UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE INGENIERÍA

DIVISIÓN DE INGENIERÍA ELÉCTRICA

INGENIERÍA EN COMPUTACIÓN

COMPUTACIÓN GRÁFICA E INTERACCIÓN HUMANO COMPUTADORA

MANUAL TÉCNICO

“FERIA”

319094479

GRUPO DE LABORATORIO: 03

GRUPO DE TEORÍA: 05

SEMESTRE 2025-2

FECHA DE ENTREGA LÍMITE: 20 de mayo del 2025

CALIFICACIÓN: _____

Índice

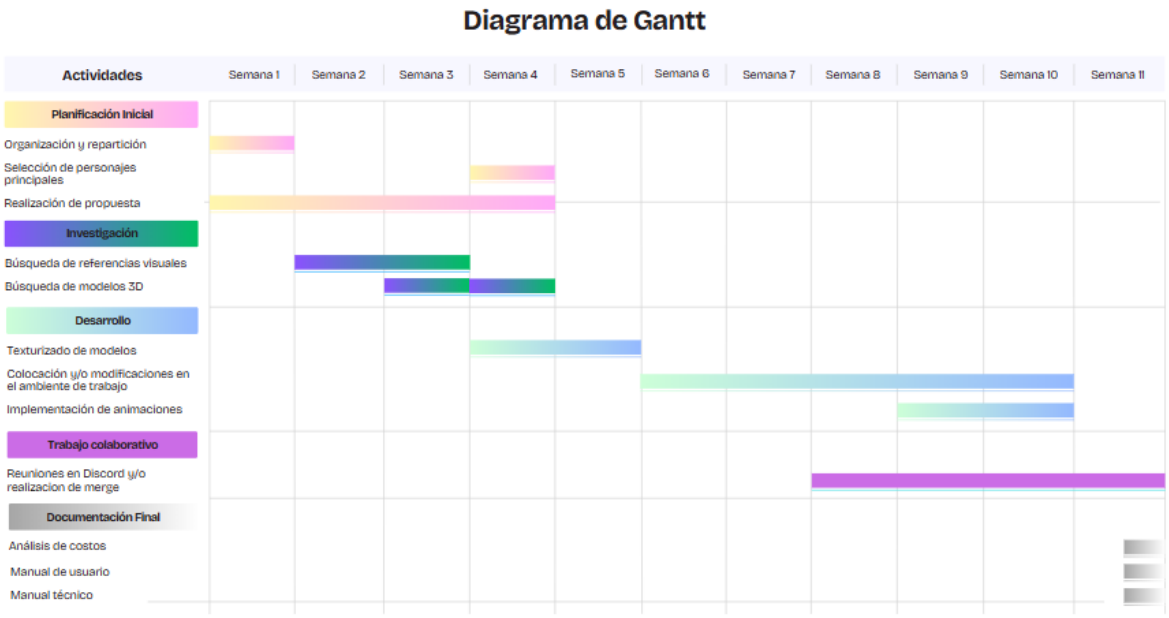
Objetivo

Este proyecto, propuesto por mi profesor de laboratorio, el Ing. José Roque Román, desarrollará un ambiente virtual por el cual se podrá navegar y estará inspirado en una feria de juegos ambientada con elementos de los universos de *Mario Bros*, *Snoopy* y *El increíble mundo de Gumball*; esta feria estará centrada en 6 puestos de juego los cuales son: lanzamiento de hacha, lanzamiento de dardos, jaula de bateo, lanzamiento de dados, línea de boliche, y “golpea al topo”. El proyecto será desarrollado utilizando herramientas especializadas en el trabajo con elementos gráficos, como *OpenGL* con *C++* en *VisualStudio*, *3dsMax*, *Blender* y *GIMP*.

Los elementos que se visualizan en el escenario serán en su mayoría modelos 3D texturizados con un correcto mapeado UV; en cada uno de los stands de los juegos habrá una animación característica del juego, así como para cada uno de los personajes principales incluidos. Como ambientación adicional, se tendrán luces que se activen por medio del teclado, una luz que funcione como un ciclo de día y noche, y luces automáticas que se activen cuando se haga de noche, y se desactiven cuando sea de día.

También se tendrán 3 diferentes cámaras: una cámara que sigue al avatar que recorre la feria en tercera persona, 6 cámaras que estarán situadas en cada juego donde desde ellas se puedan apreciar las animaciones implementadas, y una cámara aérea que visualice desde arriba toda la feria.

Cronograma



Alcance

1.Modelado

- Armado de cada stand de juego siguiendo el estilo visual de la cada uno de los universos mencionados.
- Decoración distinta para cada stand de forma que haya un aspecto único para cada atracción.
- Correcto posicionamiento de cada uno de los elementos exteriores: las bancas, árboles, postes de luz, NPC's, arco de entrada, fuente, y puestos de comida.

2. Texturizado

- Edición de imágenes para mantener las texturas en el formato necesario (.png), y escaldas en ambos ejes a la misma potencia de 2.
- Aplicación y modificación de los mapas UV para cada modelo 3D, de forma que las texturas estén alineadas correctamente con la geometría del modelo cuidando los detalles.

3. Animación

- Las animaciones características de los juegos están implementadas de forma que el usuario pueda activarlas para interactuar con el entorno, y también hay animaciones que están ejecutándose todo el tiempo en los personajes principales (Yoshi, Gumball y Snoopy), así como el letrero animado que está en el arco de entrada.

4. Iluminación

- Se utiliza una *DirectonalLight* para simular el efecto del sol, *PointLight* para los postes de luz, y *SpotLight* para los focos dentro de cada stand.
- Inclusión de un sistema de iluminación para que las luces en postes sean automáticas de acuerdo con el ciclo de día y noche.

5. Cámaras

- Implementación de 3 distintas cámaras para distintos puntos de visualización de la feria y observación de todos los detalles.

Costos

El costo del desarrollo de este proyecto se desglosa de la siguiente manera:

Costo de Recursos Humanos

Este proyecto fue llevado a cabo con una carga de trabajo total de 110 horas. Considerando un nivel de experiencia intermedio en lenguaje C++ y *OpenGL,* y tomando como referencia una tarifa promedio de $200.00 MXN por hora, el costo por concepto de recursos humanos asciende a:

Costo de Recursos Humanos = $22,000.00 MXN

Costos Fijos

Son 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.
- **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 + 1,191.67 = $2,776.67 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: $2,776.67
- Costos Variables: $0.00

Costo Neto = $24,726.67 MXN

Costo con Margen de Ganancia

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

- Ganancia: $24,726.67 x 0.30 = $7,432.80
- Subtotal con ganancia = $32,159.47 MXN

Costo Final al Cliente

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

- IVA (16%) = $5,145.51
- Subtotal con IVA = $37,304.98
- Comisión por pago con tarjeta (5.5%) = $2,051.77

Costo Total al Cliente = $39,356.75 MXN

Se debe tomar en cuenta que el proyecto es un trabajo en equipo de 3 personas, por lo que el costo mostrado debe multiplicarse por 3.

Este monto garantiza tanto la cobertura de los gastos como una remuneración adecuada al trabajo realizado.

Documentación

Archivo main

- **dardos(…)**

Renderiza todos los objetos visuales del juego de dardos, como el tablero, globos decorativos, la base y los dardos en sí. También se incluyen cuadros visuales relacionados con la temática del juego.

Parámetros:

- glm::mat4& model: matriz para transformaciones del modelo.
- GLuint uniformModel: localización del uniforme model en el shader.
- std::vector<Model*>& objetosDardos: punteros a modelos del entorno del juego de dardos.

- **topos(…)**

Encargada de renderizar el entorno del juego "golpea al topo". Dibuja objetos como la mesa, tubos de aparición, personajes emergentes (como plantas carnívoras), y elementos visuales adicionales.

Parámetros:

- glm::mat4& model: matriz de transformación.
- GLuint uniformModel: ubicación del uniforme en el shader activo.
- std::vector<Model*>& objetosTopos: modelos del puesto interactivo de topos.

- **ambientacion(…)**

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 Snoopy, Charlie Brown, Bowser, Donkey Kong, 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.

- **actualizarDardos(float velocidad)**

Actualiza la posición del dardo en vuelo, simulando su trayectoria hacia el tablero. Aumenta el realismo del lanzamiento al moverlo cuadro por cuadro.

Parámetro:

- velocidad: tiempo entre cuadros, ajusta la animación del dardo.

- **actualizarTopos(float velocidad)**

Anima el movimiento de los topos o figuras emergentes en el juego "golpea al topo". Alterna entre subir y bajar con base en tiempo.

Parámetro:

- velocidad: regula la frecuencia de aparición de los topos.

- **lanzarDardo()**

Dispara el dardo hacia el tablero, iniciando una animación de vuelo lineal o parabólica. Se ejecuta al presionar la tecla I.

- **activaTopos()**

Activa la aparición de los topos del juego, iniciando su movimiento vertical hacia arriba para ser golpeados. Se asocia a la tecla P

Archivo Topos

- **void topos(glm::mat4 model, GLuint uniformModel, std::vector<Model*> objetosTopos)**

Renderiza todos los elementos del juego de topos, incluyendo el stand principal y las máquinas de donde emergen los topos. Usa múltiples funciones renderMaquina para colocarlas en diferentes posiciones.

- **void renderStand6(glm::mat4 model, GLuint uniformModel, Model& stand, Model& cuadro, glm::vec3 posicion)**

Dibuja la estructura principal del stand y un cuadro decorativo. Realiza transformaciones de posición, rotación y escala sobre los modelos.

- **void renderMaquina(...), renderMaquina2(...), renderMaquina3(...)**

Renderizan variantes del mecanismo desde donde salen los topos. Incluyen una mesa, varios tubos y plantas. renderMaquina3 incluye animación vertical, rotación y escalado dinámico para el quinto topo.

- **void activarTopos()**

Inicializa las variables de animación de los topos. Comienza activando el primer topo.

- **void actualizarTopos(float deltaTime)**

Actualiza las animaciones de los topos. Controla la subida/bajada, rotación y aparición secuencial. El último topo (topo 5) se escala al final de la animación.

Variables importantes para la animación:

- traslacionTopos[5]: Controla el desplazamiento vertical de cada topo individual. Aumenta al subir, disminuye al bajar.
- rotacionTopos[5]: Define el ángulo de rotación de cada topo sobre el eje Y mientras está activo.

- toposSubiendo[5]: Indica si un topo está en fase ascendente (true) o descendente (false).
- animacionActivaTopos[5]: Activa o desactiva la animación de cada topo individual.
- topoActual: Índice del topo que se encuentra activo actualmente en la secuencia.
- delayTopos: Tiempo de espera entre la activación de un topo y el siguiente.
- tiempoAcumulado: Acumulador de tiempo que, al superar delayTopos, activa el siguiente topo.
- escalaTopo5: Escala aplicada al quinto topo. Se reduce progresivamente hasta desaparecer.
- topo5Escalado: Activa la fase en la que el quinto topo comienza a escalarse.
- temporizadorTopo5: Controla el tiempo en que el quinto topo permanece invisible antes de reaparecer.

Archivo Dardos

- **void dardos(glm::mat4 model, GLuint uniformModel, std::vector<Model*> objetosDardos)**

Renderiza el juego de dardos: stand, globos y múltiples dardos (uno animado).

- **void renderDardos(...)**

Dibuja una fila de dardos fijos y un dardo animado que avanza hacia los globos con rotación.

- **void renderGlobos(...)**

Muestra globos organizados sobre bases. El último globo se escala al ser impactado.

- **void actualizarDardos(float deltaTime)**

Gestiona la animación del lanzamiento del dardo:

- **void lanzarDardo()**

Reinicia las variables para comenzar un nuevo lanzamiento.

Variables importantes para la animación:

- animacionDardo: Representa el progreso de la animación del dardo en el tiempo. Controla su posición y rotación.
- lanzandoDardo: Activa la secuencia de animación del dardo. Solo si está en true se actualizan los valores.
- globoImpactado: Marca el momento en que el dardo ha alcanzado el globo. Da paso a la fase de reducción.
- dardoDetenido: Detiene el avance del dardo una vez ha impactado.
- escalaGlobo: Escala del globo afectado. Se reduce desde 1.0f hasta 0.0f para simular que estalla.
- tiempoImpactoDardo: Cronómetro que avanza desde el impacto y regula cuánto tiempo permanece el globo desaparecido.

Archivo ZonaFotos

Renderiza y anima una zona fotográfica con varios personajes interactivos y en movimiento.

Woodstock

Woodstock oscila verticalmente imitando un aleteo suave. Las alas también rotan alternadamente (ambas animadas por sin(vuelaWoodstock * 5.0f)).

- vuelaWoodstock: ángulo base para calcular movimiento vertical.

Yoshi

Se mueve de un lado a otro con salto ondulado mientras agita las piernas, gira 360º al cambiar de dirección y hace una animación especial con brazos cuando cambia de sentido.

Variables importantes para la animación:

- angSaltoYoshi: ángulo para animar salto en eje Y.
- saltoYoshi: resultado del sin que define la elevación vertical.
- frecuenciaSaltoYoshi, amplitudSaltoYoshi: parámetros del salto.
- angPatasYoshi: animación de piernas basada en seno.
- movYoshi: desplazamiento horizontal sobre eje X.
- avanzaYoshi: dirección del movimiento.
- movBrazo, tiempoBrazo, angBrazoYoshi, angGiroYoshi: control de animación rotatoria de brazos y giro total cuando alcanza extremos.

Archivo NPCs

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

Este archivo únicamente posiciona personajes estáticos no jugables (NPCs) en diferentes zonas de la ferio. Estos personajes no están animados, solo se renderizan con transformaciones de traslación, rotación y escala.

Parámetros:

- model: matriz base de transformación.
- uniformModel: identificador del uniforme de modelo.
- personajesNPCs: lista de modelos 3D de personajes.

Archivo Tierra

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

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

Parámetros:

- 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.

Variables importantes:

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

Archivo Window

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

Son los constructores de la clase. 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.

Parámetros importantes:

- 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.

Archivo Ambientacion

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

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

Archivo Comida

- **void comida(glm::mat4 model, GLuint uniformModel, std::vector<Model*> objetosComida)**

Renderiza los puestos de comida.

- objetosComida[0]: palomitas

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.

- **CrearDado()**

Construye y almacena la malla de un cubo con texturas aplicadas a cada cara, simulando un dado. Se guarda 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.

Conclusión

El desarrollo de un proyecto de este calibre significó poner en práctica todos los conocimientos y técnicas aprendidas a lo largo del semestre en la materia de Computación Gráfica e Interacción Humano-Computadora, tanto de la clase de teoría, como del laboratorio.

Se consiguió general un escenario virtual completo inspirado en una feria tematizada con elementos de los universos de *Mario Bros*, *Snoopy* y *Gumball*, desde una skybox que cambia cuando “se hace de noche”; un gran número de modelos texturizados con el que se armaron los puestos de juegos interactivos, locales de comida y ambientación, tomando en cuenta que se animaron todos los juegos y se tienen a los personajes principales en movimiento todo el tiempo; un sistema de iluminación automático que simula un ciclo de día y noche; hasta un conjunto de 3 cámaras entre las cuales se puede alternar en cualquier momento.

Aunque se cumplieron de manera satisfactoria todos los requerimientos que debía contener este trabajo, creo que puede haber algunas cosas que se podrían corregir o cambiar, que desafortunadamente por falta de tiempo, no se pudieron atender como gustaría, como es el tema de las animaciones.

En este proyecto pudimos aplicar lo antes mencionado siguiendo un enfoque creativo y fundamentado en el trabajo en equipo. También quiero mencionar que fue una gran oportunidad para aprender a utilizar un sistema de almacenamiento remoto como GitHub, y trabajar de una forma en que podría suceder en el futuro en la vida profesional al colaborar en un equipo.

Links Modelos:

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

Casa (cabinas de juegos): https://free3d.com/3d-model/old-farm-house-91130.html

Woodstock: https://sketchfab.com/3d-models/wii-u-the-peanuts-movie-woodstock-d7a8747fc83f49e194dd9ccde0cc2d49

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

Piraña: https://www.cgtrader.com/search?keywords=mario+piraña

Donkey Kong: https://www.models-resource.com/wii/mariosupersluggers/model/58606/

Yoshi: https://www.models-resource.com/wii/mariosupersluggers/model/58610/

Wario: https://www.models-resource.com/wii/mariosupersluggers/model/58612/

Bowser: https://www.models-resource.com/wii/mariosupersluggers/model/58611/

Árboles: https://free3d.com/es/modelo-3d/free-tree-pack-109238.html

Palomitas: https://www.turbosquid.com/es/3d-models/3d-popcorn-corn-pop-model-1645559

Base globos: https://sketchfab.com/3d-models/ocf-easel-b124839ed52a41379ba4fe973eb05fde

Dardo:https://sketchfab.com/3d-models/dart-f4eadd9979de4378aa523228c992d5eb#download

Globo: https://free3d.com/3d-model/balloon-darts-v1--421125.html

Mesa topos: https://sketchfab.com/3d-models/whack-em-allwhack-a-mole-machine-66e181857962436ab4b7a806618cab61

Fuente: https://www.turbosquid.com/3d-models/3d-model-water-fountain-1387533

Links Texturas:

Agua: https://www.shutterstock.com/es/image-photo/dark-blue-water-texture-close-2563388969

Cuadro gumball: https://www.youtube.com/watch?v=Mp_4noPC7Qc

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

Mesa topos:
https://www.youtube.com/watch?v=OWn0XgBJuLI&ab_channel=Badabun

Palomitas: https://es.vecteezy.com/foto/5273536-palomitas-de-maiz-sobre-un-fondo-amarillo-como-imagen-de-fondo

Piedra: https://unsplash.com/es/fotos/un-primer-plano-de-una-roca-con-pequenas-rocas-en-ella-wuoRhml_xNU