in vec3 aNormal;
4 layout (location = 2) in vec2 aTexCoords;
5 layout (location = 3) in vec3 tangent;
6 layout (location = 4) in vec3 bitangent;
7
8 out vec2 TexCoords;
9
10 uniform mat4 model;
11 uniform mat4 view;
12 uniform mat4 projection;
13
14 uniform float time;
15 uniform float radius;
16 uniform float height;
17
18 void main()
19 {
20
21     vec4 PosL = vec4(aPos, 1.0f);
22     PosL.x += radius * cos(time);
23     PosL.y += radius * sin(time);
24     PosL.z += height;
25
26     gl_Position = projection * view * model * PosL;
27
28     TexCoords = aTexCoords;
29 }

```

Ilustración 59

Sonido

El sonido fue la parte más sencilla respecto a la carga de trabajo, ya que en laboratorio se explicó brevemente la biblioteca IrrKlang. Por lo tanto, en el proyecto se utilizó esta biblioteca como dependencia, se creó un objeto “Sound Engine” o el motor de sonido, al cual se le dirigió o apunto al mp3 que se quería sonando en la escena y esto se colocó en la función main o inicialización de la escena, logrando así música en la reconstrucción 3D realizada.

```
120 // Audio
121 ISoundEngine *SoundEngine = createIrrKlangDevice();
```

Ilustración 60

```
316
317 //Configuración del audio global de la escena al inicializarla
318 SoundEngine->play2D("sound/1-02 Main Theme.mp3", true);
319
```

Ilustración 61

5. Estudio técnico para la venta de la aplicación

Actividad	Descripción	Costo estimado por hora de trabajo (MXN)	Tiempo estimado en horas	Total, en pesos mexicanos
Modelado	Estos profesionales se encargarán de modelar y texturizar al menos 24 modelos, y de adaptar los restantes importados para completar la escena	\$110	35	\$3,850
Iluminación	Estos profesionales se encargarán de toda la iluminación que requiere la escena, ya sea la solar u otras fuentes de iluminación como focos, esto implica también el tratamiento de materiales.	\$100	15	\$1,500
	Estos profesionales son los encargados de hacer las 8 animaciones complejas del			

Animación	proyecto con los modelos necesarios.	\$70	25	\$1,750
Programación	Programadores que desarrollarán código para crear la escena, juntarla, generar interacción, integrar la musicalización, crear las cámaras, implementar botones, etc. También deberán encargarse de actividades como la documentación de lo realizado	\$95	96	\$9,120
			171	\$16,220

Este tipo de actividades entra en el consumo bajo de CFE, por lo cual el Kilowatt/hora tendría el precio de \$0.595, según una página consultada: “un estudio de animación con los equipos necesarios para implementar este proyecto, encendidos durante toda una hora, gasta 1064 watts por hora aproximadamente” (Alejandro, 2011).

$$\text{Precio de la hora de electricidad} = 1064 \approx 1[kW] = \$0.595$$

$$\text{Precio por todas las horas de trabajo} = 171 * \$0.595 = \$101.745 \approx \$150$$

Se dejó el valor de 150 pesos por gastos que no son de equipo como electrodomésticos, luz de las habitaciones, etc.

El precio de un internet de 1000 [mb] simétricos, que se necesita, ya que los archivos gráficos suelen ser pesados es de \$999 al mes, tomado de la compañía Totalplay. A lo que se agregaría también un extra de \$5000 por algún faltante de licencias o equipo de cómputo, ya sea hardware gráfico o licencias de algún software.

$$\text{Costo} = \$16,220 + \$5,000 + \$150 + \$999 = \$22,369$$

El costo dado en la última ecuación es el que tendrá el estudio para poder ejecutar el proyecto con las características mencionadas en este documento. A esto se agregará la ganancia del estudio, la cual se espera de un 30% del costo total para este proyecto. Entonces, el precio final al cliente sería el costo total sumado a la ganancia del estudio.

$$\text{Ganancia} = 22,369 * 0.30 = \$6,711$$

$$\text{Precio final al cliente} = \$22,369 + \$6,711 = \$29,080$$

Se encontró un costó promedio para esta clase de trabajos de \$9,950 (Habitissimo, 2019). Se halló también las siguientes tablas, que pueden ser útiles para tener un precio inicial o base del cual según la solución que se construirá podrá determinarse el precio, la segunda tiene el precio en dólares

Tipos de renders	Costo
Render estático	\$4.800 MXN
Render 360°	\$10.000 MXN
Maquetas render 3d	\$15.000 MXN
Recorrido virtual	\$10.000 MXN

Ilustración 62

Tipo de Servicio de Representación 3D	Costo Inicial del Servicio	Tiempo de Entrega
Planos 3D	\$199 por piso	3 - 4 días
Interiores 3D	\$199 cada uno	3 - 4 días
Exteriores 3D	\$349 cada uno	3 - 4 días
Vistas Aéreas 3D	\$499 cada uno	5 - 6 días
Animaciones 3D	\$2500/por minuto	1 - 2 semanas
Recorridos de Realidad Virtual	\$1/sq ft.	1 - 2 semanas

Ilustración 63

Estas tablas o precios promedio, se obtuvieron de páginas de arquitectura ya que luego de buscar, se determinó que esa clase de trabajos tienen requisitos muy similares al trabajo realizado.

Tomando como base esos datos, se considera que el precio que se había estimado es un precio correcto para el trabajo propuesto, sin embargo, hay detalles como la cafetería, algunas animaciones u elementos que no pudieron ser implementados, lo que implica que el proyecto no podría costar lo mismo.

Se comprobó a su vez, que las horas de trabajo propuestas para obtener lo propuesto, no son nada lejanas a la realidad. Por lo cual, con el resultado final entregado se considera que se tiene que bajar ligeramente el costo por los elementos que no se cumplieron de la propuesta, a pesar de estar completo respecto a la rúbrica y también porque fueron algunas horas menos de trabajo. Se considera un costo final adecuado de \$24,000.



Ilustración 64

6. Conclusiones individuales

Este proyecto ha servido como un ejercicio práctico que ha permitido consolidar y profundizar los conocimientos adquiridos a lo largo del semestre. La ejecución de este trabajo requirió una revisión profunda de conceptos y herramientas, evidenciando la importancia de una comprensión sólida de los fundamentos teóricos vistos durante todo el curso para abordar el proyecto.

La fase de modelado ha representado uno de los mayores desafíos, requiriendo no solo conocimientos de geometría analítica, sino también una sensibilidad estética que no siempre resulta sencilla de desarrollar para todas las personas. A pesar de estas dificultades, se ha logrado construir un conjunto de modelos satisfactorio, o casi todos los modelos, para los objetivos del proyecto.

Por otro lado, la programación y animación han resultado ser las etapas más gratificantes, permitiendo explorar y comprender los mecanismos y todo lo que está detrás de la creación de experiencias virtuales inmersivas que se suelen ver día a día. La implementación exitosa de los algoritmos y técnicas aprendidas ha sido muy satisfactoria.

Este proyecto se destaca de forma personal por lo integrador que es, al reunir una amplia gama de conocimientos adquiridos a lo largo de la carrera, desde metodologías de desarrollo y programación orientada a objetos, hasta gestión de versiones y programación paralela en la parte de dibujado con la tarjeta gráfica. En este sentido, ha servido como una valiosa oportunidad para identificar tanto los logros alcanzados como las áreas que requieren mayor fortalecimiento aún.

En cuanto a la gestión del proyecto, se han evidenciado las dificultades propias e inherentes a la planificación y ejecución de iniciativas de desarrollo de software. Si bien no se cumplieron todos los objetivos iniciales en términos de plazos y alcance, la metodología de desarrollo seleccionada ha demostrado ser efectiva para superar los obstáculos y entregar un producto final que cumple con la rúbrica propuesta por el profesor.

Finalmente, el proyecto ha cumplido con su objetivo principal de consolidar los conocimientos teóricos y prácticos adquiridos. A pesar de los desafíos enfrentados, se ha obtenido una experiencia satisfactoria que ha contribuido al desarrollo y fortalecimiento de competencias, muchas necesarias para abordar proyectos en el campo de la ingeniería de software y más específicamente relacionados a la parte gráfica, por lo cual se considera que se cumplió con los objetivos que este proyecto pudo plantear.

7. Referencias bibliográficas

Alejandro. (2011, 12 junio). *Cuánto pagáis de luz con vuestro estudio?* Hispasonic.

<https://www.hispasonic.com/foros/cuanto-pagais-luz-vuestro-estudio/370983>

CFE. (2024). *Tarifas*. <https://www.cfe.mx/hogar/tarifas/Pages/Acuerdosdetarifasant.aspx>

Cuánto cobrar por un Render / Modelado 3D - [El método 100% efectivo]. (2018, June 14). Yo

Soy Arquitecto. <https://yosoyarquitecto.com/cuanto-cobrar-por-un-render-modelado/>

Glassdoor. (2021, 26 octubre). *Sueldo: Lighting Artist en México en 2024*.

https://www.glassdoor.com.mx/Sueldos/lighting-artist-sueldo-SRCH_KO0,15.htm

Glassdoor. (2024a, agosto 8). *Sueldo: Modelador 3d en México en 2024*.

https://www.glassdoor.com.mx/Sueldos/modelador-3d-sueldo-SRCH_KO0,12.htm

Glassdoor. (2024b, septiembre 17). *Sueldo: Animador 3d en México en 2024*.

https://www.glassdoor.com.mx/Sueldos/animador-3d-sueldo-SRCH_KO0,11.htm

Glassdoor. (2024c, septiembre 25). *Sueldo: Programador en México en 2024*.

https://www.glassdoor.com.mx/Sueldos/programador-sueldo-SRCH_KO0,11.htm

=1000&federated_search_id=3f1c162b-0419-4c34-9d9f-a6bcfee5492f

Habitissimo. (2019, December 11). *renders 3d: Precio y Presupuestos*. Habitissimo.com.mx.

<https://www.habitissimo.com.mx/presupuesto/renders-3d>

Smith, A. (2023, January 3). *3D Rendering Services: #1 Architectural Visualization Company*.

3D Rendering Services: #1 Architectural Visualization Company.

<https://render3dquick.com/blog/cuanto-cuesta-el-renderizado-3d>

Modelos (4 modelos humanoids de animaciones por KeyFrames)

Adobe. (2019). *Mixamo*. Mixamo. <https://www.mixamo.com/#/>

Texturas

ambientCG. (n.d.). ambientCG - Free Public Domain PBR Materials. Ambientcg.com.

<https://ambientcg.com/>

[Texturas generadas por IA]. (27/10/2024 a 14/11/2024). [Imágenes png con resolución 1024x1024

pedidas mediante prompts]. Imagen generada por Eduardo Cervantes mediante ChatGPT.

Sonido

Nintendo. (2017). The Legend of Zelda: Breath of the Wild (Original Soundtrack) [Audio recording].

Nintendo.