

Manual Técnico - Proyecto Final - CGIHC

Álvarez Sánchez Casandra Itzel 317229505

Muñoz Tamés María Ángel 314124638

Recinos Hernández Luis Mario 317244331

La ejecución de este proyecto de Computación Gráfica deberá de existir dentro del Entorno de Desarrollo Integrado (IDE, por sus siglas en inglés) de Visual Studio, ya sea en sus versiones del 2017, 2019 o 2022. Se debe considerar que el tipo de aplicación presentada corresponde a un proyecto de Visual C++, por lo que el lenguaje empleado para programar los gráficos es C++.

Por tanto, el uso de Visual Studio, con esta modalidad de proyecto pedirá por lo menos los siguientes requerimientos:

Categoría	Especificaciones mínimas
Sistema Operativo Admitido	<ul style="list-style-type: none">• Windows 7 SP1 (con las actualizaciones más recientes de Windows): Home Premium, Professional, Enterprise y Ultimate, o alguna versión superior.
Hardware	<ul style="list-style-type: none">• Procesador de 1.8 GHz y dos núcleos.• 2 GB de memoria RAM o 2.5 GB si se ejecuta en una máquina virtual.• 130 GB de espacio libre en el disco duro.• Una tarjeta de video que admita una resolución de pantalla mínima de 720p (esto es equivalente a 1280 x 720)
Instrucciones Adicionales	<ul style="list-style-type: none">• Contar con derechos de administrador.• Contar con un .NET framework 4.5.2 o una versión superior.• No será posible usar Visual Studio en dos sesiones de una misma cuenta a la vez

Metodología de desarrollo.

El propósito de implementar una metodología en un proyecto de desarrollo de software es establecer un enfoque estructurado y organizado para guiar el proceso de creación del software.

El objetivo principal de este proyecto ha sido aplicar los principios y técnicas de la programación gráfica para crear un entorno virtual convincente y visualmente atractivo. La representación precisa de la fachada y el cuarto, así como la disposición y la interacción de los objetos en el entorno, han sido aspectos clave en nuestro enfoque.

Hemos creado y ubicado varios objetos dentro del pineal, cada uno con sus características y comportamientos únicos. Todo esto se ha llevado a cabo utilizando las capacidades versátiles y flexibles de OpenGL.

Como modelo de metodología de software se utilizó Scrum, esto gracias a la flexibilidad, adaptabilidad, gestión efectiva del tiempo y prioridades, y sobre todo por la mejora continua que tendríamos gracias a utilizar esta metodología ágil.

Control de calidad

Para el proyecto el control de calidad se basa en dos aspectos principales, en que el objeto haya sido modelado con respecto a los objetos propuestos y que esté bien optimizado para su posterior carga con ayuda de OpenGL. Conforme se fue avanzando en las prácticas del semestre 2023-2 el control de calidad aumentaba en cuanto a sus requerimientos, el primero criterio que se agregó para considerar el modelo adecuado para el proyecto fue que se estuviera bien texturizado para que al incluirlo al producto final fuera visualmente atractivo para el espectador. El siguiente criterio agregado fue el uso correcto de la ambientación, es decir estuviera iluminado con sentido y adecuadamente. Por último, se consideró el criterio de animar con sentido, en caso de que el objeto tuviera una animación dentro del escenario primero se debe proponer y justificar el contexto del porqué aparecerá la animación en el producto final.

Cumplimiento de plazos y presupuesto

Para poder terminar el proyecto adecuadamente a los requisitos planteados desde el inicio, se siguió un cronograma, el cual tenía el cumplimiento de tareas, tomando en cuenta posibles retrasos o desviaciones que se podrían presentar durante el proyecto y así tomar las medidas necesarias para administrar esas situaciones y evitar un alto impacto en el proyecto.

Alcance del proyecto

El alcance del proyecto consiste en que el alumno seleccionará una fachada y un espacio, ya sea real o ficticio, y presentará imágenes de referencia de dichos espacios. Estas imágenes servirán como base para recrearlos en un entorno tridimensional utilizando la tecnología OpenGL.

El objetivo es lograr una representación virtual lo más fiel posible a las imágenes de referencia, tanto en términos de los 7 objetos presentes en ellas como en la ambientación general del espacio.

El alumno deberá utilizar sus habilidades en modelado y diseño 3D para recrear los objetos de manera precisa, tomando como referencia las imágenes proporcionadas. Además, se espera que preste atención a los detalles y características específicas de cada objeto, así como a la atmósfera y estilo general del entorno.

El resultado final del proyecto será una representación tridimensional del espacio seleccionado, donde los objetos recreados se asemejen lo más posible a las imágenes de referencia y se refleje adecuadamente la ambientación deseada.

Es importante destacar que el proyecto se enfoca principalmente en la recreación visual y estética de los espacios y objetos seleccionados, utilizando OpenGL como herramienta para lograrlo.

Limitantes del proyecto

Disponibilidad de imágenes de referencia: Puede haber limitaciones en cuanto a la disponibilidad de imágenes de referencia de alta calidad y detalladas para la fachada, el espacio y los objetos seleccionados. Esto podría dificultar la precisión y la fidelidad en la recreación 3D.

Complejidad de la fachada y el espacio seleccionados: Si la fachada y el espacio elegidos son muy complejos en términos de arquitectura, detalles estructurales o características específicas, podría requerir un mayor nivel de habilidad y esfuerzo para recrearlos fielmente en el entorno 3D.

Recursos y tiempo disponibles: El proyecto puede tener limitaciones en cuanto a los recursos disponibles, como el tiempo asignado para completarlo, la capacidad de procesamiento de la computadora o la disponibilidad de software y herramientas necesarias para la creación en OpenGL.

Nivel de detalle de los objetos: Si los objetos de referencia son muy detallados o tienen características intrincadas, puede ser desafiante recrearlos virtualmente en OpenGL con un alto nivel de fidelidad y realismo.

Conocimientos y habilidades del alumno: El nivel de habilidad y conocimiento del alumno en OpenGL y modelado 3D puede afectar la calidad y la complejidad de la recreación. Limitaciones en términos de conocimientos técnicos podrían influir en las técnicas de renderizado y en la implementación de efectos visuales avanzados.

Limitaciones legales: Si se utilizan imágenes de referencia protegidas por derechos de autor, se deben cumplir las leyes y regulaciones correspondientes. Esto podría requerir obtener permisos o utilizar imágenes de dominio público o con licencias adecuadas para evitar problemas legales.

Una vez señaladas las principales limitantes del proyecto, procedimos a realizar un análisis de estos y tratar de encontrar un equilibrio para poder recrear el ambiente correctamente.

Para evitar los problemas de complejidad en la fachada que va de la mano con el nivel de detalle de los objetos, se optó por realizar el modelado de manera personal por el alumno, sin embargo; esto nos llevo a otra limitante, en la cual se destacan los conocimientos y habilidades del alumno para poder modelar los espacios en 3D seleccionados, como algunos modelos presentaban un reto mayor al conocimiento adquirido hasta el momento de la realización se optó por recurrir a plataformas que proporcionan modelos de manera libre, siempre respetando las limitaciones legales, al tratarse de un proyecto escolar y sin fin de lucro, podemos utilizar estos modelos y adaptarlos a nuestro escenario, siempre teniendo en cuenta los aspectos mencionados en el control de calidad.

Software de modelado

Para realizar los modelos correspondientes al proyecto se utilizó el software de modelado Maya y blender para poder realizar, manipular y exportar los modelo 3D seleccionados.

Maya es un software de modelado 3D y animación ampliamente utilizado en la industria del entretenimiento, la producción de películas, los videojuegos y la creación de efectos visuales. Desarrollado por Autodesk, Maya ofrece una amplia gama de herramientas y funcionalidades que permitieron crear modelos 3D detallados.

Es importante mencionar que para poder utilizar este software correctamente debemos contar con un ordenador que cuente con las especificaciones de hardware mínimas proporcionadas por el desarrollador, las cuales son:

CPU: Procesador Intel® o AMD® de 64 bits y varios núcleos, con el conjunto de instrucciones SSE4.2 Los modelos de Apple Mac con el chip de la serie M son compatibles en modo Rosetta 2

Hardware de gráficos: Consulte las siguientes páginas para obtener una lista detallada de los sistemas y las tarjetas gráficas recomendados:

Hardware certificado para Maya

RAM: 8 GB de RAM (se recomiendan 16 GB o más)

Espacio en disco: 4 GB de espacio libre en disco para la instalación

Dispositivo señalador: Ratón de tres botones

Software para la creación de texturas para modelos

En este proyecto se utilizó GIMP para la edición y creación de texturas que posteriormente serían asignadas a los modelos realizados.

GIMP (GNU Image Manipulation Program) es un software de edición de imágenes de código abierto y gratuito. Con GIMP, los usuarios pueden realizar una amplia gama de tareas de manipulación y edición de imágenes, desde simples ajustes de color y retoques hasta la creación de ilustraciones complejas y composiciones

avanzadas.

Para poder utilizar el software GIMP de igual manera contamos con algunos requisitos mínimos para poder ser ejecutado en un ordenador correctamente, estos son:

Sistema operativo: GIMP es compatible con varios sistemas operativos, incluyendo Windows, macOS y Linux. Se recomienda utilizar la versión más actualizada del sistema operativo para garantizar un mejor rendimiento y compatibilidad.

Procesador: Se recomienda un procesador con al menos 1 GHz de velocidad para un funcionamiento fluido de GIMP.

Memoria RAM: GIMP requiere al menos 2 GB de memoria RAM para un rendimiento óptimo. Sin embargo, se recomienda disponer de 4 GB o más para trabajar con imágenes de mayor resolución y realizar operaciones más complejas.

Espacio en disco: GIMP requiere al menos 350 MB de espacio libre en el disco duro para su instalación. Además, se debe considerar espacio adicional para almacenar las imágenes y archivos creados o editados con el software.

Resolución de pantalla: GIMP funciona adecuadamente en una resolución de pantalla de al menos 1024x768 píxeles. Sin embargo, una resolución más alta permitirá una visualización más cómoda y detallada de las imágenes y herramientas.

Requerimientos Cumplidos

- **Carga de modelos**

Cada uno de ellos se introdujo al proyecto mediante el seguimiento de tres pasos:

1. **Descarga del archivo .obj**

Ya sea desde alguna plataforma con modelos como en el caso de

```

Model CajonPiso("resources/objects/CajonPiso/PisoCajon.obj");
Model CoraRosita("resources/objects/CoraRosita/cora.obj");
Model Cajon("resources/objects/Cajon/Cajon.obj");
Model Bumper("resources/objects/bumper/bumperNaranja.obj");
Model Pared("resources/objects/Pared/Pared.obj");
Model Barrera1("resources/objects/Barrera1/Barrera1.obj");
Model Barrera2("resources/objects/Barrera2/Barrera2.obj");
Model PlataformaPiso("resources/objects/PlataformaPiso/PlataformaPiso.obj");

Model piso("resources/objects/piso/piso.obj");

```

Mixamo o TurboSquid, o desde algún software de modelado como Maya o 3D Max para el caso de los objetos de nuestra propia

Pinball_Computer_Graphics_MTMA_RHLM_ASCII / Codes / resources / objects / Add file ...

hello-angel Add files via upload b9ed997 · 18 hours ago History

Name	Last commit message	Last commit da...
..		
Barrera1	Add files via upload	18 hours ago
Barrera2	Add files via upload	18 hours ago
Cajon	Add files via upload	18 hours ago
CajonPiso	Add files via upload	18 hours ago
CoraNaranja	Add files via upload	18 hours ago
CoraRosita	Add files via upload	18 hours ago
Flipper	Add files via upload	18 hours ago
Pared	Add files via upload	18 hours ago
PlataformaPiso	Add files via upload	18 hours ago
Pompompurin	Add files via upload	18 hours ago
bumper	Add files via upload	18 hours ago
piso	Add files via upload	18 hours ago

creación.

Cada uno de estos archivos fueron guardados en la ruta **resources/objects** dentro de la carpeta de solución del proyecto, así mismo, al interior de **objects** se podrán observar carpetas especiales para cada elemento del proyecto, donde se encontrarán archivos necesarios para el despliegue de los mismos, como los png para las texturas o los archivos .mtl.

2. Declaración de constantes Model

Dentro del código del proyecto, en el archivo **ProyectoFinal.cpp**, se usará el formato:

Model [NOMBRE_MODELO]("RUTA_DE_MODELO");

Para hacer que la solución del proyecto note la presencia de los modelos junto con los archivos adicionales que este requiera para un correcto despliegue.

3. Declaración de estados iniciales

Finalmente, cada uno de los modelos tuvo que ser colocado dentro del tablero siguiendo a las operaciones geométricas de traslación, escala y rotación, que nos permiten colocar al centro del objeto siguiendo un sistema de coordenadas **x**, **y** y **z**, modificar el tamaño y con respecto a qué eje se tendrá la orientación del modelo

```
model = glm::mat4(1.0f);
model = glm::translate(model, glm::vec3(0.0f, 0.0f, 0.0f));
model = glm::scale(model, glm::vec3(10.0f));
staticShader.setMat4("model", model);
Cajon.Draw(staticShader);

model = glm::mat4(1.0f);
model = glm::translate(model, glm::vec3(0.0f, 0.0f, 0.0f));
model = glm::scale(model, glm::vec3(10.0f));
staticShader.setMat4("model", model);
CajonPiso.Draw(staticShader);

model = glm::mat4(1.0f);
model = glm::translate(model, glm::vec3(0.0f, 0.0f, 0.0f));
model = glm::scale(model, glm::vec3(10.0f));
staticShader.setMat4("model", model);
CoraRosita.Draw(staticShader);

model = glm::mat4(1.0f);
model = glm::translate(model, glm::vec3(0.0f, 0.0f, 0.0f));
model = glm::scale(model, glm::vec3(10.0f));
staticShader.setMat4("model", model);
Bumper.Draw(staticShader);

model = glm::mat4(1.0f);
model = glm::translate(model, glm::vec3(0.0f, 0.0f, 0.0f));
model = glm::scale(model, glm::vec3(10.0f));
staticShader.setMat4("model", model);
Pared.Draw(staticShader);

model = glm::mat4(1.0f);
model = glm::translate(model, glm::vec3(0.0f, 0.0f, 0.0f));
model = glm::scale(model, glm::vec3(10.0f));
staticShader.setMat4("model", model);
Barrera1.Draw(staticShader);

model = glm::mat4(1.0f);
model = glm::translate(model, glm::vec3(0.0f, 0.0f, 0.0f));
model = glm::scale(model, glm::vec3(10.0f));
staticShader.setMat4("model", model);
Barrera2.Draw(staticShader);

model = glm::mat4(1.0f);
model = glm::translate(model, glm::vec3(0.0f, 0.0f, 0.0f));
model = glm::scale(model, glm::vec3(10.0f));
staticShader.setMat4("model", model);
PlataformaPiso.Draw(staticShader);
```

respectivamente. Para ello, en la sección de **Escenario**, dentro de la estructura **while(!glfwwindowshouldclose(window))** se colocaron las líneas de código:

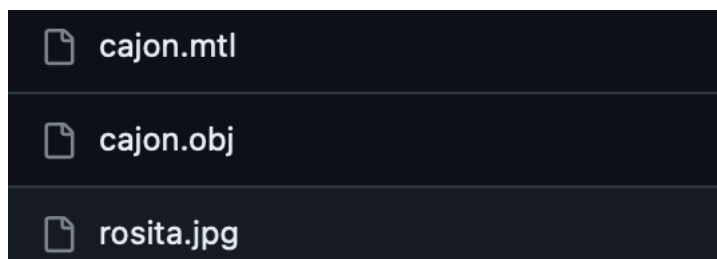
```
model = glm::mat4(1.0f);  
model = glm::translate(model, glm::vec3([coord_X], [coord_Y], [coord_Z]));  
model = glm::scale(model, glm::vec3([valor_Escala]));  
model = glm::rotate(model, glm::radians([var_rotación]), glm::vec3([eje_X], [eje_Y], [eje_Z]))
```

Finalmente, para tomar todas las configuraciones programadas, se hace uso del método **Draw** incluido en las variables modelo, y de la sentencia **staticShader** para colocar al mismo dentro del escenario. Estas líneas se ven inmediatamente después del código para declarar los estados iniciales del modelo de la siguiente forma:

```
staticShader.setMat4("model", model);  
[modelo_a_cargar].Draw(staticShader);
```

- **Texturas de Modelo**

La textura en los modelos venían incluidas en la carpeta de cada uno de los modelos importados, junto con el archivo .obj, se tuvo un archivo mtl.



Archivos como los mostrados aquí, se repitieron para cada uno de los modelos cargados en el proyecto.

- **Creación e implementación de avatar (Pompompurin)**

Al consistir en otro de los modelos que tuvimos que cargar dentro de la escena del proyecto, nuestro personaje también requirió de estar almacenado en su propia carpeta dentro de la ruta resources/objects; ser declarado como una variable modelo para finalmente ser «dibujado» dentro del ambiente virtual con ciertas operaciones geométricas, sin embargo, dado que nuestro personaje, Pompompurin debía de poder animarse de tal forma que mueva sus 4 extremidades cada vez que este avance por el tablero de pinball, fue


```
//*****
//VALORES DE POSICIÓN Y ANIMACIÓN DE POMPOMPURIN
//*****
model = glm::translate(glm::mat4(1.0f), glm::vec3(0, 1.7, 0));
model = glm::translate(model, glm::vec3(posX, posY, posZ));
//model = glm::translate(model, glm::vec3(0,0,0));Puede que esta línea no sea importante para la posición de Pompompurin
model = glm::scale(model, glm::vec3(0.05f));
//La variable orienta se encarga de girar a Pompompurin, para que rote de izquierda a derecha, se hace en referencia al eje Z
tmp = model = glm::rotate(model, glm::radians(orienta), glm::vec3(0.0f, 1.0f, 0.0f));
staticShader.setMat4("model", model);
torso.Draw(staticShader);

//Pierna Der
model = glm::translate(tmp, glm::vec3(-1.8f, 0.0f, 0.1f));
model = glm::rotate(model, glm::radians(giroPiernaDer), glm::vec3(1.0f, 0.0f, 0.0f));
staticShader.setMat4("model", model);
piernaDer.Draw(staticShader);

//Pierna Izq
model = glm::translate(tmp, glm::vec3(0.5f, 0.0f, 0.1f));
model = glm::rotate(model, glm::radians(giroPiernaIzq), glm::vec3(1.0f, 0.0f, 0.0f));
staticShader.setMat4("model", model);
piernaIzq.Draw(staticShader);

//Brazo Der
model = glm::translate(tmp, glm::vec3(-2.8f, 3.0f, 0.0f));
model = glm::translate(model, glm::vec3(-0.75f, 2.5f, 0));
model = glm::rotate(model, glm::radians(giroBrazoDer), glm::vec3(1.0f, 0.0f, 0.0f));
staticShader.setMat4("model", model);
brazoDer.Draw(staticShader);

//Brazo Izq
model = glm::translate(tmp, glm::vec3(2.8f, 3.0f, 0.0f));
model = glm::translate(model, glm::vec3(0.75f, 2.5f, 0));
model = glm::rotate(model, glm::radians(giroBrazoIzq), glm::vec3(1.0f, 0.0f, 0.0f)); //Hacemos
// que el giro dependa de la variable declarada para el giro
staticShader.setMat4("model", model);
brazoIzq.Draw(staticShader);
```

Usando la estructura de control mostrada, podremos mover a los brazos y piernas de Pompompurin de manera alternada un total de 15 grados siempre que se presione alguna de las teclas de dirección (en la captura se muestra únicamente el código correspondiente la presión de la tecla T, el código se mantiene exactamente igual, modificando únicamente la GLFW_KEY del movimiento). Esto se logró conseguir mediante la manipulación de una variable booleana **regresoWalkCycle**.

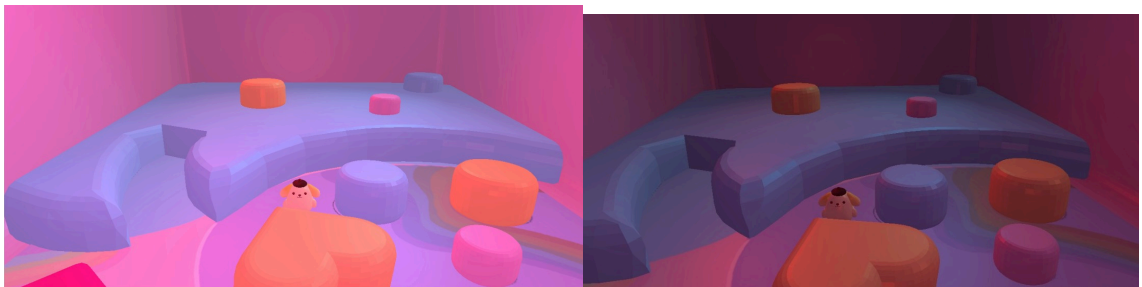
- **Recorrido (Cambio entre cámaras)**

Se consiguieron realizar dos: una que colocar a a todo el tablero en el campo de visión, simulando una vista de águila. Esto se controló mediante una pulsación de la tecla 1



- **Iluminación**

Cada cierto tiempo se implementó el código suficiente para alternar cada cierto tiempo la iluminación en el escenario, esto mediante la modificación de la componente que controla a la luz distante, direccional que afecta a espacio virtual cada determinado tiempo.



- **Implementación de Skybox**

Para la implementación de nuestro skybox de proyecto, en primer lugar se guardaron diferentes archivos .jpg que correspondieran a cada una de las caras del skybox dentro del proyecto. Estas imágenes se encontrarán en la ruta resources/skybox/2.

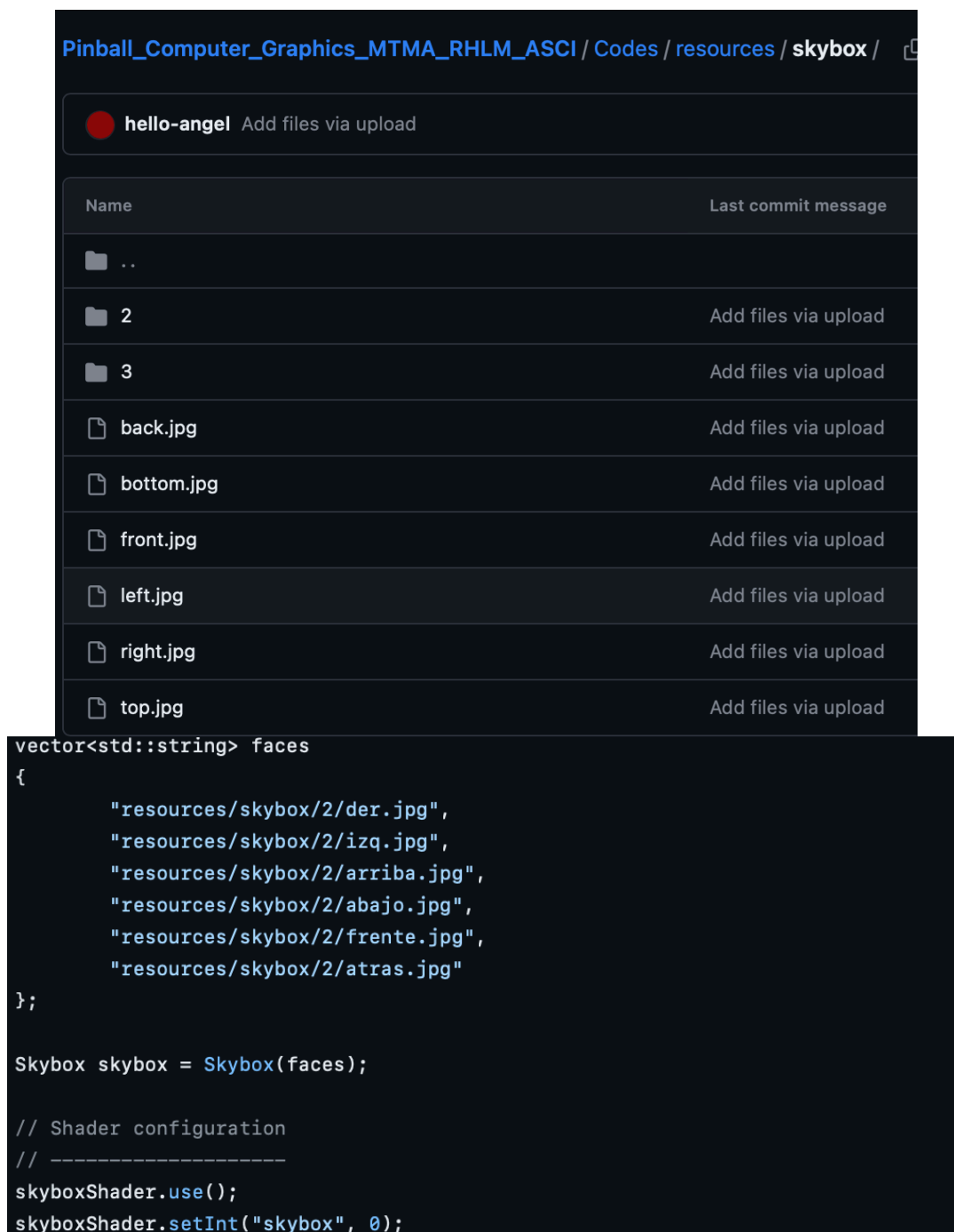
Los archivos de la carpeta podrán ser utilizados al emplear, de forma similar a como se hizo para la generación de los modelos del proyecto, el método Draw de la variable skybox creada, esto lleva el siguiente formato:

De igual forma, se aplica el uso de un shader especial para el skybox.

Conclusiones individuales

Álvarez Sánchez Casandra Itzel

El trabajo en la creación y programación del ambiente virtual del Pinball fue muy útil para terminar de reforzar todos los conocimientos que se fueron viendo a lo largo del curso. Pude combinar mis prácticas dentro de mis prácticas de laboratorio junto con la teoría dada en el salón de clases para llegar a una solución de ambiente virtual compleja, con diferentes piezas



The image shows a file explorer window for a project named 'Pinball_Computer_Graphics_MTMA_RHLM_ASCII'. The current directory is 'resources/skybox'. It contains a table of files and folders. Below the table, there is a C++ code snippet defining a skybox with six faces and a corresponding shader configuration.

Name	Last commit message
..	
2	Add files via upload
3	Add files via upload
back.jpg	Add files via upload
bottom.jpg	Add files via upload
front.jpg	Add files via upload
left.jpg	Add files via upload
right.jpg	Add files via upload
top.jpg	Add files via upload

```
vector<std::string> faces
{
    "resources/skybox/2/der.jpg",
    "resources/skybox/2/izq.jpg",
    "resources/skybox/2/arriba.jpg",
    "resources/skybox/2/abajo.jpg",
    "resources/skybox/2/frente.jpg",
    "resources/skybox/2/atras.jpg"
};

Skybox skybox = Skybox(faces);

// Shader configuration
// -----
skyboxShader.use();
skyboxShader.setInt("skybox", 0);
```

```
skyboxShader.use();
skybox.Draw(skyboxShader, view, projection, camera);

skyboxShader.use();
skybox.Draw(skyboxShader, view, projection, camera);

Shader staticShader("Shaders/shader_Lights.vs", "Shaders/shader_Lights_mod.fs");
Shader skyboxShader("Shaders/skybox.vs", "Shaders/skybox.fs");
Shader animShader("Shaders/anim.vs", "Shaders/anim.fs");
```

móviles operadas con su lógica distintiva y apegándonos a una estética en particular. El desarrollo de este proyecto, aunado a todos los conocimientos teóricos y prácticos que obtuve, me permitió desarrollar un poco más mi resiliencia con el cambio de equipo que tuve que hacer antes de comenzar con las actividades del proyecto.

Muñoz Tamés María Ángel

Gracias a las actividades del proyecto, tuve la oportunidad de cimentar todos mis condimentos con respecto al manejo, declaración y diseño de gráficos por computadora asistiendo de computadoras, por medio de cada uno de los requerimientos del proyecto pude poner a prueba mis habilidades para coordinarme, controlar y diseñar el trabajo que conlleva el desarrollo de una compleja escena virtual, con movimientos, diferentes modelos con texturas, iluminaciones, skyboxes cambiantes, etc. Con el fin de este proyecto, logro asegurar todos los conocimientos que he ido adquiriendo desde las sesiones teóricas y prácticas de la materia.

Del mismo modo, el desarrollo del proyecto me permitió hacerme con una mejor idea de qué cosas son necesarias para llevar un proyecto de escala relativamente grande en orden, con el uso de controles de versiones y un trabajo colaborativo coordinado.

Recinos Hernández Luis Mario

Con el desarrollo de este proyecto final pude aplicar todos los conocimientos que fui adquiriendo lo largo de curso práctico y teórico en un único paquete que a su vez demandaba de una gran organización con mis compañeras de equipo y habilidades de investigación para cubrir con todos los requerimientos del proyecto.

De igual forma, puedo mencionar que el trabajo en este proyecto me permitió no solo reparar en cada uno de los temas abordados en el curso, sino observar como mezclarlos para obtener un

adecuado y funcional producto final; relacionando el manejo de las cámaras con el movimiento de algún modelo, modificando la textura de skybox a través de un temporizador, o el movimiento de diferentes modelos para generar ciclos de animación son ejemplos de las técnicas y conocimientos que fui capaz de adquirir gracias a la realización de este trabajo.

Conclusiones grupales

Aunado a todos los conocimientos que pudimos obtener y reforzar del diseño e implementación de este proyecto, como ya mencionamos, el trabajo que llevamos a cabo principalmente nos enseñó la importancia de planear a futuro y llevar un delicado control de las actividades que cada quien desarrollará. Con el trabajo realizado, pudimos darnos cuenta de muchos hábitos que pudimos haber tenido a la hora de empezar que nos hubieran facilitado el avance de los diferentes puntos tocados en el trabajo. De igual forma, este proyecto nos dejó practicar nuestras habilidades de investigación; en los requerimientos que no habían sido tocado en las sesiones de laboratorio de cada uno, tuvimos que aprender a buscar por la documentación necesaria para completar el trabajo. Así como también, practicar una comunicación efectiva entre cada integrante para asignar las fortalezas de cada integrante a las tareas indicadas.