



Universidad Nacional Autónoma de  
México



Facultad de Ingeniería

Laboratorio de Computación Gráfica e  
Interacción Humano Computadora

Semestre 2021-2

Profesor: Dr. Sergio Teodoro Vite

## **Reporte del proyecto final: Thousand Sunny**

Fecha de entrega: 26 de junio de 2021

Fuentes Huizar Carlos Apolo  
García Delgado Ian Ricardo  
Lázaro Martínez Abraham Josué

# Índice

<b>Resumen</b>	<b>4</b>
<b>Introducción</b>	<b>4</b>
<b>Materiales y métodos</b>	<b>6</b>
Materiales	6
Metodología	6
Conceptualización	6
Modelado	7
Aplicación de Texturas y colores	7
Iluminación	8
Animación	9
Renderizado	9
<b>Experimentos</b>	<b>11</b>
<b>Resultados</b>	<b>14</b>
<b>Link Video demostrativo</b>	<b>16</b>
<b>Conclusiones</b>	<b>16</b>
<b>Referencias</b>	<b>17</b>
<b>Bibliografía complementaria</b>	<b>17</b>

## Tabla de rúbricas

Rúbrica de evaluación del reporte	Porcentaje
El alumno presenta una portada clara y formal, con referencias institucionales, título visible, fecha de entrega y lista de autores con información de contacto.	<b>5</b>
El alumno presenta un resumen, que describe de manera muy concreta el tema del proyecto. El resumen incluye palabras clave que redirigen al lector a ubicar el campo de estudio.	<b>5</b>
El alumno realizó una investigación formal del estado del arte de su proyecto y lo presenta bien fundamentado en la introducción, haciendo referencia a las fuentes de información usando las normas APA. Presenta un análisis económico, social y ético del tema de su proyecto.	<b>10</b>
El alumno presenta la metodología que empleó para el desarrollo de la práctica, presentando algoritmos, diagramas de flujo, diagramas de arquitecturas del programa, así como requerimientos de hardware y software. Presenta segmentos de código relevantes, figuras ilustrativas debidamente tituladas y referenciadas en el texto.	<b>25</b>
El alumno presenta experimentos concretos efectuados, exitosos o fallidos, mediante figuras y capturas de pantalla debidamente tituladas y referenciadas en el cuerpo del reporte.	<b>25</b>
El alumno presenta los resultados finales de los experimentos (versión final) mediante figuras y capturas de pantalla debidamente tituladas y referenciadas en el cuerpo del reporte. Analiza las ventajas y desventajas de la metodología aplicada. El alumno realiza una discusión de los resultados obtenidos comparado con otros desarrollos existentes, reconociendo sus contribuciones y limitaciones.	<b>10</b>
El alumno es capaz de expresar sus conclusiones del trabajo realizado de manera concreta y coherente, resumiendo en breves palabras cómo impactó la metodología empleada en los resultados finales y posibles mejoras a futuro.	<b>10</b>
El alumno presenta las referencias bibliográficas en formato APA y las cita correctamente en el cuerpo del reporte. Si sugiere otras fuentes no citadas, las coloca en una sección llamada “bibliografía complementaria”.	<b>5</b>
El alumno incluye un enlace a un video demostrativo sobre el proyecto final	<b>5</b>
Total	<b>100</b>

## Resumen

El presente proyecto tiene como propósito aplicar todos los conceptos vistos durante el curso del laboratorio de computación gráfica e interacción humano- computadora, de forma que al seguir los pasos establecidos principalmente por el pipeline gráfico y el pipeline de renderizado se tenga como resultado final una escena gráfica con elementos animados, texturas y con modelos de iluminación implementados que diferencien objetos metálicos y de vidrio, de otros con aspecto opaco y de tipo plástico. En especial durante la etapa de conceptualización se buscó realizar una aproximación al barco de One Piece denominado “Thousand Sunny”, tratando de apegarnos lo más posible al diseño original pero añadiendo algunos aspectos que lo diferencien para crear algo.

Para lograr esto se hizo uso de las funciones proporcionadas por la API OpenGL, las cuales nos permitieron crear el código para cargar nuestros modelos a la escena (ya sea los que creamos nosotros o los de personajes que se obtuvieron de la plataforma sketchfab), implementar animaciones obtenidas de la plataforma Mixamo y aplicar los modelos de iluminación de Phong y Fresnel. Además de la posibilidad de usar shaders de vértice y shaders de fragmento desde donde se realizan los cálculos pertinentes para visualizar nuestras geometrías.

## Introducción

La computación gráfica nos permite representar cualquier tipo de objeto, ya sea imaginario o real en un dispositivo de despliegue gráfico como puede ser la pantalla de una computadora. En la actualidad incluso, se ha llegado a tener un grado de calidad tan alto en las escenas construidas que es sorprendente el realismo que se puede alcanzar. Para poder lograr estos resultados se debe seguir una serie de pasos los cuales se encuentran establecidos en el denominado pipeline de renderizado, el cual podemos observar en la figura 1.

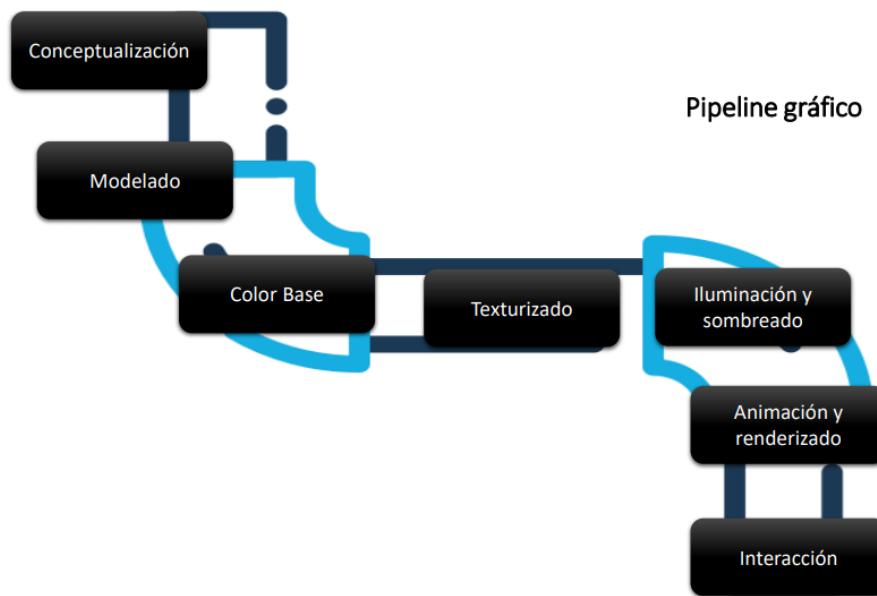


Figura 1. Pipeline Gráfico

Este nos guía desde la conceptualización donde debemos definir qué es lo que queremos hacer y cómo es que queremos que se vean representadas todas nuestras ideas, para de esta forma comenzar a modelar cada uno de los objetos que pertenecen a la escena con algún software de modelado como

lo es blender; donde además podemos añadir características como su color base o el texturizado con el que contarán los objetos. Posteriormente con las etapas de iluminación y sombreado podremos implementar los modelos de iluminación como el de Phong o el de Fresnel, donde aplicaremos las características inherentes de objetos metálicos, cristalinos, plásticos, opacos, entre otros. Finalmente en las etapas de animación podremos darle movimiento a los modelos para que de esta forma se añada dinamismo a nuestro escenario. También el usuario podrá interactuar con ellos y de forma indirecta aplicará transformaciones de rotación, traslación o escalamiento que a su vez nos dará las herramientas para crear aplicaciones de cualquier tipo como las de entretenimiento en donde se encuentran los videojuegos o de tipo académico donde entran aplicaciones como el crear modelos del cuerpo humano para que los alumnos puedan comprender de mejor manera su funcionamiento.

Otro de los procesos importantes en el cómputo gráfico es el pipeline de renderizado. Este define la manera en que el modelo pasará de ser un conjunto de vértices que definen un modelo 2D o 3D con coordenadas locales a una representación 2D en la pantalla de nuestro dispositivo. Podemos observar un esquema de este pipeline en la figura 2.

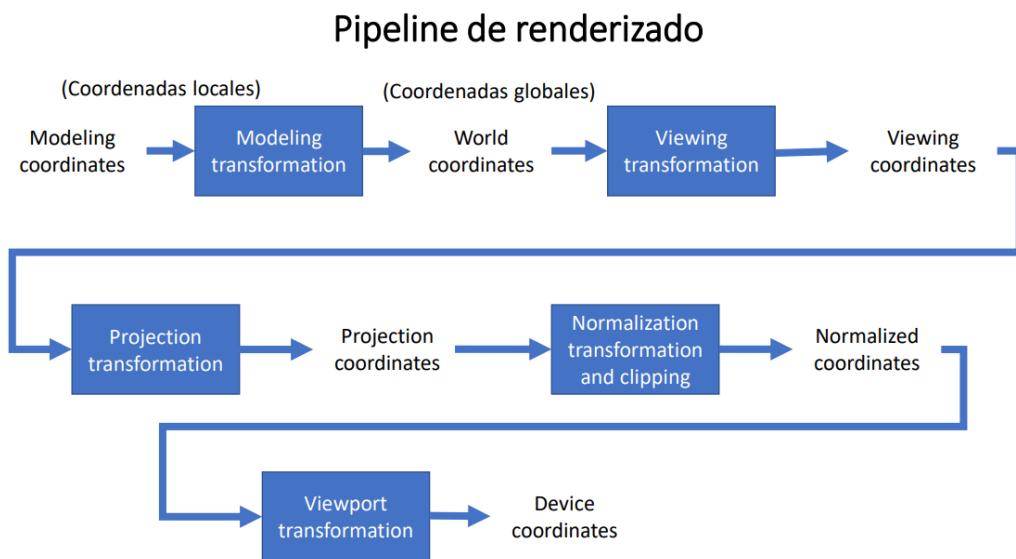


Figura 2. Pipeline de renderizado

Estos pasos se involucran en gran medida con el código que se realiza usando la API OpenGL. Este es una interfaz de programación de aplicaciones diseñado para representación 2D y 3D gráficos. Proporciona un conjunto común de comandos que se pueden usar para administrar gráficos en diferentes aplicaciones y en múltiples plataformas. (TechLib, s.f.)

Usando OpenGL y en especial shaders, como los shaders de fragmento y los shaders de vértice, podremos definir las operaciones entre matrices que contienen información de nuestro modelo. Utilizado en los gráficos por ordenador en 3D, un shader es un pequeño programa o conjunto de algoritmos que determinan cómo se renderizan las propiedades de la superficie de los objetos en 3D y cómo interactúa la luz con el objeto dentro de un programa de ordenador en 3D. Las nuevas GPUs calculan los shaders cuando antes era un algoritmo calculado por la CPU. Estas instrucciones dan el aspecto final a un objeto a partir de transformaciones, iluminación, color, efectos especiales, etcétera. (Beal, s.f.)

En particular en el shader de vértice se encontrarán las operaciones entre las matrices que contienen las coordenadas de los vértices, las matrices que nos indican las transformaciones que se harán a

cada vértice, la matriz que nos define en qué posición está la cámara, así como la matriz de proyección que nos dirá cómo se verán los objetos de la escena. De igual forma contendrá aquellas operaciones aplicadas a los huesos de los modelos que cuentan con animación. Por otro lado en el shader de fragmentos se hará cargo de las operaciones relacionadas a lo visual de nuestra escena, es decir, donde se definirán las luces y cómo interactúan con los objetos.

## Materiales y métodos

### Materiales

En cuanto al software utilizado, en las primeras etapas se hizo uso de Blender para realizar nuestros modelos, animarlos, ponerles texturas y algún color base. Cabe destacar que nos auxiliamos de la plataforma Mixamo para poder darle una estructura jerárquica a los modelos, algunos de los cuales se importaron de la plataforma sketchfab y que pudieran realizar algún tipo de animación. Posteriormente se usó la API OpenGL en la versión 3.3, utilizando los lenguajes C/C++ y GLSL para realizar todo el código como los shaders, las clases, la carga de modelos, el dibujado de los modelos, la animación y para programar las teclas de modo que el usuario pueda interactuar con los modelos. El IDE utilizado fue Microsoft Visual Studio 2019 y en algunas ocasiones Sublime Text. Fue necesario agregar librerías como GLFW, GLM, GLAD y también ASSIMP. Esta última de gran importancia para nuestro proyecto, ya que es una biblioteca que nos servirá para cargar modelos o escenas 3D almacenados en gran variedad de formatos, como: Collada (\*.dae; \*.xml), Blender (\*.blend), 3D Studio Max 3DS (\*.3ds), Wavefront Object (\*.obj ), y muchos más. Assimp puede cargar información de vértices, coordenadas de textura, normales, materiales, animación, y otros.

Para el hardware se utilizó una computadora que cuenta con las siguientes características:

- ❖ **Marca:** HP
- ❖ **Serie:** OMEN
- ❖ **Procesador:** Intel Core i7-8750H 2.20 GHz
- ❖ **Memoria Ram:** 16 GB DDR4-2666 SDRAM
- ❖ **Tarjeta gráfica:** NVIDIA GeForce GTX 1050 ti

### Metodología

Como se mencionó anteriormente nuestra metodología se basa en el pipeline gráfico mostrado en la figura 2.

### Conceptualización

En esta etapa nos reunimos en equipo para establecer qué era lo que íbamos a hacer, para esto se tomó en cuenta los requerimientos de nuestros profesores de teoría y con base en eso decidimos que la mejor opción sería crear alguna escena perteneciente a una caricatura. Dentro de las propuestas se eligió elaborar el barco de One Piece y para concretar las ideas buscamos bocetos que se encontraban en internet para visualizar qué objetos se iban a incluir y cuáles no, como fotos de juguetes inspirados en la serie o bocetos del mismo manga, como podemos observar en la figura 3. En posteriores reuniones se decidió los personajes que se incluirían así como las animaciones que realizan. También fue de gran relevancia establecer cómo se movería el personaje en el mundo y con qué teclas lo haríamos.



Figura 3. Imagenes que usamos de referencia para crear los modelos

## Modelado

En esta etapa como se mencionó previamente se usó Blender para realizar los modelos que elegimos realizar. De esta forma fue necesario familiarizarnos con el entorno al realizar los modelos requeridos para las prácticas. Gracias a esto pudimos conocer todas las herramientas que nos proporcionaba Blender como lo son las superficies, las “metaballs” o los modificados que nos permitian crear objetos más complejos que en nuestro caso se ocupan en modelos como los árboles, las velas, el timón, entre muchos más. Además fue importante una buena organización ya que debido al tiempo con el que contábamos fue necesario repartir las tareas para agilizar este proceso.

El modelo del barco y la cocina fueron realizados en blender utilizando una técnica conocida como Box Modeling, la cual consiste en tener una figura base (la mayoría de las veces, un cubo) y extruir las caras de esta figura para darle las formas deseadas. Los modelos de los personajes fueron exportados de la página sketchfab con modelos de acceso público.

## Aplicación de Texturas y colores

Ya contando con todos los modelos entonces procedimos a asignarles materiales a cada uno de los modelos para, de esta forma, darles un color o una textura que permitiera darle una mejor apariencia a ciertos objetos.

Para los objetos con colores, les asignamos colores neutros (sin modificar ninguna otra propiedad) a los objetos en blender para tener una primera vista del entorno con colores y poder escoger los indicados. Posteriormente, se exportó el modelo por material en formato FBX para guardar las propiedades de las mallas, los mapas UV y las animaciones. Los colores se “exportaron” de manera manual, anotando los valores RGB de cada material de blender en el programa final.

Para los materiales con texturas se usó la sección de UV Editing y Shading con la que cuenta Blender, estas nos permitieron asignar una determinada imagen a las caras de nuestros modelos 3D. Se utilizaron imágenes de dominio abierto de internet y otras creadas en el software Canva, como podemos ver en la figura 4 y 5.



Figura 4. Texturas descargadas de Internet.

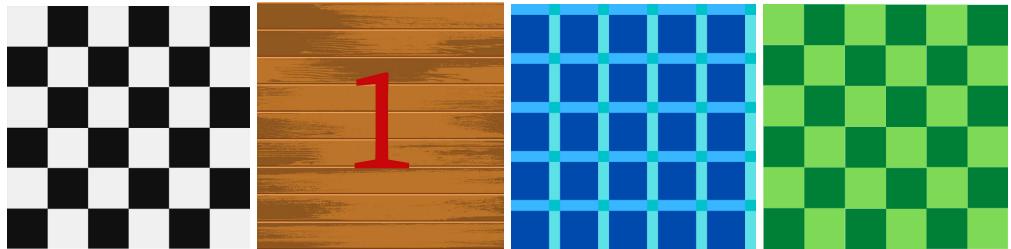


Figura 5. Texturas creadas con Canva.

También se definió la textura para poder realizar el fondo con la técnica *skybox*. Esta técnica consiste en utilizar una imagen especial recortada para un cubo para representar el fondo de nuestro modelo. En la figura 6 podemos ver la textura utilizada para el fondo.

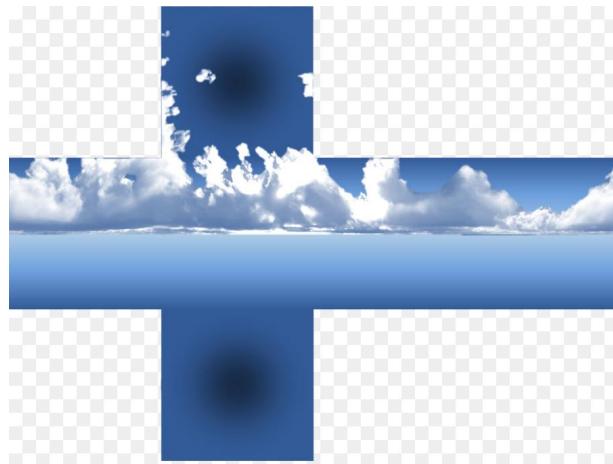


Figura 6. Textura para Skybox

## Iluminación

Para la iluminación, se decidió utilizar varias luces direccionales para poder darle una apariencia de juguete a los objetos. Se planteó el uso de luces puntuales en los modelos de faros para darle un poco más de realismo al modelo completo, pero cambiaba mucho el resultado en los materiales del modelo. También se planteó el uso de una spotlight asemejando una linterna que se moviera con el usuario.

Para implementar estos modelos de luces, se utilizó el modelo de iluminación de Phong, el cual consiste en tres componentes de luz: difusa, especular y ambiental. También se implementó el modelo de Fresnel para definir materiales translúcidos y reflectantes, como el vidrio y el oro respectivamente. Para este tipo de iluminación, también se recortó la imagen de textura del fondo para que pueda tener el efecto de refracción.

## Animación

Esta etapa se coloca aquí ya que se hizo uso de la plataforma Mixamo para importar la “armadura” que le daría dinamismo a nuestros modelos. El procedimiento consiste en primero subir el modelo que queríamos animar a Mixamo y posteriormente elegir la animación deseada para finalmente descargarla e importarla a blender para añadirla a nuestra escena. En la figura 6 podemos ver el uso de esta plataforma con nuestros personajes.

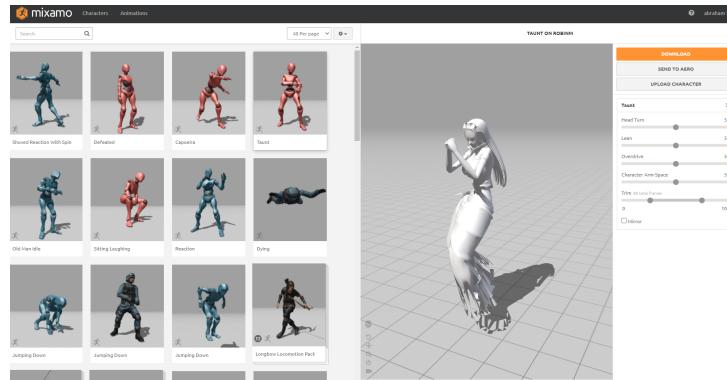


Figura 6. Plataforma de Mixamo con uno de nuestros personajes (Robin).

Para poder realizar la exportación del modelo hacia Mixamo, el modelo debe tener una pose inicial en forma de “T”, por lo cual tuvimos que modificar las posiciones de los vértices del modelo original. Una vez descargado el modelo, se estandarizó el tiempo que dura la animación para que todas duren lo mismo y no existieran problemas al importarlos al programa final con Assimp.

## Renderizado

Esta etapa consistió en crear nuestro motor de renderizado usando OpenGL. Para esto se hizo uso de archivos vistos en las prácticas como las clases model.h, camera.h, mesh.h, shader.h, material.h y stb\_imge.h que fueron proporcionados por el Dr. Sergio Teodoro Vite. Es importante mencionar que se realizaron algunas modificaciones en la clase model.h para que nuestros modelos pudieran contar con más de una animación. Además de estas clases nosotros creamos DirectLight.h, PointLight y SpotLight.h que están orientadas a la parte de iluminación.

Dentro del código principal fue importante definir nuestras variables que van desde shaders, cámaras, modelos, las luces, así como todas las que fueran necesarias para la animación o para definir los “huesos” que contendrá nuestro modelo. Para posteriormente inicializar algunas como las de los modelos, obteniendo toda la información que nos proporcionan nuestros archivos importados de blender y la de los shaders para indicar cuáles se usarán dependiendo del caso. El programa principal se divide en funciones para su modularidad. Se definen las funciones para inicializar los modelos, inicializar los shaders, para asignar los materiales y luces a cada tipo de material ya sea estático, dinámico, con iluminación de Phong o con iluminación de Fresnel, para eliminar de memoria todos los modelos y shaders inicializados, para poder realizar la correcta interpretación de la manipulación del teclado y ratón, entre otras.

Otra parte de gran relevancia fue crear nuestro ciclo de renderizado donde se envía a dibujar cada modelo y enviaremos a nuestros shaders las matrices necesarias para que realicen los cálculos que nos permitirán pasar las coordenadas 3D que nos dio como resultado Blender a coordenadas de pantalla. Para nuestro proyecto en particular se usaron dos tipos de shaders, los de vértice y los de fragmento, los cuales fueron modificados para que soporten animaciones y para usar el modelo de iluminación deseado.

Dentro de los de shaders de vértice podemos definir las operaciones que convertirán nuestras coordenadas de modelo en coordenadas de dispositivo. Esto se consigue al pasar las coordenadas que nos da Blender por varias transformaciones. Primero se aplica la transformación de modelado donde todas las transformaciones de traslación, rotación y escalamiento contenidas en la matriz de modelo afectarán a cada uno de los vértices de nuestras geometrías para moverlas a nuestra escena; de esta forma obtendremos coordenadas globales. Después las coordenadas globales deben pasar por una transformación de vista, la cual contiene la matriz de vista donde se define la posición, orientación y a hacia donde está “mirando” nuestra cámara. Al terminar esta operación obtendremos unas coordenadas de vista las cuales deberán pasar por la transformación de proyección que nos definirá como se plasmara la información. Finalmente las demás transformación como la de normalización, clipaje y la puerto de vista son realizadas internamente por OpenGL y nos darán como resultado el ver nuestra escena en una ventana que se despliega en la computadora.

Para los shaders de fragmentos, contamos con algunas variantes de los shaders para los objetos con iluminación de Phong.

1. El shader *fondo.fs* contempla el uso de 6 luces direccionales para iluminar el fondo hecho con la técnica *skybox*.
2. El shader *objetosConTexturaVariasLuces.fs* contempla el uso de texturas para los objetos, 5 luces direccionales y una spotlight para los materiales opacos que cuentan con texturas.
3. El shader *objetosSinTexturaVariasLuces.fs* contempla el uso de 5 luces direccionales y una spotlight sin textura para los materiales opacos como maderas, paredes y plásticos.
4. El shader *objetosSinTextura1Luz.fs* contempla solo el uso de una luz direccional y una spotlight para materiales reflectantes como metales opacos.
5. A su vez se implementó el shader *animados.fs* para darle iluminación con 5 luces direccionales y 1 spotlight a los modelos animados.

Para poder realizar estas operaciones, en los shaders antes mencionados se definieron las estructuras necesarias para contemplar las propiedades de cada tipo de luz. Se agregó también la estructura de luz puntual aunque esta no fue utilizada al final.

```
struct DirLight {
    vec3 direction;
    vec3 ambient;
    vec3 diffuse;
    vec3 specular;
    vec3 power;
};

struct PointLight {
    vec3 position;
    float constant;
    float linear;
    float quadratic;
    vec3 ambient;
    vec3 diffuse;
};

vec3 specular;
vec3 power;
};

struct SpotLight {
    vec3 position;
    vec3 direction;
    float cutOff;
    float outerCutOff;

    float constant;
    float linear;
    float quadratic;

    vec3 ambient;
    vec3 diffuse;
    vec3 specular;
};
```

Código de definición de las estructuras de luz.

Las propiedades de cada luz se reciben por medio de las propiedades uniform de GLSL. También se definieron funciones para calcular la aportación de cada una de las luces dependiendo del tipo de luz que sea. El realizar las funciones permitió cambiar de forma sencilla la cantidad de luces de cada tipo para cualquiera de los modelos.

```
// funciones
vec3 CalcDirLight(DirLight light, vec3 normal, vec3 viewDir);
vec3 CalcSpotLight(SpotLight light, vec3 normal, vec3 fragPos, vec3 viewDir);
vec3 CalcPointLight(PointLight light, vec3 normal, vec3 fragPos, vec3 viewDir);

void main()
{
    // Propiedades
    vec3 norm = normalize(Normal);
    vec3 viewDir = normalize(viewPos - FragPos);
    vec3 result = vec3(0.0,0.0,0.0);
    // Calculo de la luz direccional
    for(int i = 0; i < NR_DIR_LIGHTS; i++)
        result += CalcDirLight(dirLight[i], norm, viewDir);

    // Calculo de las luces puntuales
    //for(int i = 0; i < NR_POINT_LIGHTS; i++)
    //    result += CalcPointLight(pointLights[i], norm, FragPos, viewDir);

    // Calculo de la luz linterna
    result += CalcSpotLight(spotLight, norm, FragPos, viewDir);

    // se calcula el punto de textura
    vec4 texel = texture(texture_diffuse1, TexCoords);

    vec4 color = vec4(result,1.0);
    color.a = transparency;
    FragColor = texel * color;
}
```

Código Main de los shaders de Phong y funciones auxiliares.

También se utilizó el shader *II\_Fresnel.fs*, proporcionado por el profesor, para utilizar el modelo de iluminación de Fresnel y proveer a ciertos objetos como el vidrio y el oro con un efecto de reflexión. Este shader solo se modificó agregando un color difuso para proporcionarle un color difuso al material.

## Experimentos

Los primeros experimentos realizados fueron enfocados al modelado de las mallas. Utilizamos el software de Blender, para lo cual tuvimos que estudiar y tomar un curso de Blender en la plataforma de Youtube. Este aprendizaje temprano nos permitió ahorrar mucho tiempo al realizar los modelos del barco, ya que este se compone de formas complejas muy diferentes a un simple cubo y esfera. En la imagen 7 podemos observar la diferencia entre el antes y el después de aprender a utilizar el software de Blender.

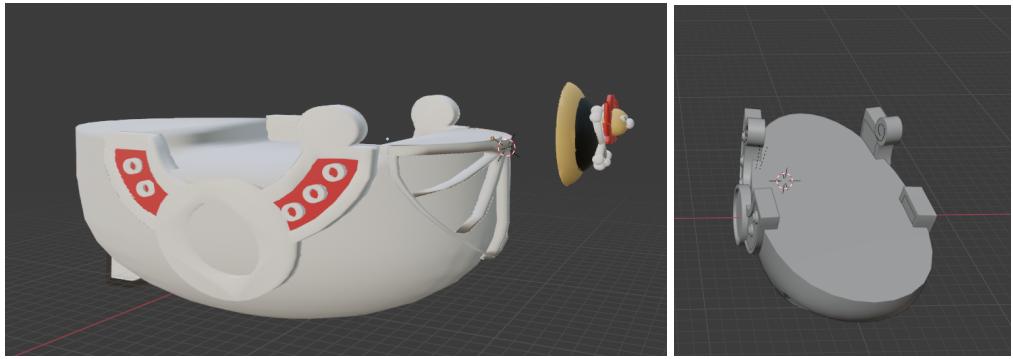


Figura 7. Modelado antes y después de tomar el curso de Blender (izquierda antes, derecha después)

La parte del barco fue realizándose tomando la parte trasera como base para poder darle buenas dimensiones al resto del barco. El proceso que tomó el modelado se puede ver en la Figura 8.



Figura 8. Evolución del modelado del barco.

Durante la carga de modelos, se utilizó una sola luz direccional, la cual dio como resultado un modelo oscuro y con sombras muy intensas que no buscábamos tener en el modelo final. En la figura 9 podemos observar los resultados de implementar solo una luz direccional y una luz spotlight.

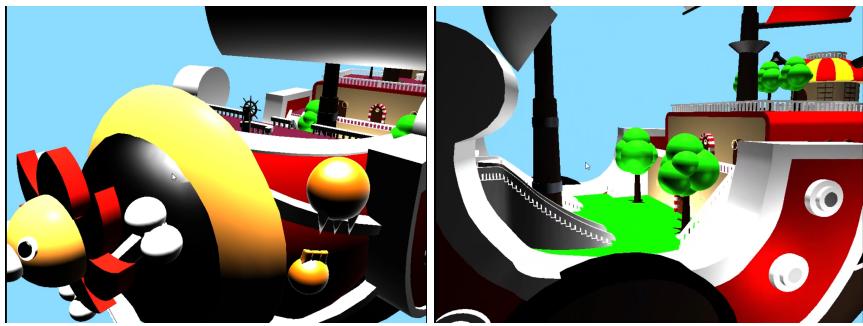


Figura 9. Modelo con iluminaciones simples.

Para tratar de corregir esto, se implementaron 4 luces puntuales al centro, costados y parte trasera del barco, pero la iluminación generada con estos 4 point lights no fue como se esperaba ya que las zonas más cercanas a la luz se iluminaban con mucha mayor intensidad, y las zonas más alejadas permanecían con las mismas sombras.

Por ello, se implementó el uso de 6 luces direccionales para poder iluminar en su totalidad el modelo. Algo que notamos fue que materiales muy reflejantes como los metales aparecían con muchos brillos especulares, resultado de las 6 luces, y otros elementos como los árboles y la madera blanca perdía su forma al no tener sombras que delimitaran sus caras. Dado esto, se dividieron los materiales entre los que ocuparían solo una luz direccional y los que ocuparían varias luces. El máximo de luces posicionales asignadas al final fueron 6 para el fondo, 5 para la mayoría de los materiales opacos y una sola luz posicional para los árboles y los metales como las cadenas. El resultado de combinar los diferentes tipos de luces los podemos ver en la parte de resultados.

Para la animación de los modelos, se utilizaron diferentes animaciones predefinidas en la plataforma Mixamo. Para el personaje 1, Zoro, no hubo ningún inconveniente al cargar el modelo y asignarle ciertas animaciones. No obstante, para el segundo personaje, Robin, hubo más problemas dada la falda que utiliza el personaje. El software de Mixamo no reconocía de manera correcta sus pies, por lo cual se tuvo que modificar los pesos de los huesos de los pies para que tuviera una correcta animación. Podemos observar la modificación a los pesos de la armadura en la figura 10.

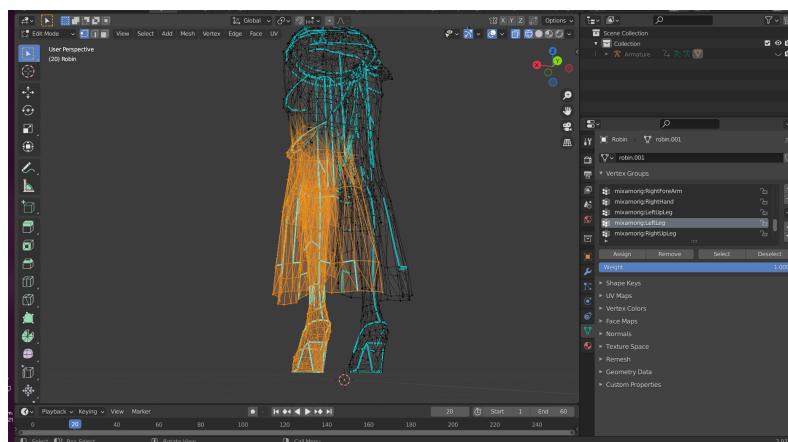


Figura 10. Modificación de los vértices a los que afecta cada hueso de la armadura

## Resultados

En los resultados podemos observar este juego de luces para el modelo, sobre todo en la parte de los árboles en los cuales permanece una sola luz puntual, y en los metales donde solo se percibe un reflejo especular, como podemos observar en la figura 11. Podemos ver también como en materiales muy reflejantes la luz de la linterna genera no solo el cambio de color del material, sino también la generación de estos brillos especulares como los que se pueden observar en las cadenas de la figura 12.

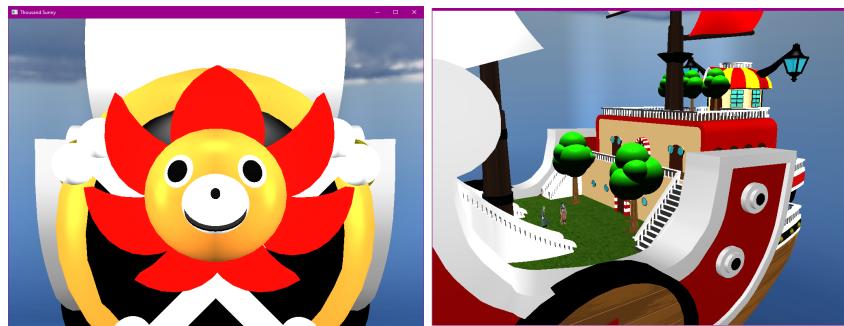


Figura 11. Juego de luces direccionales

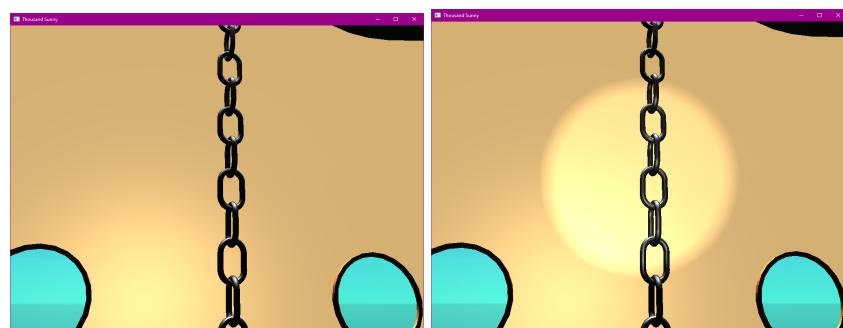


Figura 12. Efectos de la luz en materiales metálicos

Como podemos observar también en la figura 12, la iluminación para los vidrios fueron generados con el modelo de iluminación de Fresnel. Esto permite darle un efecto un poco más realista al modelo incluso cuando se contempló que el proyecto tuviera un carácter caricaturesco. El efecto de transparencia y reflexión destaca aún más los colores animados del resto del modelo.

Varios de los controles utilizados para el proyecto fueron tomados de las últimas prácticas vistas durante el laboratorio, como los movimientos de los personajes y el movimiento de la cámara. Esto permitió que tuviéramos más tiempo para realizar las pruebas de iluminación. Los controles para manejar el modelo los podemos observar en la Tabla 1.

Tecla	Efecto sobre el modelo
W	Cámara se mueve hacia adelante
S	Cámara se mueve hacia atrás
A	Cámara se mueve a la derecha
D	Cámara se mueve a la izquierda

R	Aumenta velocidad cámara
T	Disminuye velocidad cámara
Key up	Personaje 1 se mueve hacia adelante
Key down	Personaje 1 se mueve hacia atrás
Key left	Rota el Personaje 1 a la derecha
Key right	Rota el Personaje 1 a la izquierda
I	Personaje 2 se mueve hacia adelante
K	Personaje 2 se mueve hacia atrás
J	Rota el Personaje 2 a la derecha
L	Rota el Personaje 2 a la izquierda
C	Prende la linterna
V	Apaga la linterna
+	Girar a la izquierda el Timón
-	Girar a la derecha el Timón
O	Aumenta el tamaño del timón
P	Disminuye el tamaño del timón
Z	Aumenta velocidad personaje 1
X	Disminuye velocidad personaje 1
N	Aumenta velocidad personaje 2
M	Disminuye velocidad personaje 2
F1	Cambia la animación del Personaje 1
F2	Cambia la animación del Personaje 1
F3	Cambia la animación del personaje 2
F4	Cambia la animación del personaje 2
ESC	Cerrar ventana
SCROLL	Zoom in/out de la cámara

Tabla 1. Controles del proyecto

Como podemos observar también en la figura 8, contamos con un modelo en blender que es fácilmente modificable al tener un archivo en el que no se juntaron todas las mallas. Esto nos permite modificar fácilmente el modelo e incluso vender el modelo para compradores interesados.

## **Link Video demostrativo**

<https://youtu.be/1RQnYJANqV8>

## **Conclusiones**

Fuentes Huizar Carlos Apolo

Considero que el objetivo del proyecto se cumplió, ya que durante su desarrollo se logró comprender y poner en práctica cada una de las fases del pipeline gráfico. De esta forma, se aplicaron los conceptos vistos durante las clases del laboratorio y se reforzaron los conocimientos adquiridos. Además, considero que al tratarse de una temática abierta nos permitió interesarnos más en el desarrollo del proyecto, esto al elegir un tema de nuestro interés.

Por otro lado, al apoyarnos de herramientas como Mixamo, para la animación de los personajes, se logró complementar a nuestro proyecto y nos permitió contar con más tiempo para trabajar en los demás aspectos del desarrollo.

García Delgado Ian Ricardo

En conclusión, el proyecto fue de gran utilidad para poner en práctica todos los conceptos vistos durante el curso, esto ya que al pasar por todas las fases del pipeline gráfico fue posible conceptualizar nuestras ideas, crear modelos a través de blender, aplicar texturas y colores, añadir animaciones usando la aplicación Mixamo y crear nuestro motor de renderizado por medio de OpenGL. También fue posible entender a mayor detalle el uso de los shaders, los cuales son de gran importancia porque se encargan de llevar a cabo el pipeline de renderizado; sin mencionar que además realizan las operaciones necesarias para visualizar las animaciones y se encargan de aplicar los modelos de iluminación que dan aspectos realistas a nuestros objetos.

Además uno de los aspectos más importantes es que se logró que el usuario interactuara con los modelos de la escena por medio de teclas, lo que nos da una idea de cómo funcionan otros tipos de aplicaciones que requieren la intervención de una persona como lo son los videojuegos.

Lázaro Martínez Abraham Josué

Este proyecto me permitió entender mejor la metodología para desarrollar software gráfico utilizando el pipeline gráfico. También me permitió comprender en mayor medida el cómo se generan programas de software como videojuegos modernos. Algo importante de este proyecto es el uso de diferentes herramientas que nos permitieron hacer las tareas de cada etapa del pipeline gráfico mucho más sencillo, como el uso de las bibliotecas tipo Assimp y el uso de software de modelado como Blender. La incorporación de dichas herramientas

permitió que realizaramos el proyecto en un tiempo mucho menor del que se tenía previsto con anterioridad.

## Referencias

- Hearn, Donald y Baker, M. (2006). *Gráficos por computadora con OpenGL*. Pearson Educación. <https://ingenieriayeducacion.files.wordpress.com/2013/12/graficosporcomputadorayopengl.pdf>
- The Khronos Group Inc. (s.f.). *OpenGL Overview*. Recuperado el 22 de julio de 2021 de: <https://www.khronos.org/opengl/>
- The Khronos Group Inc. (s.f.). *Rendering Pipeline Overview*. Recuperado el 22 de julio de 2021 de: [https://www.khronos.org/opengl/wiki/Rendering\\_Pipeline\\_Overview](https://www.khronos.org/opengl/wiki/Rendering_Pipeline_Overview)
- TechLib. (s.f.). *OpenGL*. Recuperado el 22 de julio de 2021 de: <https://techlib.net/definition/opengl.html>
- Beal, Vangie. (s.f.). *Shader*. Webopedia. Recuperado el 22 de julio de 2021 de: <https://www.webopedia.com/definitions/shader/>

## Bibliografía complementaria

- Bailey, M., & Cunningham, S. (2009). Graphics shaders: Theory and practice. In *Graphics Shaders: Theory and Practice*. <https://doi.org/10.5860/choice.47-1464>
- Khronos. (n.d.). *OpenGL*. OpenGL. <https://www.opengl.org/>
- Laaksonen, J. (2017). OpenGL Rendering Pipeline. *OpenGL Rendering Pipeline, cli, 3–4*. [https://www.theseus.fi/bitstream/handle/10024/140206/Laaksonen\\_Jarno.pdf?sequence=1&isAllowed=y](https://www.theseus.fi/bitstream/handle/10024/140206/Laaksonen_Jarno.pdf?sequence=1&isAllowed=y)
- Molina Carmona, R., & Puchol García, J. A. (1999). *Apuntes de OpenGL*. [http://dirinfo.unsl.edu.ar/servicios/abm/assets/uploads/materiales/1080c-05\\_apuntes\\_o\\_pengl.pdf](http://dirinfo.unsl.edu.ar/servicios/abm/assets/uploads/materiales/1080c-05_apuntes_o_pengl.pdf)
- Roa, E. (n.d.). *Introducción a los Shader*. Introducción a Los Shader. <https://www.eduardoroacg.com/downloads/lectures/expoIntroShader01.pdf>
- Hearn, Baker, & Carithers. (2014). *Computer Graphics with Open GL* (4th ed.). Pearson. [https://github.com/NicholasJW/Graphics/blob/master/Computer%20Graphics%20with%20OpenGL%20\(4th%20ed.\).pdf](https://github.com/NicholasJW/Graphics/blob/master/Computer%20Graphics%20with%20OpenGL%20(4th%20ed.).pdf)
- Medellin, H. (n.d.). *Transformaciones en 3D*. Retrieved March 15, 2021, from [http://galia.fc.uaslp.mx/~medellin/Applets/Trans3D/transformaciones\\_en\\_3d.htm](http://galia.fc.uaslp.mx/~medellin/Applets/Trans3D/transformaciones_en_3d.htm)
- UNAM. (n.d.). *Transformaciones geometricas*. Página Del Colegio de Matemáticas. Retrieved March 15, 2021, from [http://prepa8.unam.mx/academia/colegios/matematicas/paginacolmate/applets/matematicas\\_VI\\_4/Applets\\_Geogebra/transfgeom.html](http://prepa8.unam.mx/academia/colegios/matematicas/paginacolmate/applets/matematicas_VI_4/Applets_Geogebra/transfgeom.html)
- Tiwari, J. (2018). *OpenGL Rendering Pipeline*. <https://www.geeksforgeeks.org/opengl-rendering-pipeline-overview/>
- JoeyDeVries. (n.d.) *Multi-light source* from <https://learnopengl-cn.github.io/02%20Lighting/06%20Multiple%20lights/>
- JoeyDeVries. (n.d) *Light casters* from <https://learnopengl.com/Lighting/Light-casters>