UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE INGENIERÍA

DIVISIÓN DE INGENIERÍA ELÉCTRICA

INGENIERÍA EN COMPUTACIÓN

COMPUTACIÓN GRÁFICA E INTERACCIÓN HUMANO COMPUTADORA

MANUAL TÉCNICO

“FERIA”

INTEGRANTES:

319156715

GRUPO DE LABORATORIO: 03

GRUPO DE TEORÍA: 05

SEMESTRE 2025-2

FECHA DE ENTREGA LÍMITE: 20/05/2025

CALIFICACIÓN: _____

Contenido

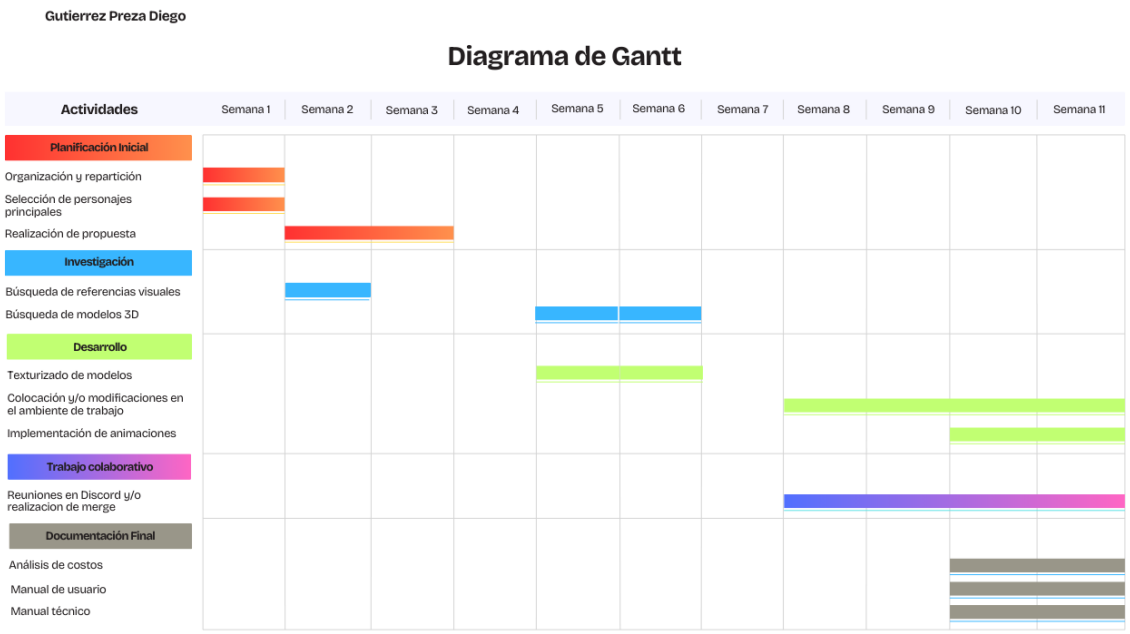
Objetivo

Por medio de herramientas orientadas al desarrollo de elementos gráficos generados por computadora como lo es lenguaje C++, OpenGL y Blender, se desarrolló una feria con distintos elementos de los universos de Mario Bros, Snoopy y el increíble mundo de Gumball, este proyecto fue propuesto por el profesor Roque, en donde se tienen 6 stands distintos los cuales corresponden a distintos puestos, que son boliche, bese Ball, dardos, topos, lanzamiento de hacha y dados cada uno de estos puestos tiene distintas animaciones las cuales hacen uso de transformaciones básicas, siendo estas escalamiento, rotación y traslación.

A su vez, entre los objetivos encontramos el uso correcto de texturizado y con ello, uso correcto del mapeado UV para ajustar las imágenes de las texturas. En cuanto a la iluminación, se desarrolló un ciclo de día y noche en el cual la iluminación disminuye cada cierto tiempo y se encienden luces automáticas, también se implementó que en el ciclo de día y noche se viera el efecto de que la luz del sol avanza sobre los elementos, simulando el efecto del sol, a su vez cada puesto cuenta con iluminación que se activa o desactiva por medio del teclado.

Finalmente se implementaron 3 cámaras distintas, las cuales son una cámara aérea que nos permite ver toda la feria, una cámara en tercera persona para recorrer la feria con el avatar, y finalmente una cámara que enfoca a cada uno de los puestos para poder visualizar tanto el puesto como la animación correspondiente a cada uno.

Cronograma de trabajo



Alcance del proyecto

1.Modelado de la feria

- Estructura exterior de cada uno de los stands siguiendo el estilo visual de la cada uno de los universos antes mencionados.
- Decoración de cada uno de los stands, la cual es distinta en todos ellos, haciendo que la feria tenga un aspecto único para cada atracción.
- Modelado de cada uno de los elementos exteriores, así como bancas, arboles, postes, NPC, etc.

2. Texturizado

- Aplicar texturas personalizadas a cada modelo, donde dichas texturas se crean fueron editadas en GIMP.
- Edición y retoque de imágenes para mantener las texturas en el formato especificado, siendo este PNG y escala en algún número de base 2.
- Modificación de los mapas UV a cada modelo 3D, para asegurarnos que las texturas se alineen correctamente con la geometría del modelo y que no haya detalles indeseados.

3. Animación

- Implementación de animaciones tanto simples como complejas. Las animaciones de los stands están para que el usuario pueda activar para interactuar con el entorno y también se implementaron animaciones que están todo el tiempo ejecutándose en los personajes principales

4. Iluminación

- implementación de DirectonalLight para el sol, PointLight para los postes de luz, y SpotLight para los focos de cada stand.
- Ajuste del sistema de iluminación para que este sea automático de acuerdo con el ciclo de día y noche.
- Creación de luz ambiental que sea la que esté presente en todo el espacio, de forma similar a como ilumina el sol en la vida real.

5. Cámaras

- Implementación de 3 distintas cámaras para la distinta visualización de la feria y que de esta forma se tenga una mejor apreciación de los detalles.

Costos

Con el objetivo de hacer mas claro y comprensible el desarrollo de costos, se dividirá esta sección en los distintos tipos:

Costo de Recursos Humanos

El desarrollo de este proyecto fue llevado a cabo de manera independiente, con una carga de trabajo total de 200 horas. Considerando un nivel de experiencia intermedio en lenguaje C++ y tomando como referencia una tarifa promedio de $110.00 MXN por hora, el costo por concepto de recursos humanos asciende a:

Costo de Recursos Humanos = $22,000.00 MXN

Costos Fijos

Se identifican como costos fijos aquellos gastos que no se ven afectados por la cantidad de trabajo realizado:

- **Electricidad:** Se empleó una computadora de escritorio, un monitor y un estabilizador. Costo mensual estimado: $430.00 MXN.
- **Internet:** Se utilizó el servicio de internet proporcionado por Totalplay, con un costo mensual de $420.00 MXN. Dado que el proyecto tuvo una duración de 11 semanas, se considera un periodo de 2.75 meses, resultando en un costo total de $1,155.00 MXN.
- **Oficina:** Se considera una oficina ubicada en la colonia Roma Norte de la Ciudad de México. El espacio de trabajo tuvo un costo mensual de $5,800.00 MXN. Por lo tanto, el total por 2.75 meses es de $15,950.00 MXN.
- **Depreciación de equipo:** Se utilizó una laptop Asus TUF Gaming A15 con un costo estimado de $26,000.00 MXN. Considerando una vida útil de cinco años (60 meses), la depreciación mensual es de $433.33 MXN. Multiplicado por el periodo del proyecto, se obtiene un costo de depreciación de $1,191.67 MXN.

Total, Costos Fijos = 430 + 1,155 + 15,950 + 1,191.67 = $18,726.67 MXN

Costos Variables

Los costos variables corresponden a los insumos y herramientas utilizados cuya necesidad depende directamente del trabajo ejecutado:

- **Software:** Se utilizaron herramientas de desarrollo gratuitas: Visual Studio 2022 Community Edition y GitHub, por lo que no se incurrió en costos adicionales.

Total, Costos Variables = $0.00 MXN

Costo Neto del Proyecto

El costo neto considera la suma de todos los rubros anteriores:

- Recursos Humanos: $22,000.00
- Costos Fijos: $18,726.67
- Costos Variables: $0.00

Costo Neto = $40,726.67 MXN

Costo con Margen de Ganancia

Se considera un margen de utilidad del 30% para asegurar la rentabilidad del proyecto:

- Ganancia: $40,726.67 x 0.30 = $12,218.00
- Subtotal con ganancia = $52,944.67 MXN

Costo Final al Cliente

Para el cálculo del costo total al cliente, se consideran impuestos y comisiones por pasarelas de pago:

- IVA (16%) = $8,471.15
- Subtotal con IVA = $61,415.82
- Comisión por pago con tarjeta (5.5%) = $3,377.87

Costo Total al Cliente = $64,793.69 MXN

El precio final este proyecto, considerando los costos operativos, insumos, honorarios profesionales, utilidad proyectada y obligaciones fiscales, asciende a **$64,793.69 MXN**. Este monto garantiza tanto la cobertura de los gastos como una remuneración adecuada al trabajo realizado.

Documentación

Archivo Main

- **bateo(…)**

Renderiza los modelos que conforman el juego de jaula de bateo. Incluye el stand, el bate, la bola, el objetivo a golpear y carteles decorativos. Esta función asegura que todos los objetos sean visualizados con la iluminación y transformaciones apropiadas.

Parámetros:

glm::mat4& model: matriz de transformación aplicada.

GLuint uniformModel: ubicación del uniforme model en el shader.

std::vector<Model*>& objetosBateo: vector con los modelos del juego de bateo.

- **bolos(…)**

Renderiza los elementos del puesto de bolos, como los bolos, la bola, el stand, la mesa y carteles decorativos. La función aplica las transformaciones correspondientes y envía cada modelo al pipeline de renderizado.

Parámetros:

glm::mat4& model: matriz transformacional general.

GLuint uniformModel: identificador del uniforme de modelo en el shader.

std::vector<Model*>& objetosBolos: modelos del puesto de bolos.

- **ambientación(…)**

Renderiza todos los objetos de ambientación general de la feria: árboles, luminarias, bancas, basureros, laguna, entre otros. Sirve para dar vida y coherencia visual al entorno no jugable.

Parámetros:

glm::mat4& model: matriz de transformación global.

GLuint uniformModel: uniforme de transformación en el shader.

std::vector<Model*>& objetosAmbientacion: modelos del escenario estático.

- **comida(…)**

Dibuja los puestos de comida y sus respectivos elementos como palomitas, helados, algodones de azúcar. Añade atractivo visual y realismo temático a la feria.

Parámetros:

glm::mat4& model: matriz utilizada para ubicar modelos.

GLuint uniformModel: ubicación del uniforme en el shader.

std::vector<Model*>& objetosComida: modelos que representan comida y puestos.

- **NPCs(…)**

Renderiza los modelos de los personajes no jugables (NPCs) como Carrie, Teri, etc. Están repartidos por la feria con fines decorativos y de ambientación.

Parámetros:

glm::mat4& model: matriz de transformación común.

GLuint uniformModel: referencia al uniforme del modelo.

std::vector<Model*>& personajesNPCs: lista de personajes a renderizar.

- **zonaFotos(…)**

Renderiza una sección temática pensada como zona de fotos, que incluye personajes posando y escenarios decorativos. Se utilizan transformaciones animadas para ciertos elementos (alas, patas, etc.).

Parámetros:

glm::mat4& model: matriz base para transformaciones.

GLuint uniformModel: ubicación del uniforme model.

std::vector<Model*>& objetosZonaFotos: modelos que componen el escenario.

float deltaTime: tiempo entre frames, usado para animaciones.

- **actualizarBateo(float velocidad)**

Controla el movimiento de la bola y del bate en el juego de bateo. Se encarga de la animación continua de ambos objetos durante el juego (antes o después del impacto).

Parámetro:

velocidad: delta time multiplicado por un factor, controla velocidad de animación.

- **actualizarBolos(float velocidad)**

Se encarga de la animación de la bola de boliche, haciéndola avanzar hasta impactar contra los bolos. También puede gestionar rebotes o reinicio de la bola.

Parámetro:

velocidad: control temporal para animar la bola.

- **lanzarBateo()**

Inicia el ciclo de animación donde la bola se lanza y el bate se mueve para golpearla. Se vincula a la tecla Y en el teclado.

- **lanzarBola()**

Activa el lanzamiento de la bola de boliche, controlando su movimiento hasta impactar con los bolos. Responde a la tecla O.

Funciones dadas por el Ing. Jose Roque RG

- **mainWindow.Initialise()**

Inicializa la ventana principal de OpenGL con GLFW y GLEW, estableciendo los parámetros de contexto gráfico, tamaño de buffer, y capturando eventos.

Parámetros:

No recibe parámetros directamente (usa los valores definidos al construir mainWindow).

- **CreateObjects()**

Crea varias mallas geométricas primitivas (pirámide, piso, vegetación, letrero) con sus vértices e índices, y las almacena en meshList.

- **CreateShaders()**

Carga los archivos de shader (vertex y fragment) desde las rutas vShader y fShader, y los agrega al vector shaderList.

- **LoadModel(...)**

Carga un modelo .obj desde una ruta de archivo específica al objeto Model. Se llama muchas veces para cada objeto 3D de la feria (puestos, ambientación, personajes, etc.).

Parámetros:

const char* fileName: ruta relativa al archivo .obj del modelo.

- **Skybox(...)**

Crea una caja de cielo con 6 imágenes que se renderiza en segundo plano para simular un entorno envolvente. Se crean dos: una para el día y otra para la noche.

Parámetros:

std::vector<std::string> faceLocations: rutas de imágenes para cada cara del cubo (derecha, izquierda, abajo, arriba, frente, atrás).

- **Material(...)**

Crea un material con propiedades de reflectancia para aplicar en iluminación especular.

Parámetros:

float specIntensity: intensidad del reflejo especular.

float shininess: brillo del material (mayor valor = reflexión más pequeña y brillante).

- **DirectionalLight(...)**

Constructor que inicializa una luz direccional, la cual simula una fuente de luz lejana como el sol. No tiene posición, solo dirección, y afecta de manera uniforme todos los objetos en la escena.

Parámetros:

GLfloat red, green, blue: componentes del color de la luz.

GLfloat ambientIntensity: intensidad con la que la luz afecta las zonas no iluminadas directamente (luz ambiente).

GLfloat diffuseIntensity: intensidad de la luz directa sobre los objetos.

GLfloat xDir, yDir, zDir: dirección en la que apunta la luz.

- **PointLight(...)**

Constructor de una luz puntual, la cual emite luz en todas las direcciones desde una posición específica. Este tipo de luz se atenúa con la distancia, simulando una lámpara o farol.

Parámetros:

GLfloat red, green, blue: componentes de color de la luz.

GLfloat ambientIntensity: intensidad ambiental.

GLfloat diffuseIntensity: intensidad difusa.

GLfloat xPos, yPos, zPos: posición de la luz en el mundo 3D.

GLfloat constant, linear, exponent: coeficientes de atenuación de la luz. Estos determinan cómo disminuye la intensidad con la distancia (modelo de atenuación estándar).

- **SpotLight(...)**

Constructor de una luz tipo foco (spotlight), que es como una luz puntual pero con una dirección definida y un ángulo de apertura. Es útil para simular linternas, faros o reflectores.

Parámetros:

GLfloat red, green, blue: componentes del color.

GLfloat ambientIntensity: luz ambiental emitida.

GLfloat diffuseIntensity: intensidad difusa.

GLfloat xPos, yPos, zPos: posición en el espacio.

GLfloat xDir, yDir, zDir: dirección hacia la que apunta el foco.

GLfloat constant, linear, exponent: coeficientes de atenuación.

GLfloat edge: ángulo de apertura del cono de luz (en grados), define el límite de iluminación del foco.

- **getCamaraActiva()**

Devuelve el ID entero de la cámara actualmente seleccionada por el usuario (0 = aérea, 1 = tercera persona, 2 = cámara de puestos).

- **mouseControl(x, y)**

Recibe los desplazamientos del mouse en X y Y, y ajusta la orientación (yaw, pitch) de la cámara activa.

Parámetros:

GLfloat x: desplazamiento horizontal del mouse.

GLfloat y: desplazamiento vertical del mouse.

- **calculateViewMatrix()**

Devuelve la matriz de vista (glm::mat4) calculada a partir de la posición y dirección de la cámara. Usada para transformar la escena desde el punto de vista del usuario.

- **UseShader(...)**

Activa el shader compilado previamente para empezar a usarlo en las operaciones de dibujo.

- **SetSpotLights(...)**

Envía un arreglo de luces tipo spotlight al shader activo. Cada luz contiene información de color, dirección, atenuación, intensidad y ángulo de corte. Se usa para iluminar zonas específicas de la escena como los juegos individuales (bateo, bolos, etc.).

Parámetros:

SpotLight* lights: puntero a un arreglo de luces SpotLight activas.

unsigned int lightCount: cantidad de luces SpotLight que se deben enviar al shader

- **SetPointLights(...)**

Enlaza al shader un conjunto de luces puntuales, como faroles o postes. Estas luces emiten en todas las direcciones desde un punto y se atenúan con la distancia.

Parámetros:

PointLight* lights: arreglo de luces PointLight.

unsigned int lightCount: número de luces a activar y pasar al shader.

- **SetDirectionalLight(...)**

Asigna una única luz direccional al shader, la cual simula una fuente de luz como el sol. Esta luz afecta toda la escena desde una dirección fija.

Parámetros:

DirectionalLight* dLight: puntero a una instancia de luz direccional con los parámetros ya definidos.

- **RenderModel()**

Dibuja en pantalla un modelo previamente cargado en OpenGL.

- **swapBuffers()**

Intercambia el buffer de dibujo en la ventana para mostrar la imagen renderizada. Hace parte del ciclo de renderizado en tiempo real.

Archivos separados

Archivo: ZonaFotos.cpp

void zonaFotos (glm::mat4 model, GLuint uniformModel, std::vector<Model*> objetosZonaFotos, float deltaTime)

Renderiza y anima los personajes Snoopy, Woodstock, Gumball y Yoshi dentro de una zona específica.

- model: matriz de transformación.
- uniformModel: ubicación del uniforme en el shader para la matriz de modelo.
- objetosZonaFotos: modelos 3D usados (Snoopy, Gumball, Woodstock, Yoshi y sus partes).
- deltaTime: tiempo entre frames, usado para animaciones suaves.

Animación de Gumball

- **Variables principales:**
  - angulovaria, tiempoGumball: acumuladores de tiempo y ángulos.
  - movGumball: desplazamiento sobre el eje Z.
  - avanzaGumball: controla dirección del movimiento.
  - anguloExtremidadesG: base para calcular animaciones de brazos, piernas y cola.
  - rotacionContinuaGumball: rotación del cuerpo mientras camina.

Comportamiento: Gumball se desplaza de un lado a otro sobre el eje Z, gira cuando cambia de dirección, salta suavemente y mueve extremidades y cola con un efecto de aplauso y abrir y cerrar las piernas.

Archivo: NPCs.cpp

void NPCs(glm::mat4 model, GLuint uniformModel, std::vector<Model*> personajesNPCs)

Renderiza múltiples NPCs (personajes estáticos) ubicados en diferentes puestos del mapa.

- model: matriz base de transformación.

- uniformModel: identificador del uniforme de modelo.
- personajesNPCs: lista de modelos 3D de personajes.

Funciones auxiliares por personaje:

Todas estas funciones usan la misma estructura:

void render[Nombre](glm::mat4 model, GLuint uniformModel, Model& personaje, glm::vec3 posicion, int grados)

Renderizan un modelo personaje en la posición y rotación especificadas.

- posición: vector de posición en el mundo.
- grados: rotación sobre eje Y en grados.

Aplica a: renderCharlie, renderCarrie, renderTeri

Archivo: Tierra.cpp

void tierra(glm::mat4 model, GLuint uniformModel, Texture& tierra, std::vector<Mesh*> meshList)

Dibuja caminos y zonas de piso para cada sección del mapa.

- tierra: textura que se aplicará.
- meshList: lista de mallas; se usa meshList[5] para el terreno.

void renderTierra(glm::mat4 model, GLuint uniformModel, Texture& tierra, Mesh& piso, glm::vec3 posicion, glm::vec3 escala)

Renderiza una porción de terreno texturizado con escala personalizada. *

- piso: malla a dibujar.
- posicion: posición global.
- escala: vector de escalado.

Archivo: Window.cpp

Window::Window() / Window::Window(GLint windowWidth, GLint windowHeight)

Constructores de la clase Window. Inicializan tamaño, estado del teclado y cámara.

- int Window::Initialise(): Inicializa la ventana, GLEW y el contexto OpenGL.

- void Window::createCallbacks(): Asigna las funciones de manejo de teclado y mouse.
- void Window::ManejaTeclado(GLFWwindow* window, int key, int code, int action, int mode): Maneja eventos de teclado para control de cámaras y luces.
    - key: tecla presionada.
    - action: tipo de evento (presionar, soltar).
- void Window::ManejaMouse(GLFWwindow* window, double xPos, double yPos): Calcula la diferencia en movimiento del mouse para la cámara.
- GLfloat getXChange() / GLfloat getYChange(): Devuelven el cambio de posición del mouse en X o Y.

Variables importantes:

- camaraActiva: selecciona entre cámara aérea, tercera persona o por puestos.
- lucesSpot[MAX_SPOT_LIGHTS]: controla encendido/apagado de luces por puesto.
- puestoActual: índice del puesto activo para animación o iluminación.
- muevex: modificador de movimiento horizontal general.

Archivo: Ambientacion.cpp

void ambientacion(glm::mat4 model, GLuint uniformModel, std::vector<Model*> objetosAmbientacion)

Renderiza decoraciones como bancas, árboles, botes, estantes, etc.

- objetosAmbientacion: lista de modelos ambientales.

Funciones auxiliares de render:

void render[Objeto](glm::mat4 model, GLuint uniformModel, Model& modelo, glm::vec3 posicion[, int grados])

- Aplican transformaciones y renderizan el modelo.
- Las versiones con grados permiten rotación (e.g. bancas, botes).
- Incluyen: renderBanca, renderLuminaria1/2/3, renderArbol1/2/3, renderBote1/2, renderPlantaDecorativa, renderEstante, renderLuminariaTecho, renderLaguna.

Archivo: Bateo.cpp

void bateo(glm::mat4 model, GLuint uniformModel, std:vector<Model*> objetosBateo)

Renderiza el puesto de bateo: jaula, bardas, cartel, bola y bats.

- void renderStand2(...): Renderiza el stand principal.
- void renderBardas(...): Dibuja tres bardas para el entorno del juego.
- void renderbolaBateo(...): Renderiza dos bolas: una fija y otra animada que se mueve y rota.
- void renderCartelBateo(...): Dibuja el cartel en la parte trasera del escenario.
- void renderBats(...): Renderiza dos bats: uno animado que rota y se eleva, y otro estático decorativo.
- void actualizarBateo (float deltaTime): Gestiona el avance de la bola, el impacto del bat, y la animación de retorno para ambos.
- void lanzarBateo (): Activa el lanzamiento de la bola si no hay otra animación activa.

Variables importantes:

- posicionBola: posición animada de la bola.
- rotacionBat, alturaBat: estado del bat.
- bolaEnMovimiento, batGolpeando, bolaRegresando, batRegresando: flags de control.
- VELOCIDAD_BOLA_ORIGINAL: velocidad base de la bola.

Archivo: Bolos.cpp

void bolos(glm::mat4 model, GLuint uniformModel, std::vector<Model*> objetosBolos)

Renderiza el puesto de bolos con stand, bolas, pines, cartel y mesa.

- **void renderStand5(...):** Renderiza el stand de bolos.
- **void renderPines(...):** Dibuja los 10 pinos en formación triangular. Simula caídas con rotaciones.
    - base: posición inicial.
    - anguloAnimado: ángulo progresivo para caída simulada.

- **void renderBola(...):** Renderiza bolas. Si está animada, se rot dinámicamente.
    - posición: coordenadas 3D.
    - animada: si es verdad, activa la rotación.
- **void renderMesa(...):** Renderiza una mesa larga detrás del área de juego.
- **void renderCartelBoliche(...):** Renderiza el cartel trasero.
- **void actualizarBolos(float deltaTime):** Simula el lanzamiento y rotación de la bola, derribo progresivo de los pinos, y reinicio de posiciones tras caída.
- **void lanzarBola():** Activa el lanzamiento de la bola.

Variables clave:

- posicionBola, rotacionBola: estado animado de la bola.
- pinesDerribados, anguloAnimacion: lógica de caída de los pinos.

Conclusiones

El desarrollo de este proyecto representó una experiencia integral en la que se combinaron conocimientos técnicos, habilidades de diseño, planificación y gestión de recursos. A lo largo de las 11 semanas de trabajo, se logró construir un producto funcional, estéticamente atractivo y técnicamente sólido, cumpliendo con los objetivos planteados desde el inicio.

La correcta distribución del tiempo y la aplicación de herramientas como Visual Studio 2022 y GitHub permitieron mantener un flujo de trabajo eficiente y ordenado. Este proyecto es de una complejidad considerable debido a la cantidad de modelos, animaciones, lógica de iluminación, etc. que se implementó de acuerdo con las especificaciones del profesor Roque. Este proyecto no solo permite la visualización e interacción de distintos elementos, sino que nos permite un recorrido mas realista e interactivo debido a las distintas funciones que se tienen para que el usuario interactúe. Esto demuestra el dominio de conceptos fundamentales en programación gráfica y estructuración modular de código.

En el aspecto económico, se realizó un análisis detallado de costos que contempló tanto los recursos humanos como los insumos y gastos operativos. Este desglose permitió establecer un precio justo de venta, que considera no solo la inversión realizada, sino también un margen razonable de utilidad y los compromisos fiscales asociados a la comercialización del proyecto. también se estableció un cronograma por medio de un diagrama de Gantt el cual nos permite ver el flujo de trabajo, así

como la documentación realizada permite conocer los detalles del proyecto para distintas personas.

En suma, el proyecto no solo logró cumplir con los requisitos técnicos y funcionales solicitados, sino que también representa un ejemplo claro de trabajo profesional y autodirigido.

Links Modelos:

Algodon de azucar: https://www.cgtrader.com/items/4617534/download-page

Tabla: https://sketchfab.com/3d-models/cutting-board-61eadc1d646c47ac892ce418677dfb76#download

Snoopy: https://sketchfab.com/3d-models/snoopy-72116d2e288f4c45a11f323d76142a6c#download

Luminarias: https://www.turbosquid.com/es/3d-models/free-max-model-classical-street-light/965356

Botes de basura: https://www.turbosquid.com/es/3d-models/free-obj-model-street-bin/1106299

Puesto de comida: https://free3d.com/es/modelo-3d/temporary-food-stalls-80503.html

Gumball waterson: https://sketchfab.com/3d-models/gumball-fusionfall-heroes-39c568e694e1478bb0e3ed81cde9c359

Librería Premios: https://sketchfab.com/3d-models/libreria-vuota-098205aa418f4237b361328a282e21e3

Links Texturas:

Cuadros:https://www.arte-mio.mx/MLM-763335879-set-nintendo-super-mario-bros-4-cuadros-en-tela-canvas-list-_JM

The Amazing World Of Gumball Clown Mystery Explained

peanuts-the-art-of-snoopy-history-snoopy-flying-ace-snoopy-come-home

Pasto: https://img.freepik.com/fotos-premium/textura-hierba-verde-que-esta-hecha-empresa-empresa_612834-258.jpg

Skybox: https://opengameart.org/content/sky-box-sunny-day

Tabla: https://www.istockphoto.com/es/fotos/madera-clara-textura

Tierra:https://png.pngtree.com/thumb_back/fw800/background/20220711/pngtree-brown-paper-texture-ground-parcel-photo-image_999914.jpg