UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE INGENIERÍA

DIVISIÓN DE INGENIERÍA ELÉCTRICA

INGENIERÍA EN COMPUTACIÓN

COMPUTACIÓN GRÁFICA e INTERACCIÓN HUMANO COMPUTADORA

MANUAL TÉCNICO

Nº de Cuenta: 319111347

GRUPO DE TEORÍA: 05

SEMESTRE 2025-2

FECHA DE ENTREGA LÍMITE: 20/05/2025

CALIFICACIÓN: _____

MANUAL TÉCNICO

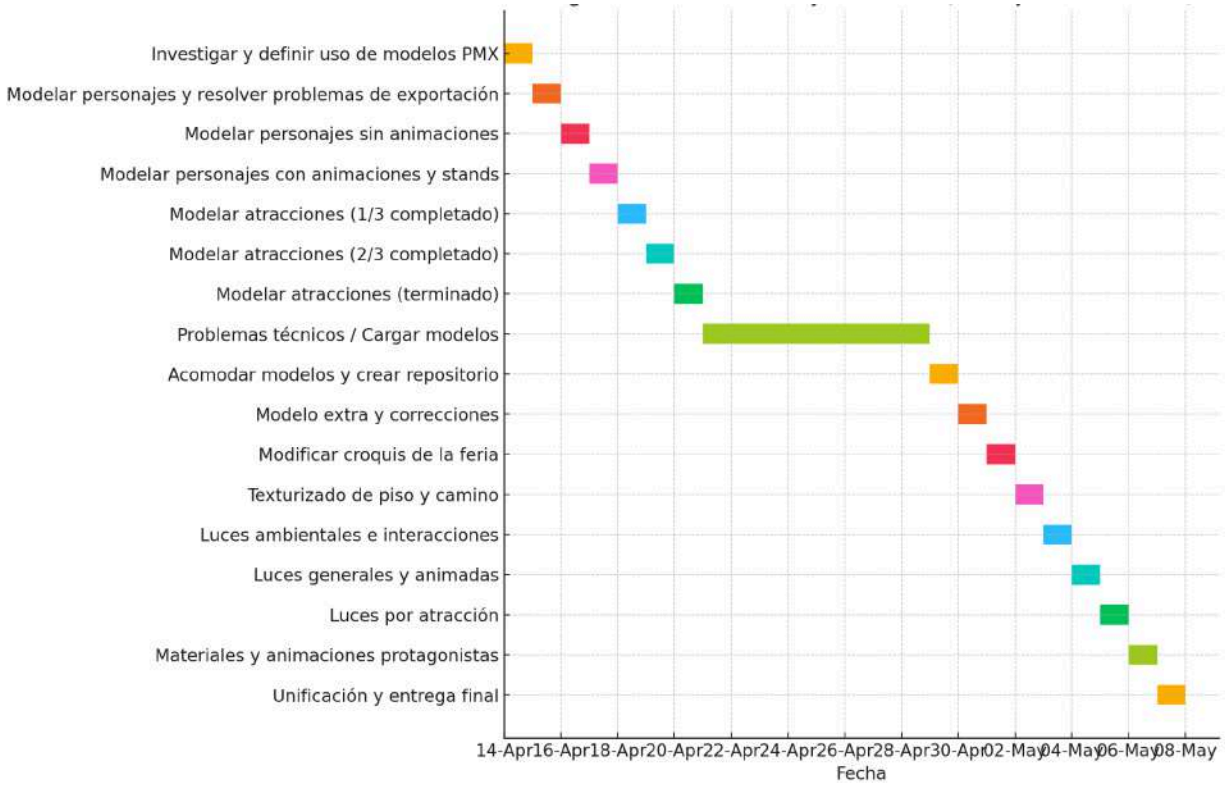
DESCRIPCIÓN Y OBJETIVO

Consta de un escenario el cual girará en torno del tema de: “Feria de juegos de destreza” es decir, que se creará un escenario el cual tendrá como temática principal el ser una feria que puede ser recorrida de forma libre o puede enfocarse la cámara en las diferentes atracciones cuando se interactúe con ellas.

ALCANCE

- **Escenario temático completo:** El entorno será montado como una feria realista, con caminos, decoraciones, iluminación, y ambientación adecuada al tema.
- **Recorrido en tercera persona:** Se implementará un sistema de cámara en tercera persona que seguirá a un personaje mientras recorre libremente el escenario.
- **Interacción con stands:** Al presionar ciertas teclas, se activará una funcionalidad que reencuadra la cámara automáticamente sobre ese stand para enfocarlo y que se realice una animacion correspondiente a dicho stand
- **Iluminación y ambiente:** Se incluirá un sistema de iluminación con fuentes de luz, que permitirá destacar zonas clave del escenario.
- **Animaciones en NPC:** Algunos NPCS tendran animaciones que no dependan de las interacciones, por lo que todo el tiempo estaran realizando dichos movimientos.
- **Ambientacion:** La feria estara ambientada en el estilo de los videojuegos “Honkai: Star Rail”, “Genshin Impact” y “Tears of Themis”

DIAGRAMA DE GANTT:



ANÁLISIS DE COSTOS:

1. Costo para el equipo (inversión interna)

Se estimó que el desarrollo completo del proyecto tomó un total de 150 horas. Estas horas incluyen tareas como:

- Modelado de objetos en Blender
- Integración con OpenGL
- Programación de la cámara en tercera persona
- Configuración de iluminación y sombreado
- Interacción básica en tiempo real
- Pruebas, depuración y ajustes finales

Costo por hora estimado: $150 MXN/hora (considerando conocimientos técnicos, software, energía, equipo de cómputo y conexión a internet).

Costo base=150 horas×150 MXN/hora=$22,500 MXN

2. Costos indirectos

Se considera un 10% adicional por costos indirectos como:

- Consumo eléctrico
- Desgaste de equipo
- Internet y otros servicios auxiliares

Costo con indirectos=22,500 MXN×1.10=$24,750 MXN

3. Precio de venta

Para establecer un precio de venta justo y sustentable, se considera un margen de ganancia del 25%, que contempla:

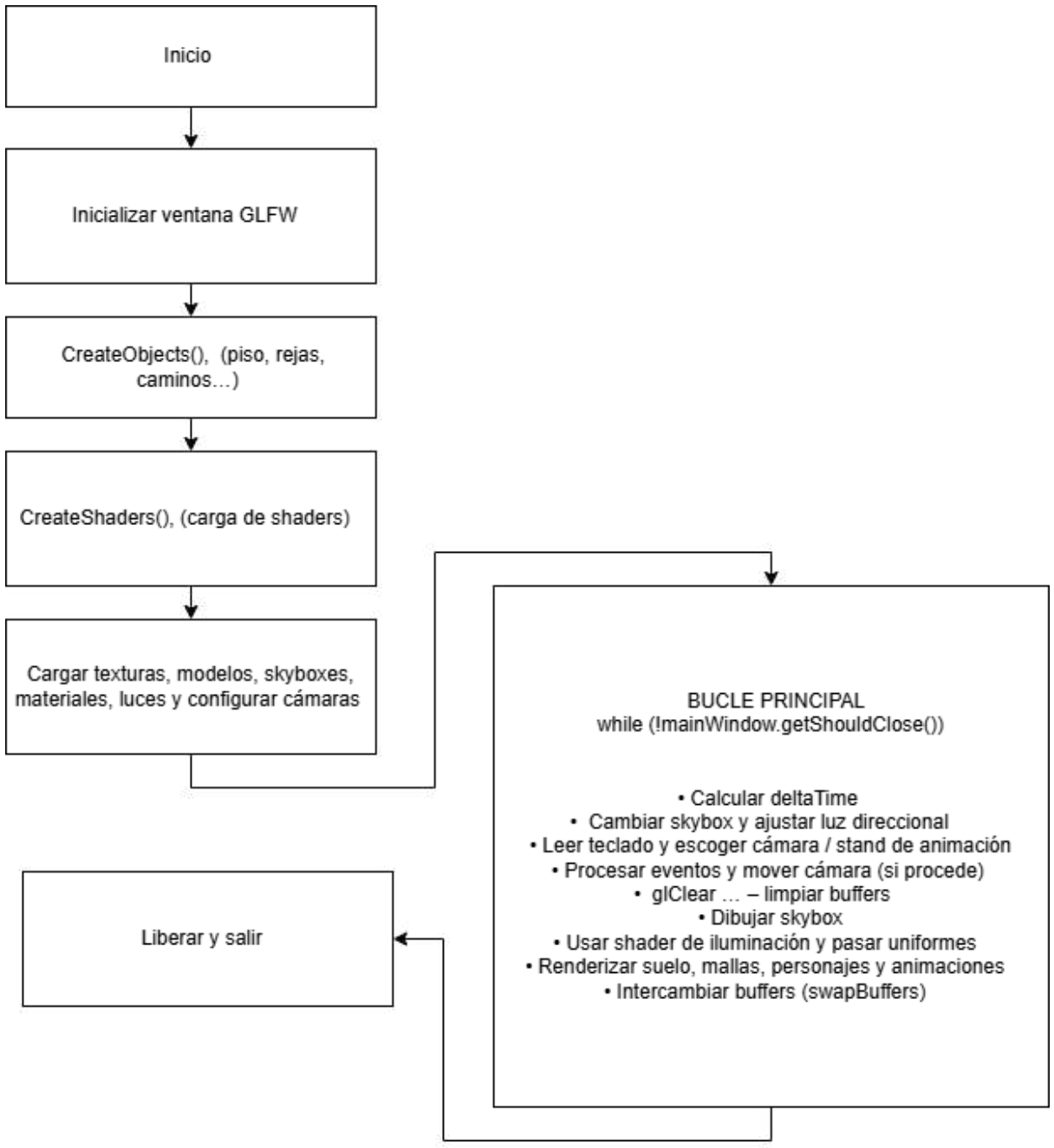
- Valor del conocimiento especializado
- Tiempo invertido en investigación y pruebas

Precio final=24,750 MXN×1.25=$30,937.50 MXN

El precio de $31,000 MXN está fundamentado en:

- El tiempo real invertido por el equipo (150 horas).
- La especialización técnica necesaria (programación, diseño 3D, OpenGL).
- El uso de herramientas y recursos de cómputo.

DIAGRAMA DE FLUJO



DICCIONARIO DE FUNCIONES:

Model.cpp

- **Model::LoadModel**
  Carga un modelo 3D desde un archivo usando Assimp, triangula y procesa vértices, luego carga nodos y materiales.
- **Model::ClearModel**
  Libera la memoria de todas las mallas y texturas almacenadas.
- **Model::RenderModel**
  Renderiza todas las mallas del modelo aplicando sus respectivas texturas.
- **Model::LoadNode**
  Recorre recursivamente los nodos del modelo y carga cada malla.
- **Model::LoadMesh**
  Extrae vértices, normales, texturas e índices de una malla de Assimp y crea una malla OpenGL.
- **Model::LoadMaterials**
  Carga y asigna las texturas correspondientes a los materiales del modelo. Usa una textura por defecto si no se encuentra una.

PointLight.cpp

- **PointLight::PointLight (constructor por defecto)**
  Inicializa una luz puntual con valores neutros.
- **PointLight::PointLight (con parámetros)**
  Inicializa una luz puntual con color, intensidades y posición personalizadas.
- **PointLight::UseLight**
  Envia los valores de la luz (color, intensidad, posición y atenuación) a los shaders.
- **PointLight::SetLight**
  Cambia el color y la posición de la luz.
- **PointLight::SetIntensity**
  Cambia las intensidades ambiental y difusa de la luz.

SpotLight.cpp

- **SpotLight::SpotLight (constructor por defecto)**
  Inicializa una luz tipo spotlight con dirección hacia abajo y sin ángulo definido.
- **SpotLight::SpotLight (con parámetros)**
  Inicializa una spotlight con color, intensidades, dirección, posición y borde de corte.
- **SpotLight::UseLight**
  Envía todos los parámetros necesarios del spotlight al shader, incluyendo dirección y ángulo.
- **SpotLight::SetFlash**
  Cambia la posición y dirección de la luz (útil para simular una linterna en movimiento).
- **SpotLight::SetColor**
  Cambia el color de la luz.
- **SpotLight::SetPos**
  Cambia la posición de la luz.

Window.cpp

- **Window::Window (constructores)**
  Inicializan la ventana con dimensiones por defecto o personalizadas, y resetean estados de teclas y variables de animación.
- **Window::Initialise**
  Inicializa GLFW, crea la ventana, configura el contexto de OpenGL y activa GLEW y el z-buffer.
- **Window::createCallbacks**
  Establece los callbacks para manejar entrada del teclado.

MODELOS:

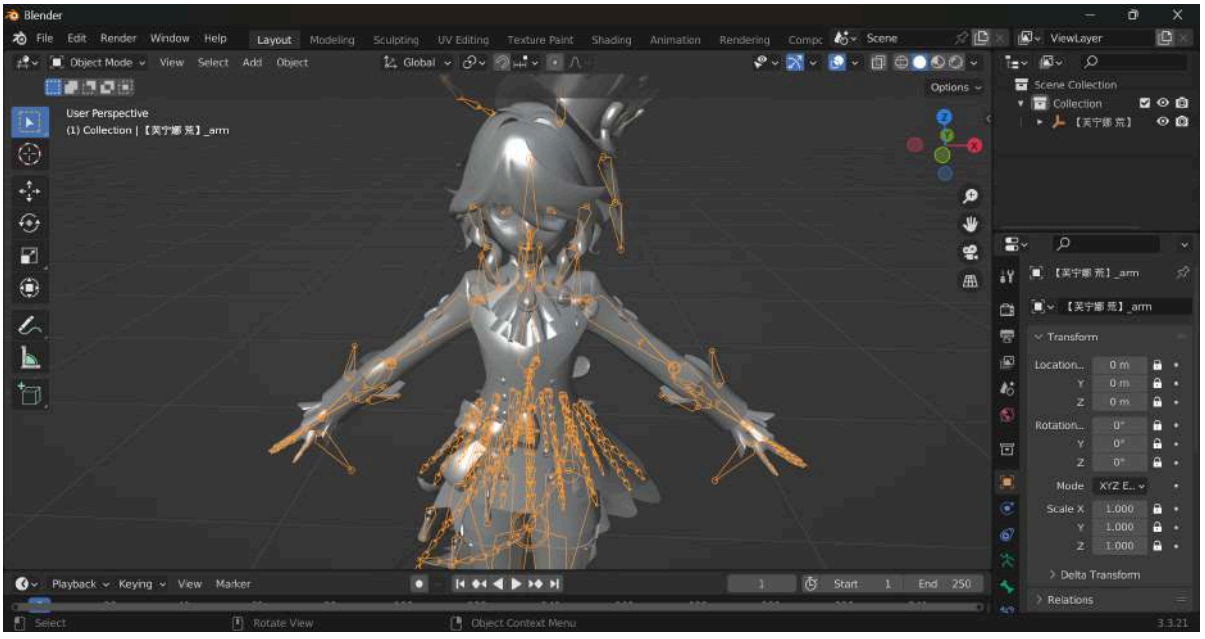
Los modelos elegidos de los personajes fueron encontrados en formato .pmx, lo cual, aunque dificultó el trabajó pues no se sabía trabajar con dicho formato, hizo que resultara más sencillo posar a los personajes pues no había necesidad de separarlos para moverlos ya que se utilizaron los “huesos” que tenían los modelos. Se tuvieron de nuevo dificultades pues los modelos, sus texturas y sus partes están en chino, sin embargo, al final se consiguió un método para trabajarlos.

Se siguieron las instrucciones de un video para poder importarlos en Blender, utilizando la versión 3.3, con un añadido llamado MMD.

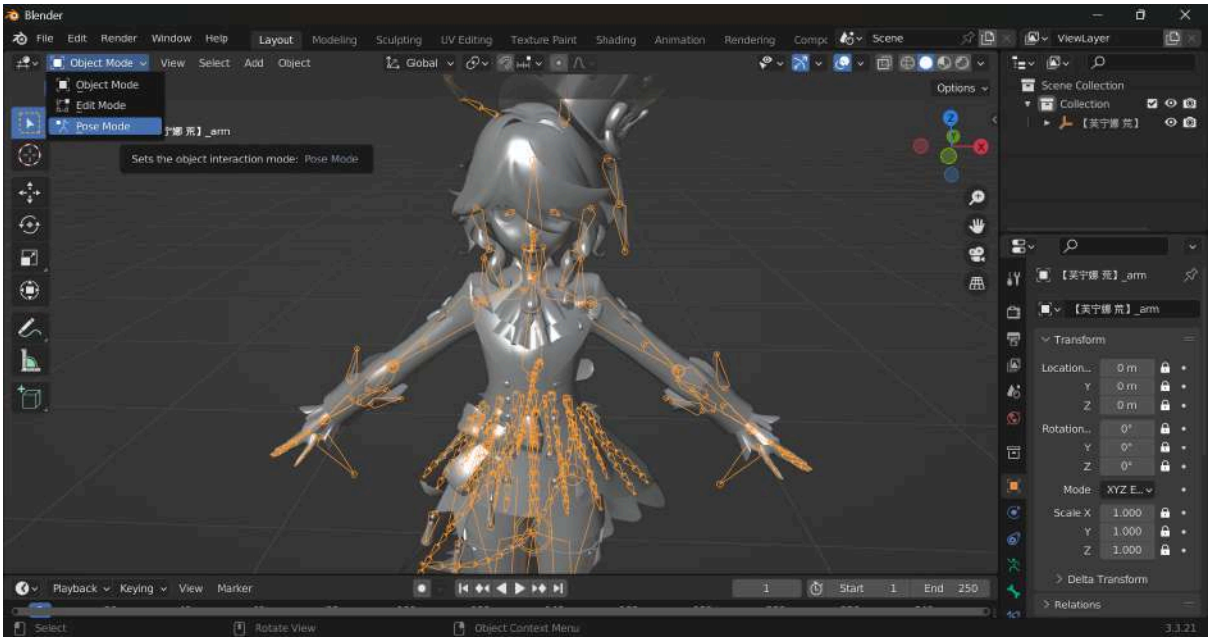
Enlace al video: https://www.youtube.com/watch?app=desktop&v=OgS31Q9KkDI

Enlace al repositorio donde se descargó MMD: https://github.com/MMD-Blender/blender_mmd_tools/tree/main

Posteriormente, se abren en Blender, en donde se les puede modificar de posición a los modelos gracias a MMD y que ya vienen diseñados para dicho propósito.

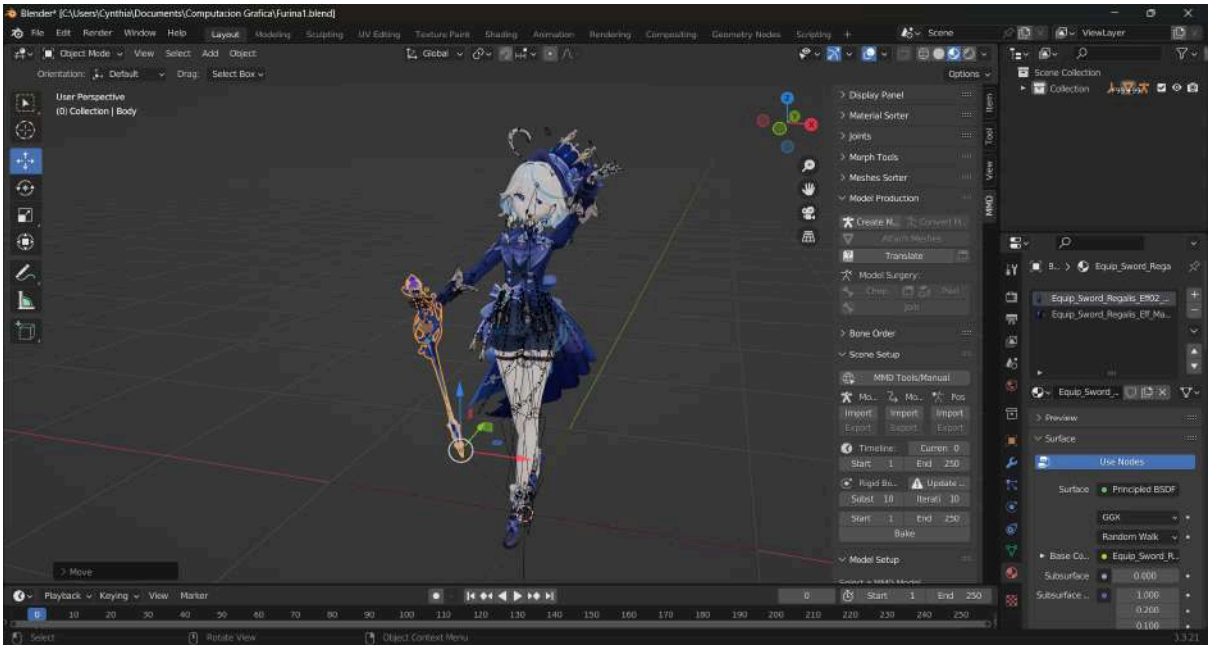


Para hacerlo, se selecciona el esqueleto, y se cambia el modo a “Pose Mode”:

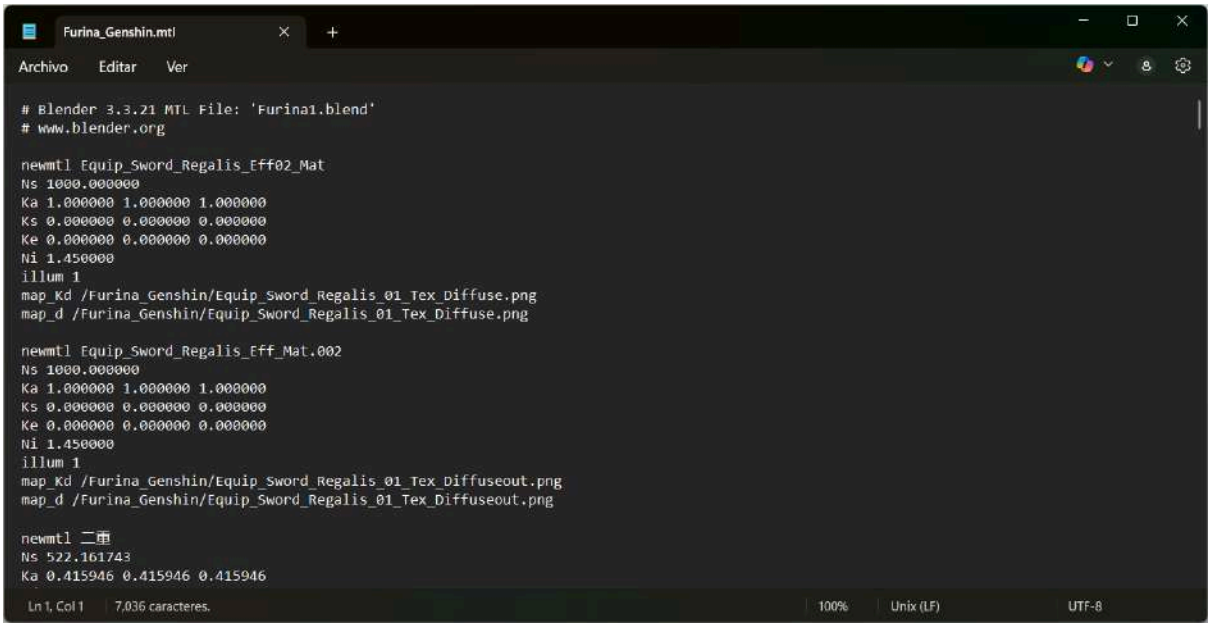


En este modo se puede seleccionar un hueso y rotarlo utilizando la letra G para indicar que se moverá, y posteriormente las letras X,Y,Z para rotarlo en dicho eje. Se selecciona fuera del hueso para terminar el movimiento.

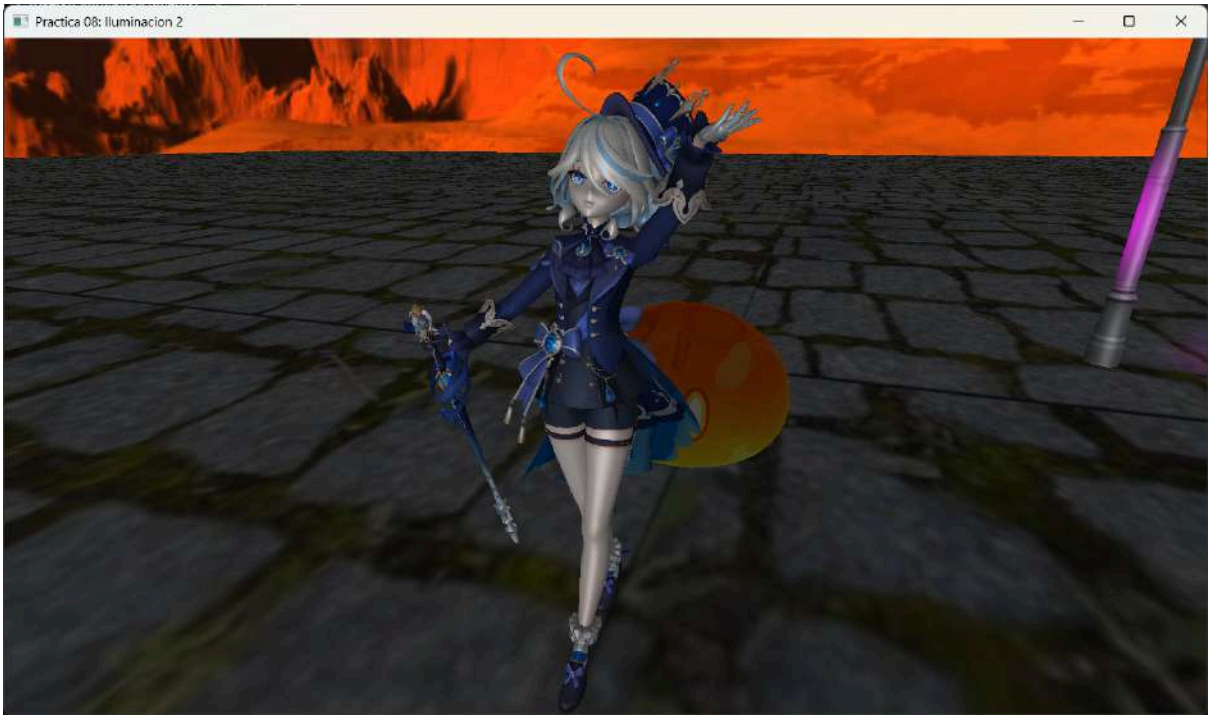
Una vez determinada la posición, se continúa agregando los objetos extra que acompañarán al modelo.



Al finalizar se importa como .obj, y ya que este modelo no se moverá, todo será un único modelo. Después se agregan los valores de las texturas de forma empírica, ya que al transformarlo de .pmx a .obj no se colocan texturas, esto se realiza en el .mtl final. También es importante mencionar que algunos nombres de texturas están en chino, y el programa no los lee bien, por lo que hay que modificarlos. Otra manera es ir texturizando a través de 3ds Max, y posteriormente exportar.



Al finalizar se tiene el modelo visualizable en el programa:



Ya que los personajes son sacados de modelos oficiales y fan-made que están apegados a los oficiales, todos los personajes de los 3 universos ya estaban igualmente escalados en altura (aunque fueron escalados un x5 para que se viera del tamaño querido en el proyecto), y es por ello los tomamos como referencia para todo lo demás de modelos.

Los stands se realizaron con modelos obtenidos de internet y fueron acomodados, retexturizados en caso de ser necesario, escalados y ubicados antes de ser exportados (en los casos que ya se tenia al modelo del personaje y una idea de donde iba a ir el stand para ahorrar transformaciones).

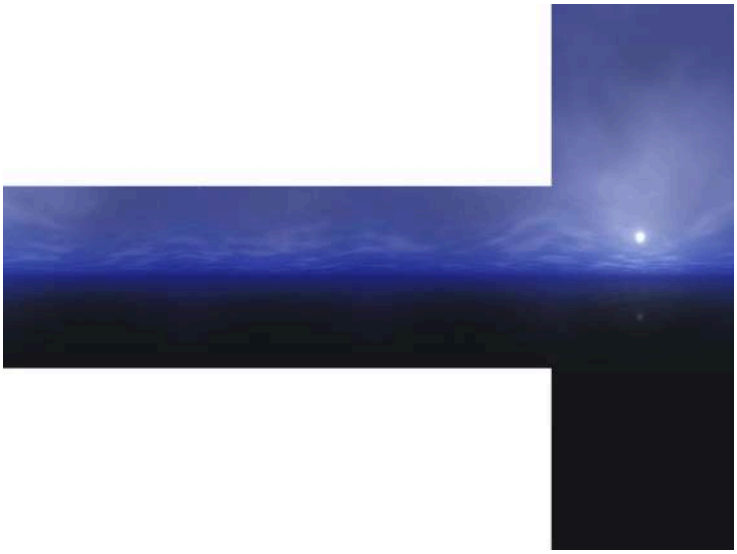
Skyboxes y ciclo de día y noche del skybox:

Día:

https://www.pngwing.com/en/free-png-dcabk/download

Atardecer:

Noche:



https://gamebanana.com/mods/7921

Para usarlos recorté cada pedazo de las imágenes y les puse sus respectivos nombres para poder ubicarlas. Luego, con el uso de GIMP les quité el canal alpha y las escalé a 512x512 cada una. Finalmente las mandé a llamar en el programa.

```
// ------------------------------------
// -----------SKYBOX TEXTURAS-----------
// ------------------------------------

std::vector<std::string> skyboxFaces; //Se crea un vector con las texturas que componen al skybox de dia
//Todo esto para que dentro del while se vayan cambiando con los skybox ya existentes creados aquí
skyboxFaces.push_back("Textures/Skybox/Nubes_Derecha.tga");
skyboxFaces.push_back("Textures/Skybox/Nubes_Izq.tga");
skyboxFaces.push_back("Textures/Skybox/Nubes_Abajo.tga");
skyboxFaces.push_back("Textures/Skybox/Nubes_Arriba.tga");
skyboxFaces.push_back("Textures/Skybox/Nubes_Detras.tga");
skyboxFaces.push_back("Textures/Skybox/Nubes_Enfrente.tga");

std::vector<std::string> skyboxFacesAtard; //Se crea un vector con las texturas que componen al skybox de atardecer
skyboxFacesAtard.push_back("Textures/Skybox/Atardecer_Derecha.tga");
skyboxFacesAtard.push_back("Textures/Skybox/Atardecer_Izq.tga");
skyboxFacesAtard.push_back("Textures/Skybox/Atardecer_Abajo.tga");
skyboxFacesAtard.push_back("Textures/Skybox/Atardecer_Arriba.tga");
skyboxFacesAtard.push_back("Textures/Skybox/Atardecer_Detras.tga");
skyboxFacesAtard.push_back("Textures/Skybox/Atardecer_Enfrente.tga");

std::vector<std::string> skyboxFacesNoche; //Se crea un vector con las texturas que componen al skybox de noche
skyboxFacesNoche.push_back("Textures/Skybox/Night_Derecha.tga");
skyboxFacesNoche.push_back("Textures/Skybox/Night_Izq.tga");
skyboxFacesNoche.push_back("Textures/Skybox/Night_Abajo.tga ");
skyboxFacesNoche.push_back("Textures/Skybox/Night_Arriba.tga");
skyboxFacesNoche.push_back("Textures/Skybox/Night_Detras.tga");
skyboxFacesNoche.push_back("Textures/Skybox/Night_Enfrente.tga");
```

Tiene que estar antes del while para que sean variables globales.

```
GLfloat now = glfwGetTime();
deltaTime = now - lastTime;
deltaTime += (now - lastTime) / limitFPS;
lastTime = now;

//Permite manipular correctamente deltaTime para las animaciones ciclicas
if (contadorInicioPrograma == 0) contadorInicioPrograma++;

// ----------------------------------------
// ----------------SKYBOX----------------
// ----------------------------------------

if (skyCount == 0) { //Amanecer
    skybox = Skybox(skyboxFaces);  //Skybox de dia

    //Recibe Color nuevo RGC, dirección nueva, ambientIntensity, diffuseIntensity
    mainLight.SetLight(glm::vec3(0.9843f, 0.9843f, 0.8980f), glm::vec3(0.0f, 0.0f, 1.0f), 0.5f, 0.6f);
}
else if (skyCount == 1000) {//Medio dia

    //Recibe Color nuevo RGC, dirección nueva, ambientIntensity, diffuseIntensity
    mainLight.SetLight(glm::vec3(1.0f, 0.9647f, 0.9333f), glm::vec3(0.0f, -1.0f, 0.0f), 0.43f, 0.37f);
}

else if (skyCount == 2000) {//Atardecer
    skybox = Skybox(skyboxFacesAtard);  //Skybox de atardecer

    //Recibe Color nuevo RGC, dirección nueva, ambientIntensity, diffuseIntensity
    mainLight.SetLight(glm::vec3(0.9450f, 0.6274f, 0.4745f), glm::vec3(0.0f, 0.0f, -1.0f), 0.45f, 0.6f);
}
else if (skyCount == 3000) { //Noche
    skybox = Skybox(skyboxFacesNoche);  //Skybox de noche

    //Recibe Color nuevo RGC, dirección nueva, ambientIntensity, diffuseIntensity
    mainLight.SetLight(glm::vec3(0.9176f, 0.9411f, 0.9686f), glm::vec3(0.0f, 0.0f, 1.0f), 0.5f, 0.4f);
}
else if (skyCount == 6000) { //Termina la noche
    skyCount = -1; //Se le resta uno porque abajo se suma 1 y al sumarse 1 ya no queda en e para el reinicio
}

skyCount += 1; //La variable contadora aumenta en uno con cada ciclo del while que es lo que lleva la cuenta del tiempo
```

Los ifs van dentro del ciclo while. Dentro del código se comentó qué se hizo y qué hace cada cosa, sin embargo, de manera resumida, revisa con una variable contadora cuando mandar a llamar al skybox de día, atardecer o noche.

RECORRIDO (CÁMARAS)

Para las cámaras se implementó un sistema en donde se crean las 4 cámaras (una extra para la cámara libre) con la posición inicial de cada una iniciada. La cámara aérea se realizó moviendo el valor de la dirección de la mira a 90 (en frente) y -90 (hacia abajo).

```
// ----------------------------------
// -------------CÁMARAS-------------
// ----------------------------------

//1 Posición inicial; 2 No relevante; 3 Rotación hacia la izquierda o derecha en grados (+ o -)
//4 Rotación hacia arriba o abajo en grados (+ o -); 5 y 6 Velocidades para la camara

//Camara Blade Avatar (F)
camera = Camera(glm::vec3(83.0f, 11.5f, 140.0f), glm::vec3(0.0f, 1.0f, 0.0f), -90.0f, 0.0f, 0.3f, 0.5f);

//Camara aerea (G)
camera2 = Camera(glm::vec3(0.0f, 270.0f, -10.0f), glm::vec3(0.0f, 1.0f, 0.0f), 90.0f, -90.0f, 0.3f, 0.5f);

//Camara stands (H,J,K,L,M,N)
camera3 = Camera(glm::vec3(0.0f, 0.0f, 0.0f), glm::vec3(0.0f, 1.0f, 0.0f), -60.0f, 0.0f, 0.3f, 0.5f);

//Cámara libre (Q)
cameraLibre = Camera(glm::vec3(83.0f, 11.5f, 140.0f), glm::vec3(0.0f, 1.0f, 0.0f), -90.0f, 0.0f, 0.3f, 0.5f);
```

Luego, dentro del while se encuentran una serie de ifs que delimitan si se puede mover de cámara con la tecla o no (banderaCamara) para que cuando se esté en la cámara del stand no se pueda salir de la animación ni de la cámara con alguna tecla o el mouse. Dentro de ese if hay otros que delimitan qué cámara usar cuando se aprieta una cierta tecla mediante el uso de las articulaciones. Para la cámara que sigue al ávatar y la cámara libre no se sobreescribió la cámara como en los stands pues la cámara tiene que seguir mirando a donde se quedó y no volver a ser iniciada. Por otro lado, la cámara de los stands es una y se le van sobreescribiendo los valores iniciales para que siempre mire al mismo lugar.

En los stands se cambia de valor la variable camaraAnimacion para que se le pase a las luces y a las animaciones para que sepan qué animación se debe de iniciar.

```
// ----------------------------------------
// ---------------CÁMARAS-----------------
// ----------------------------------------

if (banderaCamara == 0) {

    if (mainWindow.getarticulacion1() == 1.0) { //Vista Blade
        banderaCamaraMovimiento = 0;
    }
    else if (mainWindow.getarticulacion2() == 1.0) { //Vista aerea
        camera3 = camera2;
        banderaCamaraMovimiento = 1;
    }
    else if (mainWindow.getarticulacion3() == 1.0) { //Stand hacha
        camera3 = Camera(glm::vec3(-106.0f, 8.0f, -78.0f), glm::vec3(0.0f, 1.0f, 0.0f), 180.0f, 0.0f, 0.3f, 0.5f);
        banderaCamaraMovimiento = 1;
        camaraAnimacion = 1;
    }
    else if (mainWindow.getarticulacion4() == 1.0) { //Stand boliche
        camera3 = Camera(glm::vec3(-52.0f, 11.0f, 62.0f), glm::vec3(0.0f, 1.0f, 0.0f), 90.0f, -15.0f, 0.3f, 0.5f);
        banderaCamaraMovimiento = 1;
        camaraAnimacion = 2;
    }
    else if (mainWindow.getarticulacion5() == 1.0) { //Stand dados
        camera3 = Camera(glm::vec3(-102.0f, 9.0f, -35.0f), glm::vec3(0.0f, 1.0f, 0.0f), 180.0f, -30.0f, 0.3f, 0.5f);
        banderaCamaraMovimiento = 1;
        camaraAnimacion = 3;
    }
    else if (mainWindow.getarticulacion6() == 1.0) { //Stand bateo
        camera3 = Camera(glm::vec3(-03.0f, 9.0f, 93.0f), glm::vec3(0.0f, 1.0f, 0.0f), 90.0f, 0.0f, 0.3f, 0.5f);
        banderaCamaraMovimiento = 1;
        camaraAnimacion = 4;
    }
    else if (mainWindow.getarticulacion7() == 1.0) { //Stand dardos
        camera3 = Camera(glm::vec3(91.5f, 8.5f, -26.0f), glm::vec3(0.0f, 1.0f, 0.0f), 0.0f, -5.0f, 0.3f, 0.5f);
        banderaCamaraMovimiento = 1;
        camaraAnimacion = 5;
    }
```

```
else if (mainWindow.getarticulacion8() == 1.0) { //Stand topo
    camera3 = Camera(glm::vec3(61.0f, 9.0f, 62.5f), glm::vec3(0.0f, 1.0f, 0.0f), 90.0f, -30.0f, 0.3f, 0.5f);
    banderaCamaraMovimiento = 1;
    camaraAnimacion = 6;
}
else if (mainWindow.getarticulacion11() == 1.0) { //Camara Libre (EXTRA)
    banderaCamaraMovimiento = 2;
}
```

Por último, con otros ifs se limpia la ventana y se “crea” la ventana. Para la cámara libre y la del ávatar se dejaron intacto el movimiento de teclado para la primera y el del mouse para las dos.

```
//Para  clear Window según se ocupe o no que se pueda mover la cámara
if (banderaCamaraMovimiento == 0) {
    //CAMARA 1 (VISTA BLADE CON MOVIMIENTO)
    glfwPollEvents();
    //camera.keyControl(mainWindow.getsKeys(), deltaTime);
    camera.mouseControl(mainWindow.getXChange(), mainWindow.getYChange());

    // Clear the window
    glClearColor(0.0f, 0.0f, 0.0f, 1.0f);
    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
    skybox.DrawSkybox(camera.calculateViewMatrix(), projection);
    shaderList[0].UseShader();
    uniformModel = shaderList[0].GetModelLocation();
    uniformProjection = shaderList[0].GetProjectionLocation();
    uniformView = shaderList[0].GetViewLocation();
    uniformEyePosition = shaderList[0].GetEyePositionLocation();
    uniformColor = shaderList[0].getColorLocation();

    //información en el shader de intensidad especular y brillo
    uniformSpecularIntensity = shaderList[0].GetSpecularIntensityLocation();
    uniformShininess = shaderList[0].GetShininessLocation();

    glUniformMatrix4fv(uniformProjection, 1, GL_FALSE, glm::value_ptr(projection));
    glUniformMatrix4fv(uniformView, 1, GL_FALSE, glm::value_ptr(camera.calculateViewMatrix()));
    glUniform3f(uniformEyePosition, camera.getCameraPosition().x, camera.getCameraPosition().y, camera.getCameraPosition().z);
}
```

```
else if (banderaCamaraMovimiento == 2) {
    //CAMARA LIBRE (VISTA EXTRA CON MOVIMIENTO )
    glfwPollEvents();
    cameraLibre.keyControl(mainWindow.getsKeys(), deltaTime);
    cameraLibre.mouseControl(mainWindow.getXChange(), mainWindow.getYChange());

    // Clear the window
    glClearColor(0.0f, 0.0f, 0.0f, 1.0f);
    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
    skybox.DrawSkybox(cameraLibre.calculateViewMatrix(), projection);
    shaderList[0].UseShader();
    uniformModel = shaderList[0].GetModelLocation();
    uniformProjection = shaderList[0].GetProjectionLocation();
    uniformView = shaderList[0].GetViewLocation();
    uniformEyePosition = shaderList[0].GetEyePositionLocation();
    uniformColor = shaderList[0].getColorLocation();

    //información en el shader de intensidad especular y brillo
    uniformSpecularIntensity = shaderList[0].GetSpecularIntensityLocation();
    uniformShininess = shaderList[0].GetShininessLocation();

    glUniformMatrix4fv(uniformProjection, 1, GL_FALSE, glm::value_ptr(projection));
    glUniformMatrix4fv(uniformView, 1, GL_FALSE, glm::value_ptr(cameraLibre.calculateViewMatrix()));
    glUniform3f(uniformEyePosition, cameraLibre.getCameraPosition().x, cameraLibre.getCameraPosition().y, cameraLibre.getCameraPosition().z);
}
```

Para la cámara 3 de los stands se eliminaron dichas funcionalidades para que el usuario no pudiera mover la cámara ni con el teclado ni con el mouse y ésta se quedara fija a la posición donde fue fijada.

```
else {
    //CAMARA 2 Y 3 (VISTA AEREA Y CADA STAND, SIN MOVIMIENTO)
    glfwPollEvents();

    // Clear the window
    glClearColor(0.0f, 0.0f, 0.0f, 1.0f);
    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
    skybox.DrawSkybox(camera3.calculateViewMatrix(), projection);
    shaderList[0].UseShader();
    uniformModel = shaderList[0].GetModelLocation();
    uniformProjection = shaderList[0].GetProjectionLocation();
    uniformView = shaderList[0].GetViewLocation();
    uniformEyePosition = shaderList[0].GetEyePositionLocation();
    uniformColor = shaderList[0].getColorLocation();

    //información en el shader de intensidad especular y brillo
    uniformSpecularIntensity = shaderList[0].GetSpecularIntensityLocation();
    uniformShininess = shaderList[0].GetShininessLocation();

    glUniformMatrix4fv(uniformProjection, 1, GL_FALSE, glm::value_ptr(projection));
    glUniformMatrix4fv(uniformView, 1, GL_FALSE, glm::value_ptr(camera3.calculateViewMatrix()));
    glUniform3f(uniformEyePosition, camera3.getCameraPosition().x, camera3.getCameraPosition().y, camera3.getCameraPosition().z);
}
```

Cámara del avatar (Blade):

```
// 1. Dirección de la cámara (sin componente Y para que no suba/baje el personaje)
glm::vec3 cameraForward = glm::normalize(glm::vec3(camera.getCameraDirection().x, 0.0f, camera.getCameraDirection().z));
glm::vec3 right = glm::normalize(glm::cross(cameraForward, glm::vec3(0.0f, 1.0f, 0.0f))); // vector a la derecha

// 2. Rotación del personaje basada en la cámara
float bladeAngle = atan2(cameraForward.x, cameraForward.z); // Yaw en radianes

// 3. Movimiento si se presiona tecla
if (mainWindow.getAvanzaBladeW() == 1) {
    bladePosition += cameraForward * bladeSpeed * deltaTime;
}
if (mainWindow.getAvanzaBladeA() == 1) {
    bladePosition -= right * bladeSpeed * deltaTime;
}
if (mainWindow.getAvanzaBladeS() == 1) {
    bladePosition -= cameraForward * bladeSpeed * deltaTime;
}
if (mainWindow.getAvanzaBladeD() == 1) {
    bladePosition += right * bladeSpeed * deltaTime;
}

// 4. Construcción de la matriz del personaje
model = glm::mat4(1.0f);
model = glm::translate(model, bladePosition);
model = glm::rotate(model, bladeAngle, glm::vec3(0.0f, 1.0f, 0.0f)); // rotación en eje Y
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
Blade_Cuerpo.RenderModel();
modelaux = model;

// 5. Cámara que sigue al personaje desde atrás
glm::vec3 cameraOffset = -cameraForward * 10.0f + glm::vec3(0.0f, 5.0f, 0.0f); // detrás y arriba
glm::vec3 cameraPosition = bladePosition + cameraOffset;
camera.setPosition(cameraPosition);

glm::vec3 lookTarget = bladePosition + glm::vec3(0.0f, 2.0f, 0.0f); // mira al torso
camera.setFront(glm::normalize(lookTarget - cameraPosition));
```

Para resolver el problema de que la cámara siga al personaje blade, se realizó lo siguiente.

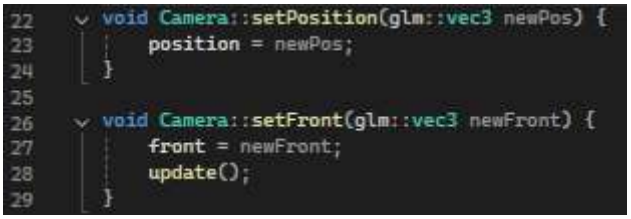
Primero, se obtiene la dirección de la cámara y se proyecta en el plano XZ (horizontal) para evitar que la cámara afecte el eje Y, se detectan las teclas presionadas (por ejemplo, WASD) y se mueve el personaje en relación con la dirección de la cámara (cameraForward) y su vector lateral (right).

Se calcula un ángulo con atan2 para que el personaje Blade rote mirando hacia donde apunta la cámara, Se aplican las transformaciones al modelo para moverlo a su nueva posición y rotarlo. Después se define una posición relativa detrás y por encima del personaje para que la cámara siempre lo siga.

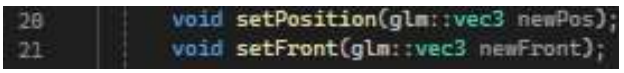
Finalmente, se obtiene como resultado que el personaje siempre se mueve en la dirección en que apunta la cámara. La cámara se mantiene detrás del personaje, actualizando su posición en cada frame.Al mover el mouse, el personaje rota y la cámara lo sigue.

También se modificó el archivo camera.cpp y camera.h para poder lograr este objetivo.

Camera.cpp:



Camera.h:



ANIMACIÓN:

Stands:

Para la animación de los stands se utilizó la variable que se cambia según la cámara del stand para que cuando esa variable tenga el número del stand, la animación comience y cuando termine la animación y los modelos dentro de ella desaparezcan y no se ejecute en el mundo.

Primero se revisa con un if cuál stand es el que fue presionado por la tecla, luego se bloquea el cambio de cámaras con la variable de banderaCamara = 1 (0 para que se puedan hacer cambios de cámara).

Dentro hay otro if que revisa el tiempo del stand (tiempoBateo) que se calcula con una suma con deltaTime al final del if, este tiempo indica el tiempo total que la cámara (y la animación) durará. Cuando se acaba el timer la animación termina y la cámara se regresa a la del avatar.

Primero se inicia una animación de entregar la moneda, la cual cambia a la moneda con la que se consiguen cosas en dichos juegos según el stand pues cada stand es de un universo.

```
// ----------------------------------
// ----------ANIMACIONES STANDS----------
// ----------------------------------

if (camaraAnimacion == 1) { //Stand hacha 3
    banderaCamara = 1; //Para que la camara no se pueda mover hasta que se acabe la animación. Al final debe volver a 0.

    if (timerBateo < 500.0) { //Timpo que va a durar la animación general del stand

        //INICIO ANIMACIÓN DE LA MONEDA

        if (brazoVariacion <= 100.0) {
            //Antebrazo derecho
            model = glm::mat4(1.0);
            model = glm::translate(model, glm::vec3(-107.2f, 7.1f, -78.6f));
            model = glm::rotate(model, -brazoVariacion * toRadians, glm::vec3(0.0f, 0.0f, 1.0f));//mueve arriba (-) o abajo (+)
            glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
            BrazoDerecho_Izquierda.RenderModel();

            brazoVariacion += 0.5f * deltaTime;
        }
        if (timerBateo > 50.0 && timerBateo < 180.0 && brazoVariacion >= 100 && brazoVariacion1 == 100.0) {
            //Antebrazo derecho
            model = glm::mat4(1.0);
            model = glm::translate(model, glm::vec3(-107.2f, 7.1f, -78.6f));
            model = glm::rotate(model, -brazoVariacion * toRadians, glm::vec3(0.0f, 0.0f, 1.0f));//mueve arriba (-) o abajo (+)
            glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
            BrazoDerecho_Izquierda.RenderModel();

            if (timerBateo > 100.0 && timerBateo < 170.0) {
                //Moneda del juego
                model = glm::mat4(1.0);
                model = glm::translate(model, glm::vec3(-108.85f, 7.6f, -78.6f));//(x -1.65, y +0.5, z misma)
                model = glm::rotate(model, 90 * toRadians, glm::vec3(0.0f, 1.0f, 0.0f));
                glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
                Destino_Genshin.RenderModel();
            }
        }
```

```
    if (brazoVariacion1 >= 20.0 && brazoVariacion >= 100 && timerBateo > 180.0) {
        //Antebrazo derecho
        model = glm::mat4(1.0);
        model = glm::translate(model, glm::vec3(-107.2f, 7.1f, -78.6f));
        model = glm::rotate(model, -brazoVariacion1 * toRadians, glm::vec3(0.0f, 0.0f, 1.0f));//mueve arriba (-) o abajo (+)
        glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
        BrazoDerecho_Izquierda.RenderModel();

        brazoVariacion1 -= 0.5f * deltaTime;
    }
    //FIN ANIMACIÓN DE LA MONEDA
```

Una vez acaba la animación de la moneda, inicia la animación de cada stand con sus ifs y sus variables.

```
    if (movBateoPelotaRegreso < 10.0 && movBateo >= 30.0 && inicioBrazo >= 100.0) { //Esperar
        //Antebrazo derecho
        model = glm::mat4(1.0);
        model = glm::translate(model, glm::vec3(-107.2f, 7.1f, -78.6f));
        model = glm::rotate(model, -inicioBrazo * toRadians, glm::vec3(0.0f, 0.0f, 1.0f));//mueve arriba (-) o abajo (+)
        model = glm::rotate(model, -70 * toRadians, glm::vec3(0.0f, 1.0f, 0.0f));
        //model = glm::rotate(model, -movBateo * toRadians, glm::vec3(1.0f, 0.0f, 0.0f));
        glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
        BrazoDerecho_Derecha.RenderModel();

        //Hacha
        model = glm::mat4(1.0);
        model = glm::translate(model, glm::vec3(-109.0f, 4.0f + (inicioBrazo * 0.03), -78.5f));
        model = glm::rotate(model, 90 * toRadians, glm::vec3(0.0f, 1.0f, 0.0f));
        model = glm::rotate(model, movBateo * toRadians, glm::vec3(1.0f, 0.0f, 0.0f));
        glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
        Hacha_M.RenderModel();

        movBateoPelotaRegreso += 0.25f * deltaTime;
    }
    if (movBateoPelota < 45.0 && movBateoPelotaRegreso >= 10.0 && movBateo >= 30.0 && inicioBrazo >= 100.0) { //Hace el hacha hacia d
        //Antebrazo derecho
        model = glm::mat4(1.0);
        model = glm::translate(model, glm::vec3(-107.2f, 7.1f, -78.6f));
        model = glm::rotate(model, -inicioBrazo * toRadians, glm::vec3(0.0f, 0.0f, 1.0f));//mueve arriba (-) o abajo (+)
        model = glm::rotate(model, -70 * toRadians, glm::vec3(0.0f, 1.0f, 0.0f));
        //model = glm::rotate(model, -movBateo * toRadians, glm::vec3(1.0f, 0.0f, 0.0f));
        glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
        BrazoDerecho_Derecha.RenderModel();

        //Hacha
        model = glm::mat4(1.0);
        model = glm::translate(model, glm::vec3(-109.0f, 4.0f + (inicioBrazo * 0.03), -78.5f));
        model = glm::rotate(model, 90 * toRadians, glm::vec3(0.0f, 1.0f, 0.0f));
        model = glm::rotate(model, (movBateo - movBateoPelota) * toRadians, glm::vec3(1.0f, 0.0f, 0.0f));
        glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
        Hacha_M.RenderModel();

        movBateoPelota += 0.7f * deltaTime;
    }
```

No se muestra toda la animación ya que es muy larga (el código). Sin embargo, al final del if del timer hay un else que resetea todas las variables utilizadas a su valor inicial para que cuando se vuelva a presionar la tecla las variables actúen como es esperado. En la imagen se observa que al final hay un if else para comenzar el siguiente stand (su animación).

```
        }

        ////

        timerBateo += 0.5f * deltaTime;
    }
    else {
        //VARIABLES FUNCIONALIDAD CÁMARA Y ANIMACIÓN MONEDA (NO MOVER)
        banderaCamaraMovimiento = 0;//para que la camara regrese a la camara de blade de manera automatica
        banderaCamara = 0;//para que ya se puedan usar las teclas de camaras
        camaraAnimacion = 0;//
        brazoVariacion = 20.0;
        brazoVariacion1 = 100.0;
        timerBateo = 0.0;

        //VARIABLES ANIMACIÓN DEL STAND
        inicioBrazo = 0.0;
        movBateo = 0.0;
        movBateoPelota = 0.0;
        movBateoPelotaRegreso = 0.0;
        movBateoRegreso = 0.0;
        movDardoNuevo = 0.0;
        //
    }
}
else if (camaraAnimacion == 2) { //Stand boliche
    banderaCamara = 1; //Para que la camara no se pueda mover hasta que se acabe la animación. Al final debe volver a 0.

    if (timerBateo < 1650.0) { //Timpo que va a durar la animación general del stand
```

Animaciones utilizando función seno:

Para el proyecto, tanto Blade como el bote de basura se utilizaron funciones seno para simular los movimientos necesarios para que ambos parecieran que estaban caminando. Aunque ambos tienen pequeñas diferencias en común.

En el caso de blade se utilizaron valores completos del seno para las articulaciones de las piernas y los hombros, mientras que para los antebrazos y las pantorrillas se utilizaron exclusivamente los valores positivos del seno, esto con el propósito de que estas extremidades no se fueran en una dirección antinatural, como por ejemplo, el codo flexionandose en una dirección que no debería.

```
if (mainWindow.getAvanzaBlade() == 1) {

    angulovaria += 2.2f * deltaTime;
    angulovariaAux += 2.2f * deltaTime;
    if (sin(glm::radians(angulovariaAux)) > 0) {
        angulovaria2 = 0.0f;
    }
    else {
        angulovaria2 = angulovariaAux;
    }
    if (sin(glm::radians(angulovariaAux)) < 0) {
        angulovaria3 = 0.0f;
    }
    else {
        angulovaria3 = -angulovariaAux;
    }
    //std::cout << "angulovaria = " << sin(glm::radians(angulovaria2)) << std::endl;

}
```

En el caso de blade, se añadió un apartado para que, cuando el personaje haya dejado de caminar y la animación no haya terminado, este regrese a la pose original para que no parezca tener movimientos abruptos al momento de moverse.

```
if ((sin(glm::radians(angulovaria)) > 0.1 && mainWindow.getAvanzaBlade() == 0) || (sin(glm::radians(angulovaria)) < -0.1 && mainWindow.getAvanzaBlade() == 0))
    angulovaria += 3.0f * deltaTime;
    angulovariaAux += 3.0f * deltaTime;
    if (sin(glm::radians(angulovariaAux)) > 0) {
        angulovaria2 = 0.0f;
    }
    else {
        angulovaria2 = angulovariaAux;
    }
    if (sin(glm::radians(angulovariaAux)) < 0) {
        angulovaria3 = 0.0f;
    }
    else {
        angulovaria3 = -angulovariaAux;
    }
    //std::cout << "angulovaria = " << sin(glm::radians(angulovaria)) << std::endl;
}
```

En el caso del bote de basura, se siguió la misma lógica para que se moviera, con la diferencia de que este se mueve permanente, por lo que no hizo falta el agregado que tiene Blade para terminar bien sus animaciones.

Como el bote de basura tiene una animación infinita, se agregaron modificaciones para que este se moviera en una dirección en forma de ida y vuelta, todo esto con una función seno de igual manera, el programa detecta cuando el seno comienza bajar o a subir y de acuerdo a esto le asigna una dirección al bote de basura.

```
float senoActual = sin(glm::radians(BanguloAvanza));

if (senoActual >= 1.0f - epsilon && !yaCambio) {
    BanguloMira = 180.0f;  // Mira hacia un lado
    yaCambio = true;
}
else if (senoActual <= -1.0f + epsilon && !yaCambio) {
    BanguloMira = 00.0f; // Mira al otro, je
    yaCambio = true;
}
else if (senoActual < 1.0f - epsilon && senoActual > -1.0f + epsilon) {
    yaCambio = false;
}
senoAnterior = senoActual;
```

Para los protagonistas y NPC:

Se integró en donde se manda a llamar a cada puesto y personaje la animación. Se les partieron las articulaciones necesarias para el movimiento y se animaron con ifs y variables que se resetean al terminar el valor que se calcula con deltaTime para que la animación pueda ser cíclica.

```
// ------------------------------------
// ----------PUESTO HELADOS------------
// ------------------------------------

//Puesto helado
model = glm::mat4(1.0);
model = glm::translate(model, glm::vec3(65.0f, 0.0f, 25.0f));
model = glm::rotate(model, 90 * toRadians, glm::vec3(0.0f, 1.0f, 0.0f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
PuestoHelado_M.RenderModel();

//Marius Casual Tears of Themis Cuerpo
model = glm::mat4(1.0);
model = glm::translate(model, glm::vec3(66.5f, 0.0f, 24.0f));
//model = glm::rotate(model, 90 * toRadians, glm::vec3(0.0f, 1.0f, 0.0f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
MariusCasual_M.RenderModel();
```

```
if (ContadorInicioPrograma == 1) {
    //Marius brazo
    if (banderaMarius == 0 && MariusBrazo < 40.0f) {
        MariusBrazo += 0.2 * deltaTime;
    }
    else if (MariusBrazo >= 0.0f) {
        MariusBrazo -= 0.2 * deltaTime;
        banderaMarius = 1;
        if (helado1 == 0 && MariusContador2 == 0) helado1 = 1;
        if (helado2 == 0 && MariusContador2 == 1) helado2 = 1;
        if (helado3 == 0 && MariusContador2 == 2) helado3 = 1;

    }
    else {
        banderaMarius = 0;
        MariusContador2 ++;
        if (MariusContador < 2) MariusContador += 1;
        else MariusContador = 0;
    }

    if (MariusContador2 == 3) {
        MariusContador2 = 0;
        helado1 = 0;
        helado2 = 0;
        helado3 = 0;
    }

    model = glm::translate(model, glm::vec3(-0.44f, 7.13f, 2.73f));
    model = glm::rotate(model, MariusBrazo * toRadians, glm::vec3(0.0f, 1.0f, 0.0f));
    glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
    MariusCasualBrazo_M.RenderModel();
```

```
//Para que la bola gire en el cucharón
if (MariusBrazo <= 40.0f && banderaMarius == 0 && MariusContador == 0) {
    model = glm::translate(model, glm::vec3(1.7f, -1.6f, -0.45f));
    model = glm::rotate(model, -50 * toRadians, glm::vec3(0.0f, 1.0f, 0.0f));
    model = glm::rotate(model, -180 * toRadians, glm::vec3(1.0f, 0.0f, 0.0f));
    model = glm::rotate(model, -MariusBrazo * 25 * toRadians, glm::vec3(0.0f, 0.0f, 1.0f));
    glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
    BolaHelado_Suelta_M.RenderModel();//azul
}
else if (MariusBrazo <= 40.0f && banderaMarius == 0 && MariusContador == 1) {
    model = glm::translate(model, glm::vec3(1.7f, -1.6f, -0.45f));
    model = glm::rotate(model, -50 * toRadians, glm::vec3(0.0f, 1.0f, 0.0f));
    model = glm::rotate(model, -180 * toRadians, glm::vec3(1.0f, 0.0f, 0.0f));
    model = glm::rotate(model, -MariusBrazo * 25 * toRadians, glm::vec3(0.0f, 0.0f, 1.0f));
    glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
    BolaHelado_Suelta_Morada_M.RenderModel();//morada
}
else if (MariusBrazo <= 40.0f && banderaMarius == 0 && MariusContador == 2) {
    model = glm::translate(model, glm::vec3(1.7f, -1.6f, -0.45f));
    model = glm::rotate(model, -50 * toRadians, glm::vec3(0.0f, 1.0f, 0.0f));
    model = glm::rotate(model, -180 * toRadians, glm::vec3(1.0f, 0.0f, 0.0f));
    model = glm::rotate(model, -MariusBrazo * 25 * toRadians, glm::vec3(0.0f, 0.0f, 1.0f));
    glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
    BolaHelado_Suelta_Naranja_M.RenderModel();//naranja
}
```

```
//Para aparecer las bolas en el cono
if (helado1 == 1) {
    model = glm::mat4(1.0);
    model = glm::translate(model, glm::vec3(66.5f, 0.0f, 24.0f));
    glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
    BolaHelado_Izquierda_M.RenderModel();//primera bola
}
if (helado2 == 1) {
    model = glm::mat4(1.0);
    model = glm::translate(model, glm::vec3(66.5f, 0.0f, 24.0f));
    glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
    BolaHelado_Derecha_M.RenderModel();//segunda bola
}
if (helado3 == 1) {
    model = glm::mat4(1.0);
    model = glm::translate(model, glm::vec3(66.5f, 0.0f, 24.0f));
    glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
    BolaHelado_Arriba_M.RenderModel();//tercera bola
}
```

En este caso el personaje mueve el antebrazo para simular que está haciendo una bola de helado, la bola rota para que parezca que está siendo creada y luego se pone la bola (slime de Genshin Impac) en el helado como una bola de helado, la animación sigue hasta que termina de hacer las 3 diferentes bolas y se reinicia.

En general, se realizaron con rotates y translate.

ILUMINACIÓN Y MATERIALES:

Se crearon dos arreglos principales de luces, las de día y las de noche. Para el caso de las luces de día se mueven por teclado dos elementos, por lo que se requirieron de 4 condicionales. Para el caso de las luces de noche interactúa una sola luz, por lo que se utilizaron dos condicionales. Adicional a esto se crearon 12 arreglos, uno para cada atracción fuera de día y de noche.

```
// ----------------------------------------
// -------------DEFINIR LUCES--------------
// ----------------------------------------

//Luz direccional
DirectionalLight mainLight;

//para declarar varias luces
PointLight pointLightsDia[MAX_POINT_LIGHTS];
SpotLight spotLightsDia[MAX_SPOT_LIGHTS];

PointLight pointLightsDia2[MAX_POINT_LIGHTS]; // Para las luces interactivas con teclado

PointLight pointLightsNoche[MAX_POINT_LIGHTS];
SpotLight spotLightsNoche[MAX_SPOT_LIGHTS];

//Luz Hacha
SpotLight spotLightsDiaHacha[MAX_SPOT_LIGHTS];
SpotLight spotLightsNocheHacha[MAX_SPOT_LIGHTS];

//Luz Boliche
SpotLight spotLightsDiaBoliche[MAX_SPOT_LIGHTS];
SpotLight spotLightsNocheBoliche[MAX_SPOT_LIGHTS];

//Luz Dados
SpotLight spotLightsDiaDados[MAX_SPOT_LIGHTS];
SpotLight spotLightsNocheDados[MAX_SPOT_LIGHTS];

//Luz Bateo
SpotLight spotLightsDiaBateo[MAX_SPOT_LIGHTS];
SpotLight spotLightsNocheBateo[MAX_SPOT_LIGHTS];

//Luz Dardos
SpotLight spotLightsDiaDardos[MAX_SPOT_LIGHTS];
SpotLight spotLightsNocheDardos[MAX_SPOT_LIGHTS];

//Luz Topos
SpotLight spotLightsDiaTopos[MAX_SPOT_LIGHTS];
SpotLight spotLightsNocheTopos[MAX_SPOT_LIGHTS];
```

Para el ciclo de día y de noche se utilizaron las condicionales creadas para el skybox, y se modificaron elementos de la luz.

```
if (skyCount == 0) { //Amanecer
    skybox = Skybox(skyboxFaces);  //Skybox de dia

    //Recibe Color nuevo RGC, dirección nueva, ambientIntensity, diffuseIntensity
    mainLight.SetLight(glm::vec3(0.9843f, 0.9843f, 0.8980f), glm::vec3(0.0f, 0.0f, 1.0f), 0.5f, 0.6f);
}
else if (skyCount == 1000) {//Medio dia

    //Recibe Color nuevo RGC, dirección nueva, ambientIntensity, diffuseIntensity
    mainLight.SetLight(glm::vec3(1.0f, 0.9647f, 0.9333f), glm::vec3(0.0f, -1.0f, 0.0f), 0.43f, 0.37f);
}

else if (skyCount == 2000) {//Atardecer
    skybox = Skybox(skyboxFacesAtard);  //Skybox de atardecer

    //Recibe Color nuevo RGC, dirección nueva, ambientIntensity, diffuseIntensity
    mainLight.SetLight(glm::vec3(0.9450f, 0.6274f, 0.4745f), glm::vec3(0.0f, 0.0f, -1.0f), 0.45f, 0.6f);
}
else if (skyCount == 3000) { //Noche
    skybox = Skybox(skyboxFacesNoche);  //Skybox de noche

    //Recibe Color nuevo RGC, dirección nueva, ambientIntensity, diffuseIntensity
    mainLight.SetLight(glm::vec3(0.9176f, 0.9411f, 0.9686f), glm::vec3(0.0f, 0.0f, 1.0f), 0.5f, 0.4f);
}
else if (skyCount == 6000) { //Termina la noche
    skyCount = -1; //Se le resta uno porque abajo se suma 1 y al sumarse 1 ya no queda en e para el reinicio
}

skyCount += 1; //La variable contadora aumenta en uno con cada ciclo del while que es lo que lleva la cuenta del tiempo
```

Para animar las luces de la feria, se utilizaron contadores y condicionales para, dentro de éstas, modificar valores de la luz. Esto se realizó para la rueda de la fortuna, el carrusel, los carros chocones, puesto de tickets, lámparas; también para las atracciones de bateo y topos.

```
// ------------------------------------
// ----------------LUCES---------------
// ------------------------------------

//Rueda Fortuna
if (luzContador == 0) {
    //Color, posición, constante, lineal, exponencial (Rosa)
    pointLightsDia[0].SetLight(glm::vec3(0.9843f, 0.0862f, 0.8705f), glm::vec3(0.0f, 67.0f, 0.0f));
}
else if (luzContador == 150) {
    //Color, posición, constante, lineal, exponencial (Azul)
    pointLightsDia[0].SetLight(glm::vec3(0.1529f, 0.8862f, 1.0f), glm::vec3(0.0f, 67.0f, 0.0f));
}
else if (luzContador == 300) {
    //Color, posición, constante, lineal, exponencial (Morado)
    pointLightsDia[0].SetLight(glm::vec3(0.7333f, 0.2156f, 0.9764f), glm::vec3(0.0f, 67.0f, 0.0f));
}
else if (luzContador == 450) {
    //Color, posición, constante, lineal, exponencial (Amarillo)
    pointLightsDia[0].SetLight(glm::vec3(0.9764f, 0.9725f, 0.2156f), glm::vec3(0.0f, 67.0f, 0.0f));
}
else if (luzContador == 600) {
    luzContador = -1;
}
luzContador += 1;
```

Materiales implementados:

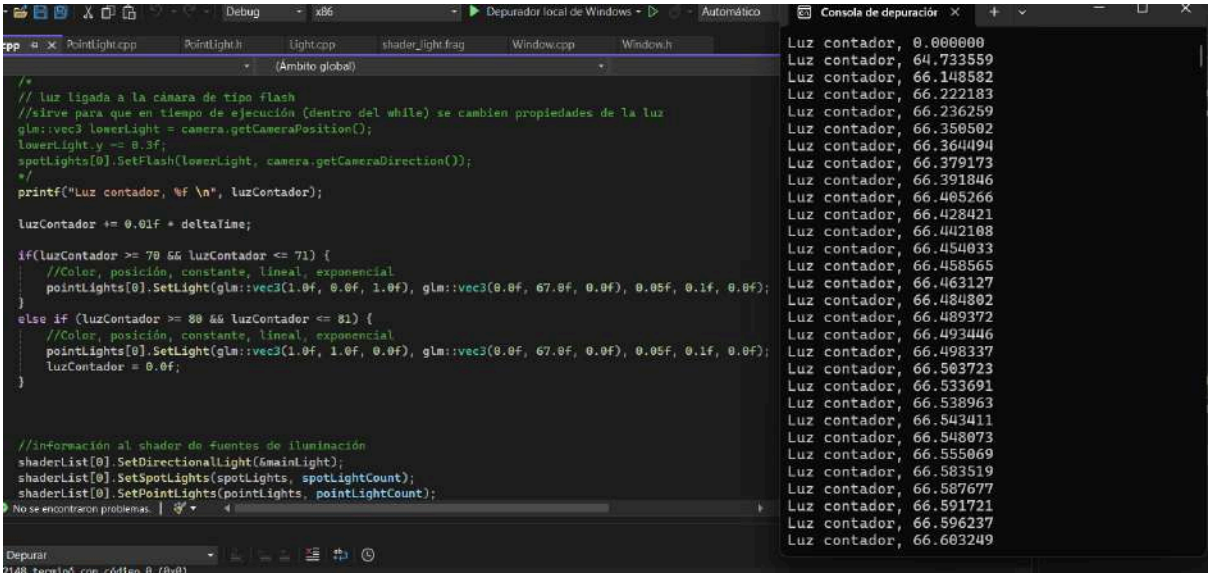
Además de los dos proporcionados por el profesor, se utilizaron:

- Personaje caricatura 3D (Avatar): Specular Intensity 0.1, Shininess 10. Para un estilo no realista, se evitaron los brillos fuertes, esto para mantener la estética "mate".
- Madera: Specular Intensity 0.4, Shininess 10. Superficie con poco reflejo.
- Reja (metal): Specular Intensity 0.8, Shininess 50. Superficie con mayor reflexión.
- Terciopelo: Specular Intensity 0.1, Shininess 5. Muy poca reflexión.
- Madera pulida: Specular Intensity 0.7, Shininess 50. Superficie con reflejos notables.
- Plástico: Specular Intensity 0.8, Shininess 30. Una mayor reflexión que la madera simple, pero menor que la madera pulida.

```
// ---------------------------------------
// ----------CREAR MATERIALES-------------
// ---------------------------------------

Material_brillante = Material(4.0f, 256);
Material_opaco = Material(0.3f, 4);

Material_Avatar = Material(0.1f, 10);
Material_Madera = Material(0.4f, 10);
Material_Reja = Material(0.8f, 50);
Material_Terciopelo = Material(0.1f, 5);
Material_MaderaPulida = Material(0.7f, 50);
Material_Plastico = Material(0.8f, 30);
```

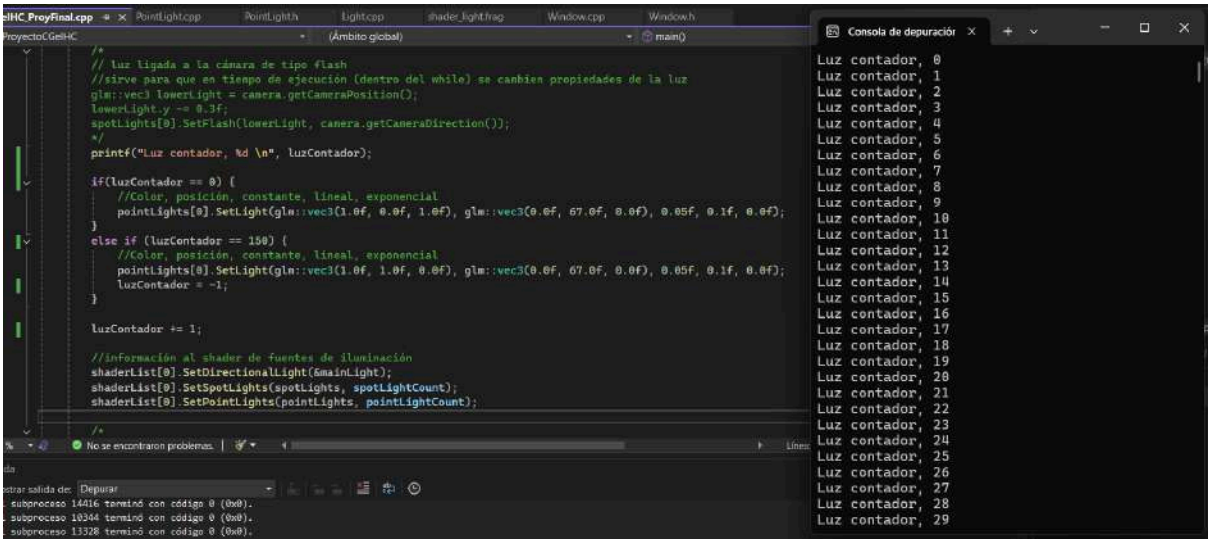
PROBLEMAS OCURRIDOS:

Problemas con el uso de deltaTime para animaciones iniciales (donde se debe de tomar en cuenta un tiempo transcurrido):



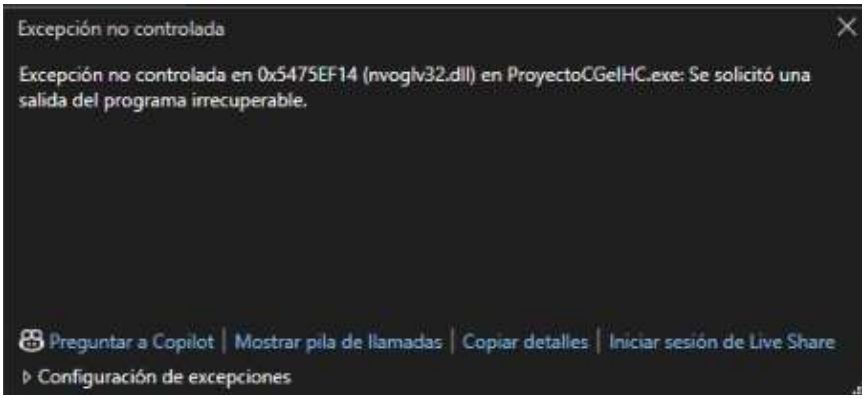
Por alguna razón, deltaTime tiene un intervalo masivo al correrse desde el principio del programa, aún cuando se está multiplicando el valor por 0.01, por lo que dificulta trabajar con valores homogéneos al inicio y al intermedio de la ejecución, en dichos casos se utilizaron contadores, por ejemplo, para las luces y skybox.

Diferencia: el contador sí comienza desde 0 y se va actualizando homogéneamente.



Error de carga de texturas duplicadas:

Durante la carga de modelos se presento un problema, este problema se generaba cuando se trataban de cargar muchos modelos al mismo tiempo pese a que estos funcionasen de manera individual

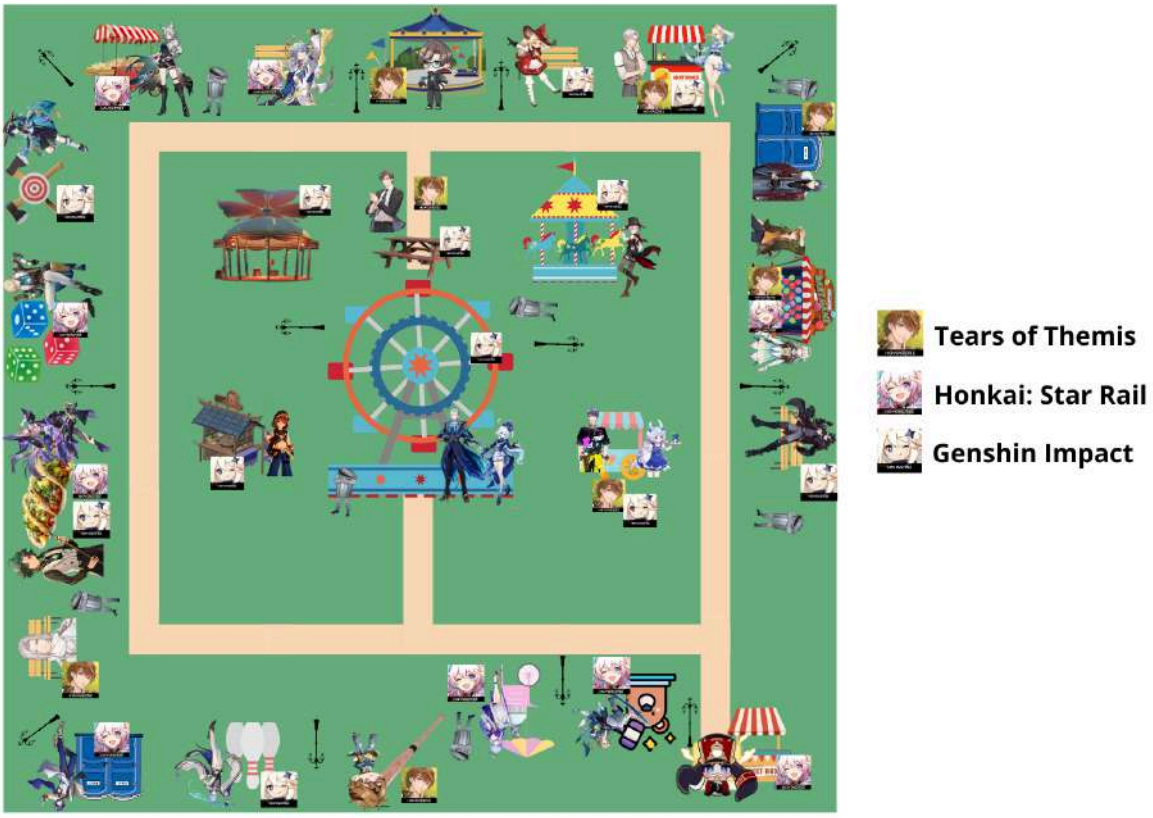


Después de presentarse el problema probé el proyecto en diferentes computadoras, en las computadoras con tarjeta Nvidia daba error directamente mientras que en las computadoras con tarjeta AMD el proyecto cargaba pero en cada ejecución solo se ejecutaban algunas texturas de manera aleatoria por lo que se infirió que el problema era el texturizado.

Para solucionar el problema se agregó una estructura mapa, la cual utilizamos para verificar que no existieran texturas duplicadas dentro del código, esta modificación hace que si existe la textura la reutiliza, en caso contrario, la guarda. Todo esto con el propósito de que no se guarde la misma textura muchas veces. La modificación es la siguiente.

```cpp
void Model::LoadMaterials(const aiScene* scene)
{
    std::unordered_map<std::string, Texture*> loadedTextures; // Mapa para evitar duplicados
    TextureList.resize(scene->mNumMaterials);

    for (unsigned int i = 0; i < scene->mNumMaterials; i++)
    {
        aiMaterial* material = scene->mMaterials[i];
        TextureList[i] = nullptr;

        if (material->GetTextureCount(aiTextureType_DIFFUSE))
        {
            aiString path;
            if (material->GetTexture(aiTextureType_DIFFUSE, 0, &path) == AI_SUCCESS)
            {
                int idx = std::string(path.data).rfind("\\"); // quitar path anterior
                std::string filename = std::string(path.data).substr(idx + 1);
                std::string texPath = "Textures/" + filename;

                // Verifica si ya se cargó esta textura antes
                auto it = loadedTextures.find(texPath);
                if (it != loadedTextures.end())
                {
                    TextureList[i] = it->second; // Reutiliza la textura ya cargada
                }
                else
                {
                    Texture* newTex = new Texture(texPath.c_str());
                    std::string ext = filename.substr(filename.find_last_of('.') + 1);

                    bool loaded = false;
                    if (ext == "tga" || ext == "png")
                        loaded = newTex->LoadTextureA();
                    else
                        loaded = newTex->LoadTexture();

                    if (loaded)
                    {
                        TextureList[i] = newTex;
                        loadedTextures[texPath] = newTex; // Almacena para reutilizarla
                    }
                    else
                    {
                        printf("Falló en cargar la Textura :%s\n", texPath.c_str());
                        delete newTex;
                    }
                }
            }
        }

        // Si aún no se asignó una textura válida
        if (!TextureList[i])
        {
            TextureList[i] = new Texture("Textures/plain.png");
            TextureList[i]->LoadTextureA();
        }
    }
```

MAPA FINAL:



https://www.canva.com/design/DAGh3WYsyWM/rfQ0xuv-rKtTiH8ekn7IJg/edit?utm_content=DAGh3WYsyWM&utm_campaign=designshare&utm_medium=link2&utm_source=sharebutton

Las teclas a utilizar fueron:

F → Cámara del avatar.

Q → Cámara extra “libre”.

G → Cámara aérea.

H → Stand hacha.

J → Stand boliche.

K → Stand dados.

L → Stand bateo.

M → Stand dardos.

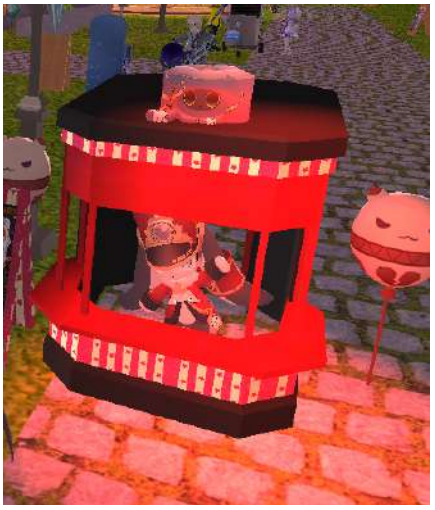
N → Stand topo.

O → Luz en puesto de Inazuma.

P → Luz en el puesto de hot dogs.

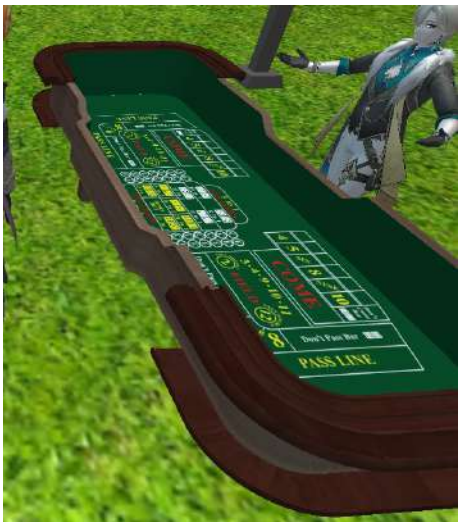
COMPARACIÓN PROPUESTA Y RESULTADO FINAL

1. Cabina de tickets



2. Puesto de algodon de azucar

3. Puesto de lanzamiento de dados



4. Baños portatiles

5. Mesas

6. Golpear al topo

7. Puesto de lanzamiento de dardos


8. Farola


9. Botes de basura de videojuego “Honkai: Star Rail”

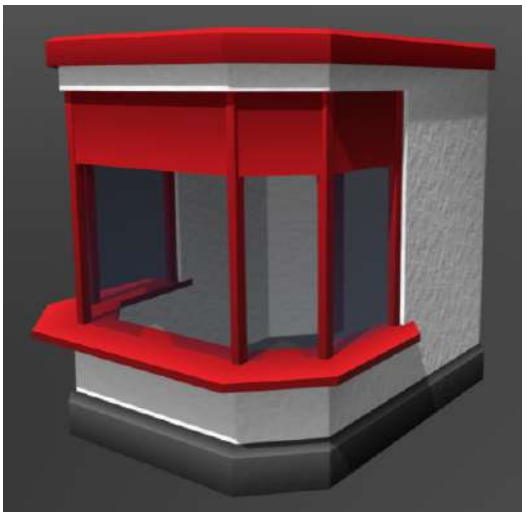
10.Banca

CONCLUSIONES

Con la realización de este proyecto se incorporaron todos los temas vistos y se demostró el manejo de la teoría y la práctica de lo visto a lo largo del semestre. Hubo muchos retos ya que es un trabajo muy ambicioso, sin embargo, se superaron y se consiguió un resultado con el que estoy satisfecho. Logre aplicar de manera clara el uso de modelos, su texturizado, la animación de los mismos y el uso correcto de luces para que la feria se viera animada. También resultó valioso ya que aprendimos muchas cosas sobre la computación gráfica y la manera en la que se expresan cosas a través de una pantalla y un programa.

REFERENCIAS DE ELEMENTOS UTILIZADOS

Modelos de la propuesta:



Sketchfab. (s. f.). Ticket booth [Modelo 3D]. Sketchfab. Recuperado de https://sketchfab.com/3d-models/ticket-booth-7e12559eed534b5699c4262e7e1a6bb1

Sketchfab. (s. f.). Cotton candy machine [Modelo 3D]. Sketchfab. Recuperado de https://sketchfab.com/3d-models/cotton-candy-machine-971fd0043c594d1b8493f4d0f93e24b7

thingiverse.com. (s. f.). 8 foot craps table [Modelo 3D]. Thingiverse. Recuperado de https://www.thingiverse.com/thing:3606436

Berk Gedik. (s. f.). Abandoned Toilet Cabin (Low Poly) [Modelo 3D]. Sketchfab. Recuperado de https://skfb.ly/6VyIo

Gamedirection. (s. f.). Park Table - Low Poly [Modelo 3D]. Sketchfab. Recuperado de https://skfb.ly/6CTuW

Sketchfab. (s. f.). Whack-a-mole machine [Modelo 3D]. Sketchfab. Recuperado de https://sketchfab.com/3d-models/whack-a-mole-machine-ba599fb0184b4d5286ef53f0e8d1bb65

Keyotine. (s. f.). Stylized Carnival Booth [Modelo 3D]. Sketchfab. Recuperado de https://skfb.ly/oIyrq

avsteir. (s. f.). HW2 10 props Genshin Impact [Modelo 3D]. Sketchfab. Recuperado de https://skfb.ly/oAuUn

aplaybox.com. (s. f.). [Modelo 3D sin título]. Aplaybox. Recuperado de https://www.aplaybox.com/details/model/qTXzeDkYaiBu

avsteir. (s. f.). HW2 10 props Genshin Impact [Modelo 3D]. Sketchfab. Recuperado de https://skfb.ly/oAuUn

Modelos utilizados en las demás atracciones, referencias.

- avsteir. (s. f.). *HW2 10 props Genshin Impact* [Modelo 3D]. Sketchfab. Recuperado de https://skfb.ly/oAuUn
- *Free png*. (s. f.). PNGwing. Recuperado de https://www.pngwing.com/en/free-png-dcabk/download
- *Mods*. (s. f.). GameBanana. Recuperado de https://gamebanana.com/mods/7921
- aplaybox.com. (s. f.-a).【崩坏:星穹铁道】帕姆 [Modelo 3D]. Aplaybox. Recuperado de https://www.aplaybox.com/details/model/R50C73gabDfS
- Sketchfab. (s. f.-a). *Ticket booth* [Modelo 3D]. Sketchfab. Recuperado de https://sketchfab.com/3d-models/ticket-booth-7e12559eed534b5699c4262e7e1a6bb1
- iStockphoto. (s. f.). *Circo carnaval retro rayas vintage con estrellas patrón sin costuras plantilla gráfica* [Ilustración]. iStock. Recuperado de https://media.istockphoto.com/id/1212893750/es/vector/circo-carnaval-retro-rayas-vintage-con-estrellas-patr%C3%B3n-sin-costuras-plantilla-gr%C3%A1fica.jpg?s=170667a&w=0&k=20&c=nXl_aY9Ad-hgjyxer7KxCeojMB26gpJA-QugSXEnUg8=
- Sketchfab. (s. f.-b). *Cotton candy machine* [Modelo 3D]. Sketchfab. Recuperado de https://sketchfab.com/3d-models/cotton-candy-machine-971fd0043c594d1b8493f4d0f93e24b7
- aplaybox.com. (s. f.-b).【崩坏:星穹铁道】知更鸟 [Modelo 3D]. Aplaybox. Recuperado de https://www.aplaybox.com/details/model/ZBeEOboYGD6z
- algodonin.es. (2021, marzo). *ALGODÓN AZÚCAR HEADER* [Imagen]. Algodonin. Recuperado de https://algodonin.es/wp-content/uploads/2021/03/ALGODO%CC%81N-AZUCAR-HEADER.png
- aplaybox.com. (s. f.-c).【崩坏:星穹铁道】砂金 [Modelo 3D]. Aplaybox. Recuperado de https://www.aplaybox.com/details/model/Njvgh4VhWuWC
- thingiverse.com. (s. f.). 8 foot craps table [Modelo 3D]. Thingiverse. Recuperado de https://www.thingiverse.com/thing:3606436

- ac-illust.com. (s. f.). [Imagen sin título]. AC Illust. Recuperado de https://thumb.ac-illust.com/52/52fee2eeca0d0ccb870f4ebb5f9171b8_t.jpeg
- aplaybox.com. (s. f.).【崩坏：星穹铁道】真理医生 [Modelo 3D]. Aplaybox. Recuperado de https://www.aplaybox.com/details/model/vLCYcQmeo9Hj
- Sketchfab. (s. f.). *Abandoned toilet cabin - low poly* [Modelo 3D]. Sketchfab. Recuperado de https://sketchfab.com/3d-models/abandoned-toilet-cabin-low-poly-8083b3d2134f4e98a34d891fa7915dcc
- aplaybox.com. (s. f.).【崩坏：星穹铁道】星期日 [Modelo 3D]. Aplaybox. Recuperado de https://www.aplaybox.com/details/model/yRxpSCxnTXzm
- X9_YT. (s. f.). *Genshin impact - Furina Sword* [Modelo 3D]. Sketchfab. Recuperado de https://skfb.ly/oOBUD
- aplaybox.com. (s. f.-f).【崩坏：星穹铁道】彦卿 [Modelo 3D]. Aplaybox. Recuperado de https://www.aplaybox.com/details/model/BbouJB2vGqv8
- Sketchfab. (s. f.-d). *Whack-a-mole machine* [Modelo 3D]. Sketchfab. Recuperado de https://sketchfab.com/3d-models/whack-a-mole-machine-ba599fb0184b4d5286ef53f0e8d1bb65
- Sketchfab. (s. f.-e). *Moles and hammers* [Modelo 3D]. Sketchfab. Recuperado de https://sketchfab.com/3d-models/moles-and-hammers-a8d4eef5d76640b889f4e3debe96ba60
- reddit.com. (s. f.). [Imagen sin título de fanart de Herta]. Reddit. Recuperado de https://i.redd.it/my-herta-fanart-v0-xeh08ff5brme1.jpg?width=4000&format=pjpg&auto=webp&s=6b0a75211442a9e04b7c93650adb076aff471333
- aplaybox.com. (s. f.).【崩坏：星穹铁道】黑塔 [Modelo 3D]. Aplaybox. Recuperado de https://www.aplaybox.com/details/model/0dSde2NajZJV
- aplaybox.com. (s. f.).【原神】那维莱特 [Modelo 3D]. Aplaybox. Recuperado de https://www.aplaybox.com/details/model/WSqyz5HczW86
- aplaybox.com. (s. f.).【原神】芙宁娜 [Modelo 3D]. Aplaybox. Recuperado de https://www.aplaybox.com/details/model/iBOW5aAVDaoH

- Sketchfab. (s. f.-f). *Ferris wheel* [Modelo 3D]. Sketchfab. Recuperado de https://sketchfab.com/3d-models/ferris-wheel-675ab80b477b40f280b7311f81fee730
- Freepik. (s. f.). *Pecat* [Vector]. Freepik. Recuperado de https://img.freepik.com/premium-vector/pecat_824631-2759.jpg
- Sketchfab. (s. f.-g). *Carousel spinning* [Modelo 3D]. Sketchfab. Recuperado de https://sketchfab.com/3d-models/carousel-spinning-be73168bd1b44dc7b42b2c18395eaa26
- aplaybox.com. (s. f.). [Modelo 3D sin título]. Aplaybox. Recuperado de https://www.aplaybox.com/details/model/Yd2F8O36aW9E
- aplaybox.com. (s. f.). [Modelo 3D sin título]. Aplaybox. Recuperado de https://www.aplaybox.com/details/model/ByIr97MuisgW
- Sketchfab. (s. f.). *Gun targets* [Modelo 3D]. Sketchfab. Recuperado de https://sketchfab.com/3d-models/gun-targets-eb9fbe283faa41ef8685359a0769d303
- aplaybox.com. (s. f.). [Modelo 3D sin título]. Aplaybox. Recuperado de https://www.aplaybox.com/details/model/8INI9RGRm2hW
- aplaybox.com. (s. f.). [Modelo 3D sin título]. Aplaybox. Recuperado de https://www.aplaybox.com/details/model/DWzYm0H7Gztp
- pngtree.com. (2022, junio 16). *Warning ribbon with yellow black stripes png image* [Vector]. PNGtree. Recuperado de https://png.pngtree.com/png-vector/20220616/ourmid/pngtree-warning-ribbon-with-yellow-black-stripes-png-image_5094127.png
- Blender3D. (s. f.). *Simple Axe* [Modelo 3D]. Sketchfab. Recuperado de https://skfb.ly/6Aqus
- aplaybox.com. (s. f.-n). [Modelo 3D sin título]. Aplaybox. Recuperado de https://www.aplaybox.com/details/model/Oh26xPqwZnJe
- tiunov.se. (s. f.). *Bowling Club* [Modelo 3D]. Sketchfab. Recuperado de https://skfb.ly/oPoAC
- samixalam. (s. f.). *Bowling Alley* [Modelo 3D]. Sketchfab. Recuperado de https://skfb.ly/psYZz
- colorate.azurewebsites.net. (s. f.). [Paleta de color]. Colorate. Recuperado de https://colorate.azurewebsites.net/SwatchColor/4B4B4B

- Terizmeh. (s. f.). *Bowling Pin* [Modelo 3D]. Sketchfab. Recuperado de https://skfb.ly/6puNs
- tijmen_h. (s. f.). *Bowling ball return* [Modelo 3D]. Sketchfab. Recuperado de https://skfb.ly/pqqxr
- color-hex.com. (s. f.). [Paleta de color]. Color-Hex. Recuperado de https://www.color-hex.com/palettes/28679.png
- aplaybox.com. (s. f.). [Modelo 3D sin título]. Aplaybox. Recuperado de https://www.aplaybox.com/details/model/PnyEtfvD7Y5A
- gstatic.com. (s. f.). [Imagen sin título]. Googleusercontent. Recuperado de https://encrypted-tbn0.gstatic.com/images?q=tbn:ANd9GcQJT67Bx0xRgA7zg5HP8wFflDaEDk0uEu7XdC9Mvx-MDHKywqClq4FSEjdP-g_NyBmntu4&usqp=CAU
- pngwing.com. (s. f.). [Imagen de textura de madera clara]. PNGwing. Recuperado de https://w7.pngwing.com/pngs/148/189/png-transparent-light-colored-wood-texture-background-wood-texture-wooden-desktop-light-colored-thumbnail.png
- aplaybox.com. (s. f.). [Modelo 3D sin título]. Aplaybox. Recuperado de https://www.aplaybox.com/details/model/oEC8Vg7464Mo
- LittleWhiteElk. (s. f.). *Cherry Soda Float* [Modelo 3D]. Sketchfab. Recuperado de https://skfb.ly/6X9Jv
- aplaybox.com. (s. f.). [Modelo 3D sin título]. Aplaybox. Recuperado de https://www.aplaybox.com/details/model/gRVXCs6EEtCp
- kyou.id. (s. f.). [Imagen de Nendoroid Libra Tears of Themis]. Kyou.id. Recuperado de https://cdn.kyou.id/items/200461-nendoroid-libra-tears-of-themis.jpg.webp
- amazon.com. (s. f.). [Imagen de libro abierto]. Amazon. Recuperado de https://m.media-amazon.com/images/I/719ApjLMi9L.jpg
- sebastianesp. (s. f.). *Open book* [Modelo 3D]. Sketchfab. Recuperado de https://skfb.ly/6WP8M
- aplaybox.com. (s. f.). [Modelo 3D sin título]. Aplaybox. Recuperado de https://www.aplaybox.com/details/model/8Fmk7HdtqzkX
- aplaybox.com. (s. f.). [Modelo 3D sin título]. Aplaybox. Recuperado de https://www.aplaybox.com/details/model/RXj6UEl5ZbYk

- hoyolab.com. (s. f.). [Artículo sin título]. HoYoLAB. Recuperado de https://www.hoyolab.com/article/12096914
- LinjieFan. (s. f.). *electro slime* [Modelo 3D]. Sketchfab. Recuperado de https://skfb.ly/oQDZz
- LinjieFan. (s. f.). *Water slime* [Modelo 3D]. Sketchfab. Recuperado de https://skfb.ly/oQpHw
- LinjieFan. (s. f.). *Fire slime* [Modelo 3D]. Sketchfab. Recuperado de https://skfb.ly/oQ6Jt
- fasteng. (s. f.). *Ice Crteam Scoop* [Modelo 3D]. Sketchfab. Recuperado de https://skfb.ly/oMUOQ
- Sketchfab. (s. f.). *Ice cream food cart* [Modelo 3D]. Sketchfab. Recuperado de https://sketchfab.com/3d-models/ice-cream-food-cart-60b72bdcd1a445abbef6a3a2cbf7de5e
- pngtree.com. (2023, octubre 11). *Vibrant ice cream inspired background texture image* [Imagen]. PNGtree. Recuperado de https://png.pngtree.com/thumb_back/fw800/background/20231011/pngtree-vibrant-ice-cream-inspired-background-texture-image_13610171.png
- gstatic.com. (s. f.). [Imagen sin título]. Googleusercontent. Recuperado de https://encrypted-tbn0.gstatic.com/images?q=tbn:ANd9GcRitdtOu1d32A3S6h-kG3bxrrubTFdGmPIdHw&s
- aplaybox.com. (s. f.).【未定事件簿】莫弈/模型配布 [Modelo 3D]. Aplaybox. Recuperado de https://www.aplaybox.com/details/model/eb0lMfxlEsxq
- aplaybox.com. (s. f.).【原神】「海风之梦」琴 [Modelo 3D]. Aplaybox. Recuperado de https://www.aplaybox.com/details/model/p6wJAW4O4dar
- Outlier Spa. (s. f.). *New York Hot Dog Cart* [Modelo 3D]. Sketchfab. Recuperado de https://skfb.ly/osZGy
- Yacob. (s. f.). *Hotdog* [Modelo 3D]. Sketchfab. Recuperado de https://skfb.ly/6U7KN
- Pablo Marquez. (s. f.). *Ketchup Bottle* [Modelo 3D]. Sketchfab. Recuperado de https://skfb.ly/6X7DU
- Citron Legacy. (s. f.). *Genshin Impact Mora* [Modelo 3D]. Sketchfab. Recuperado de https://skfb.ly/o68xX

- aplaybox.com. (s. f.).【未定事件簿】椛暗式-左然·燃动潮流夜ver1.0 [Modelo 3D]. Aplaybox. Recuperado de https://www.aplaybox.com/details/model/KGIDMnIhekXY
- Sketchfab. (s. f.). *Bumper cars game attraction* [Modelo 3D]. Sketchfab. Recuperado de https://sketchfab.com/3d-models/bumper-cars-game-attraction-446af53b3340431182d87ebbf8e48bc5
- Amir Olphat. (s. f.). *Huawei Y560 4G Mobile Phone* [Modelo 3D]. Sketchfab. Recuperado de https://skfb.ly/MwXW
- aplaybox.com. (s. f.). [Modelo 3D sin título]. Aplaybox. Recuperado de https://www.aplaybox.com/details/model/Zt4NADUTkjRZ
- MaX3Dd. (s. f.). *Old Wooden Chair Low-poly* [Modelo 3D]. Sketchfab. Recuperado de https://skfb.ly/pqJAy
- Michael Alexis. (s. f.). *Bate* [Modelo 3D]. Sketchfab. Recuperado de https://skfb.ly/66tJs
- Anastacio Games. (s. f.). *Grade Grid Game* [Modelo 3D]. Sketchfab. Recuperado de https://skfb.ly/69HqN
- Spark Games. (s. f.). *Low Poly Balls* [Modelo 3D]. Sketchfab. Recuperado de https://skfb.ly/p8GuZ
- aplaybox.com. (s. f.). [Modelo 3D sin título]. Aplaybox. Recuperado de https://www.aplaybox.com/details/model/zgKs3aPD7ElX
- yuleeeee. (s. f.). *acient fan* [Modelo 3D]. Sketchfab. Recuperado de https://skfb.ly/oxoOU
- fulcrumgallery.com. (s. f.). *Bamboo Leaves I* [Imagen]. Fulcrum Gallery. Recuperado de https://www.fulcrumgallery.com/product-images/P956317-10/bamboo-leaves-i.jpg
- Berk Gedik. (s. f.). *Abandoned Toilet Cabin (Low Poly)* [Modelo 3D]. Sketchfab. Recuperado de https://skfb.ly/6VyIo
- aplaybox.com. (s. f.).【未定事件簿】莫弈-飞雪 [Modelo 3D]. Aplaybox. Recuperado de https://www.aplaybox.com/details/model/J3Z8Qtv792V1
- tasha.kosaykina. (s. f.). *3December2020 | Plant orchid flower* [Modelo 3D]. Sketchfab. Recuperado de https://skfb.ly/6Z8tB

- Voyage. (s. f.-a). *Simple stylized Cherry blossom* [Modelo 3D]. Sketchfab. Recuperado de https://skfb.ly/oGrxx
- Voyage. (s. f.-b). *Cherry blossom petal* [Modelo 3D]. Sketchfab. Recuperado de https://skfb.ly/oGrxy
- aplaybox.com. (s. f.-z). [Modelo 3D sin título]. Aplaybox. Recuperado de https://www.aplaybox.com/details/model/U7tpI2b7hNei
- aplaybox.com. (s. f.-aa). [Modelo 3D sin título]. Aplaybox. Recuperado de https://www.aplaybox.com/details/model/sKEfNYfXwqFn
- Keyotine. (s. f.). *Stylized Carnival Booth* [Modelo 3D]. Sketchfab. Recuperado de https://skfb.ly/oIyrq
- ftcdn.net. (s. f.). [Imagen de fondo de carnaval]. Adobe Stock. Recuperado de https://t3.ftcdn.net/jpg/07/30/73/34/360_F_730733455_DcJ0aOtRxn5xunWOE1Q4AxacEN7HqTRD.jpg
- plaggy. (s. f.). *CC0 - Balloon* [Modelo 3D]. Sketchfab. Recuperado de https://skfb.ly/oUTFo
- itsdevinci. (s. f.). *Low Poly Dart* [Modelo 3D]. Sketchfab. Recuperado de https://skfb.ly/oHtAD
- aplaybox.com. (s. f.-bb). [Modelo 3D sin título]. Aplaybox. Recuperado de https://www.aplaybox.com/details/model/bnX7ynEPaP1C
- aplaybox.com. (s. f.-cc). [Modelo 3D sin título]. Aplaybox. Recuperado de https://www.aplaybox.com/details/model/F58n5j7y68EQ
- Elbolillo. (s. f.). *Tacos_ Props* [Modelo 3D]. Sketchfab. Recuperado de https://skfb.ly/oYMJH
- AVIKA. (s. f.). *Avika Street Food Cart* [Modelo 3D]. Sketchfab. Recuperado de https://skfb.ly/oBTu8
- sweddle. (s. f.). *Plastic Table* [Modelo 3D]. Sketchfab. Recuperado de https://skfb.ly/ossMV
- chilango.com. (s. f.). [Imagen de rótulos de la alcaldía Cuauhtémoc]. Chilango. Recuperado de https://img.chilango.com/cdn-cgi/image/width=1200,height=675,quality=75,format=auto,onerror=redirect/2024/10/Rotulos-alcaldia-Cuauhtemoc_foto.jpg
- shutterstock.com. (s. f.). *Plain neon green solid color* [Imagen]. Shutterstock. Recuperado de

https://www.shutterstock.com/image-illustration/plain-neon-green-solid-color-260nw-1667301652.jpg

- Nathalie Michel. (s. f.). *Big Knife* [Modelo 3D]. Sketchfab. Recuperado de https://skfb.ly/6sURV
- aplaybox.com. (s. f.-dd). [Modelo 3D sin título]. Aplaybox. Recuperado de https://www.aplaybox.com/details/model/7Ox4GS5NB2MF
- aplaybox.com. (s. f.-ee). [Modelo 3D sin título]. Aplaybox. Recuperado de https://www.aplaybox.com/details/model/ox8L165kTbDq
- pinimg.com. (s. f.-a). [Imagen sin título]. Pinterest. Recuperado de https://i.pinimg.com/236x/d9/13/48/d91348a831a52b128c5b5aacaca6ee10.jpg
- istockphoto.com. (s. f.). *Circo carnaval retro rayas vintage con estrellas patrón sin costuras plantilla gráfica* [Ilustración]. iStock. Recuperado de https://media.istockphoto.com/id/1212893750/es/vector/circo-carnaval-retro-rayas-vintage-con-estrellas-patr%C3%B3n-sin-costuras-plantilla-gr%C3%A1fica.jpg?s=170667a&w=0&k=20&c=nXl_aY9Ad-hgjyxer7KxCeojMB26gpJA-QugSXEnUg8=
- aplaybox.com. (s. f.-ff). [Modelo 3D sin título]. Aplaybox. Recuperado de https://www.aplaybox.com/details/model/0ygLuNRGnO1z
- pngwing.com. (s. f.). [Imagen sin título]. PNGwing. Recuperado de https://www.pngwing.com/es/free-png-zyqxw/download
- pinimg.com. (s. f.-b). [Imagen sin título]. Pinterest. Recuperado de https://i.pinimg.com/736x/5a/74/41/5a7441c392720453d416ce531030d173.jpg
- Gamedirection. (s. f.). *Park Table - Low Poly* [Modelo 3D]. Sketchfab. Recuperado de https://skfb.ly/6CTuW
- aplaybox.com. (s. f.). [Modelo 3D sin título]. Aplaybox. Recuperado de https://www.aplaybox.com/details/model/qTXzeDkYaiBu
- aplaybox.com. (s. f.). [Modelo 3D sin título]. Aplaybox. Recuperado de https://www.aplaybox.com/details/model/hzLyyqDYqYNp
- Sketchfab. (s. f.). *Honkai: Star Rail - Ruan Mei Creations Cat Cake* [Modelo 3D]. Sketchfab. Recuperado de https://sketchfab.com/3d-models/honkai-star-rail-ruan-mei-creations-cat-cake-f700f314430043c99bc700185f3ab0a2

- aplaybox.com. (s. f.). [Modelo 3D sin título]. Aplaybox. Recuperado de https://www.aplaybox.com/details/model/e0kNzNem9ZgQ
- aplaybox.com. (s. f.). [Modelo 3D sin título]. Aplaybox. Recuperado de https://www.aplaybox.com/details/model/txXgqWpAVzpr
- aplaybox.com. (s. f.). [Modelo 3D sin título]. Aplaybox. Recuperado de https://www.aplaybox.com/details/model/YquJuZCT6e5w
- aplaybox.com. (s. f.). [Modelo 3D sin título]. Aplaybox. Recuperado de https://www.aplaybox.com/details/model/EZXu8LRkeXY6
- aplaybox.com. (s. f.). [Modelo 3D sin título]. Aplaybox. Recuperado de https://www.aplaybox.com/details/model/j4kat4jyR9bV
- aplaybox.com. (s. f.). [Modelo 3D sin título]. Aplaybox. Recuperado de https://www.aplaybox.com/details/model/ry7ZgsrgLE4Z
- aplaybox.com. (s. f.). [Modelo 3D sin título]. Aplaybox. Recuperado de https://www.aplaybox.com/details/model/nxFIrs3fVl3Q
- aplaybox.com. (s. f.). [Modelo 3D sin título]. Aplaybox. Recuperado de https://www.aplaybox.com/details/model/1zHFnlSmjS2V
- aplaybox.com. (s. f.). [Modelo 3D sin título]. Aplaybox. Recuperado de https://www.aplaybox.com/details/model/gByJebJf5M2r
- aplaybox.com. (s. f.). [Modelo 3D sin título]. Aplaybox. Recuperado de https://www.aplaybox.com/details/model/391cKeeMyed8
- Gamedirection. (s. f.). *Park Table - Low Poly* [Modelo 3D]. Sketchfab. Recuperado de https://skfb.ly/6CTuW
- aplaybox.com. (s. f.). [Modelo 3D sin título]. Aplaybox. Recuperado de https://www.aplaybox.com/details/model/zAaIgNTApJ0a
- hoyolab.com. (s. f.). [Artículo sin título]. HoYoLAB. Recuperado de https://www.hoyolab.com/article/12096914
- tot.wiki. (s. f.). [Imagen de Luke Halloween 2024]. Tears of Themis Wiki. Recuperado de https://tot.wiki/thumb.php?f=2024_Halloween_Luke.jpg&width=900
- tot.wiki. (s. f.). [Imagen de Vyn Halloween 2024]. Tears of Themis Wiki. Recuperado de https://tot.wiki/thumb.php?f=2024_Halloween_Vyn.jpg&width=900

- tot.wiki. (s. f.). [Imagen de Artem Halloween 2024]. Tears of Themis Wiki. Recuperado de https://tot.wiki/thumb.php?f=2024_Halloween_Artem.jpg&width=900
- tot.wiki. (s. f.). [Imagen de Marius Halloween 2024]. Tears of Themis Wiki. Recuperado de https://tot.wiki/thumb.php?f=2024_Halloween_Marius.jpg&width=900
- hoyolab.com. (2024, octubre 31). [Imagen sin título]. HoYoLAB. Recuperado de https://upload-os-bbs.hoyolab.com/upload/2024/10/31/156961407/112f8e033cd1f9e031ffdeb915720faa_4847376674870410934.png?x-oss-process=image%2Fresize%2Cs_1000%2Fauto-orient%2C0%2Finterlace%2C1%2Fformat%2Cwebp%2Fquality%2Cq_70
- hoyolab.com. (2024, febrero 12). [Imagen sin título]. HoYoLAB. Recuperado de https://upload-os-bbs.hoyolab.com/upload/2024/02/12/235944470/1d8a9e5788d78775328ddb7a05442998_7814680540134215415.jpg?x-oss-process=image%2Fresize%2Cs_1000%2Fauto-orient%2C0%2Finterlace%2C1%2Fformat%2Cwebp%2Fqualit
- hoyolab.com. (s. f.). [Artículo sin título]. HoYoLAB. Recuperado de https://www.hoyolab.com/article/12096914
- tot.wiki. (s. f.). *Whac-A-Mole*. Tears of Themis Wiki. Recuperado de https://tot.wiki/wiki/Whac-A-Mole