



UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO
FACULTAD DE INGENIERÍA
DIVISIÓN DE INGENIERÍA ELÉCTRICA
INGENIERÍA EN COMPUTACIÓN
COMPUTACIÓN GRÁFICA e INTERACCIÓN HUMANO
COMPUTADORA



TECHNICAL MANUAL

Account Number: 319111347

THEORY GROUP: 05

SEMESTER: 2025-2

DEADLINE: 20/05/2025

GRADE: _____

TECHNICAL MANUAL

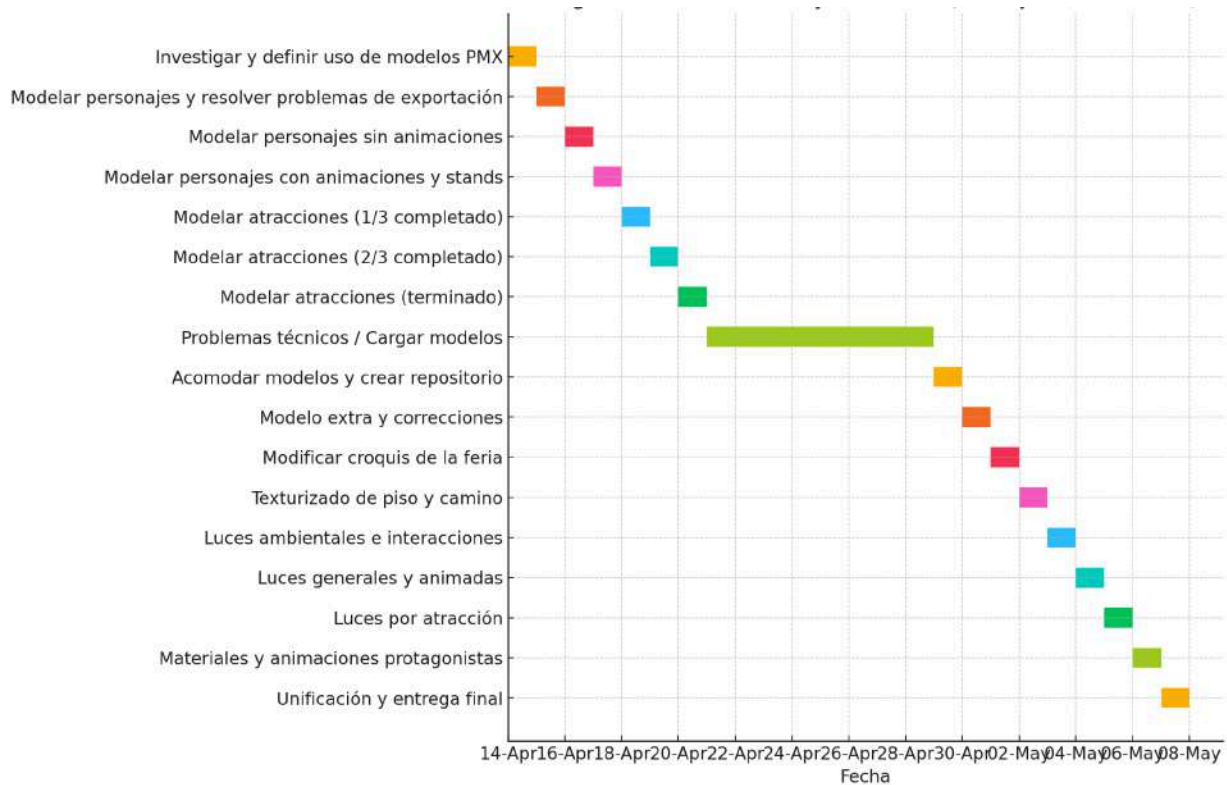
DESCRIPTION AND OBJECTIVE

The project consists of a scenario centered around the theme: "Skill Game Fair". This means a scene will be created with the main theme being a fair that can be freely explored or that can have the camera focused on different attractions when interacted with.

SCOPE

- **Complete themed environment:** The environment will be built to resemble a realistic fair, with paths, decorations, lighting, and ambiance suitable to the theme.
- **Third-person navigation:** A third-person camera system will be implemented that follows a character as they move freely through the environment.
- **Interaction with stands:** Pressing certain keys will activate functionality that automatically reframes the camera onto a stand, focusing on it and triggering an animation related to that stand.
- **Lighting and ambiance:** A lighting system with light sources will be included to highlight key areas of the scene.
- **NPC animations:** Some NPCs will have non-interaction-dependent animations, meaning they will always be performing their actions.
- **Art style:** The fair will be themed in the style of the video games *Honkai: Star Rail*, *Genshin Impact*, and *Tears of Themis*.

GANTT CHART:



COST ANALYSIS:

1. Team cost

It is estimated that the full development of the project took a total of **150 hours**. These hours include tasks such as:

- Modeling objects in Blender
- Integration with OpenGL
- Programming the third-person camera
- Lighting and shading setup
- Basic real-time interaction
- Testing, debugging, and final adjustments

Estimated cost per hour: \$150 MXN/hour (considering technical knowledge, software, electricity, computing equipment, and internet connection).

Base cost = 150 hours × 150 MXN/hour = **\$22,500 MXN**

2. Indirect costs

An additional 10% is considered for indirect costs such as:

- Power consumption
- Equipment wear
- Internet and other auxiliary services

Cost including indirects = $22,500 \text{ MXN} \times 1.10 = \$24,750 \text{ MXN}$

3. Sale price

To establish a fair and sustainable sale price, a **25% profit margin** is considered, which includes:

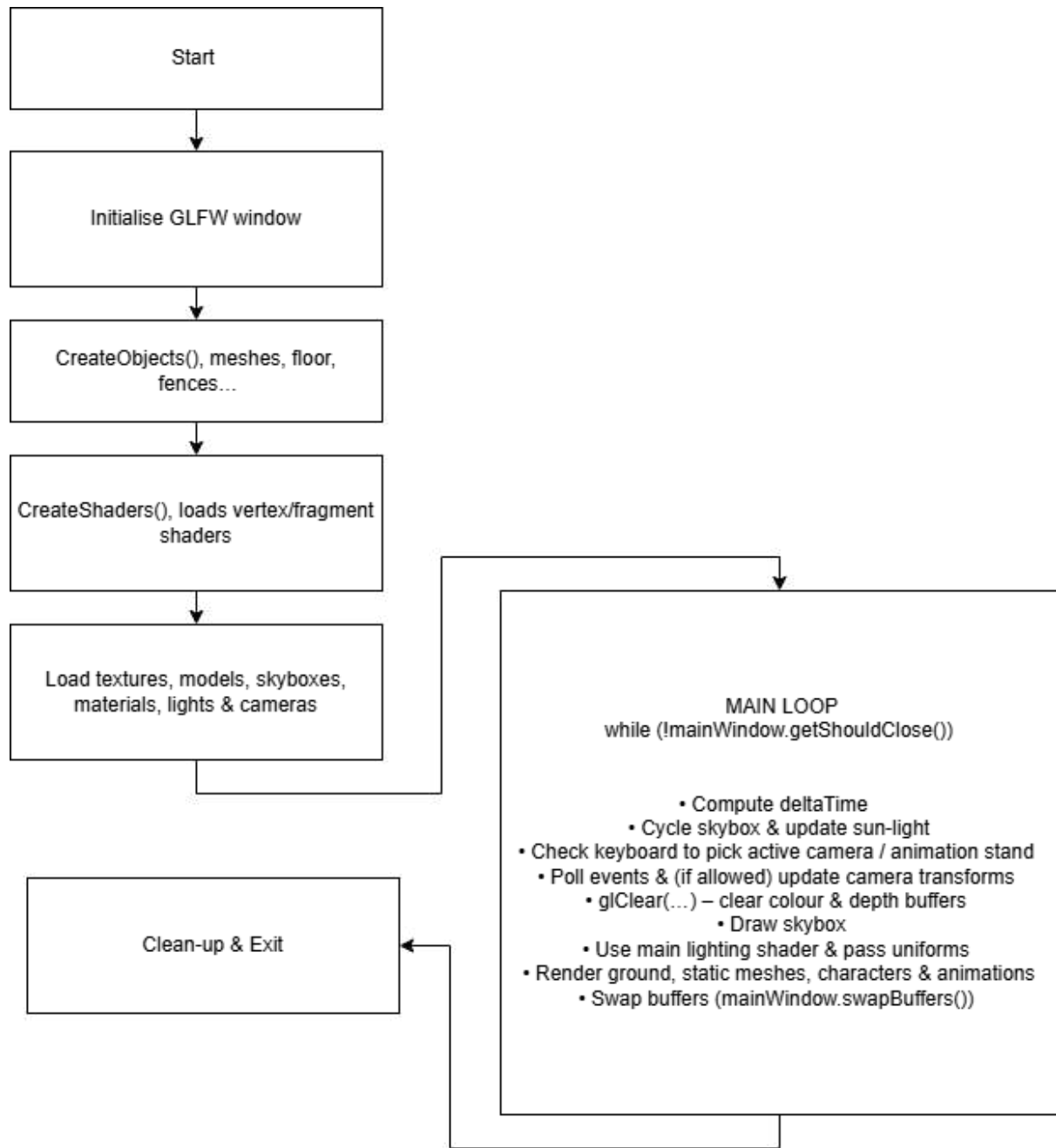
- The value of specialized knowledge
- Time invested in research and testing

Final price = $24,750 \text{ MXN} \times 1.25 = \$30,937.50 \text{ MXN}$

The final price of \$31,000 MXN is based on:

- The real time invested by the team (150 hours)
- The required technical expertise (programming, 3D design, OpenGL)
- Use of computing tools and resources

FLOWCHART



FUNCTION DICTIONARY:

Model.cpp

- **Model::LoadModel**
Loads a 3D model from a file using Assimp, triangulates and processes vertices, then loads nodes and materials.
- **Model::ClearModel**
Frees memory of all loaded meshes and textures.
- **Model::RenderModel**
Renders all the meshes of the model applying their respective textures.
- **Model::LoadNode**
Recursively traverses the model's nodes and loads each mesh.
- **Model::LoadMesh**
Extracts vertices, normals, texture coordinates, and indices from an Assimp mesh and creates an OpenGL mesh.
- **Model::LoadMaterials**
Loads and assigns the appropriate textures to the model's materials. Uses a default texture if a texture is missing.

PointLight.cpp

- **PointLight::PointLight (default constructor)**
Initializes a point light with neutral/default values.
- **PointLight::PointLight (with parameters)**
Initializes a point light with custom color, intensity, and position.
- **PointLight::UseLight**
Sends the light's values (color, intensity, position, attenuation) to the shaders.
- **PointLight::SetLight**
Updates the color and position of the light.
- **PointLight::SetIntensity**
Updates the ambient and diffuse intensities.

SpotLight.cpp

- **SpotLight::SpotLight (default constructor)**
Initializes a spotlight facing downward with no cutoff angle.
- **SpotLight::SpotLight (with parameters)**
Initializes a spotlight with custom color, intensity, direction, position, and cutoff edge.
- **SpotLight::UseLight**
Sends all necessary spotlight parameters to the shader, including direction and angle.
- **SpotLight::setFlash**
Updates the light's position and direction (useful for flashlight-style lighting).
- **SpotLight::SetColor**
Updates the light's color.
- **SpotLight::SetPos**
Updates the light's position.

Window.cpp

- **Window::Window (constructors)**
Initializes the window with default or custom dimensions, and resets key states and animation variables.
- **Window::Initialise**
Initializes GLFW, creates the window, sets up OpenGL context, enables GLEW and depth buffering.
- **Window::createCallbacks**
Sets up callback functions to handle keyboard input.

MODELS:

The selected character models were found in .pmx format, which, although challenging at first due to unfamiliarity with the format, made posing the characters easier since they didn't need to be separated for movement. Instead, the built-in "bones" of the models were used.

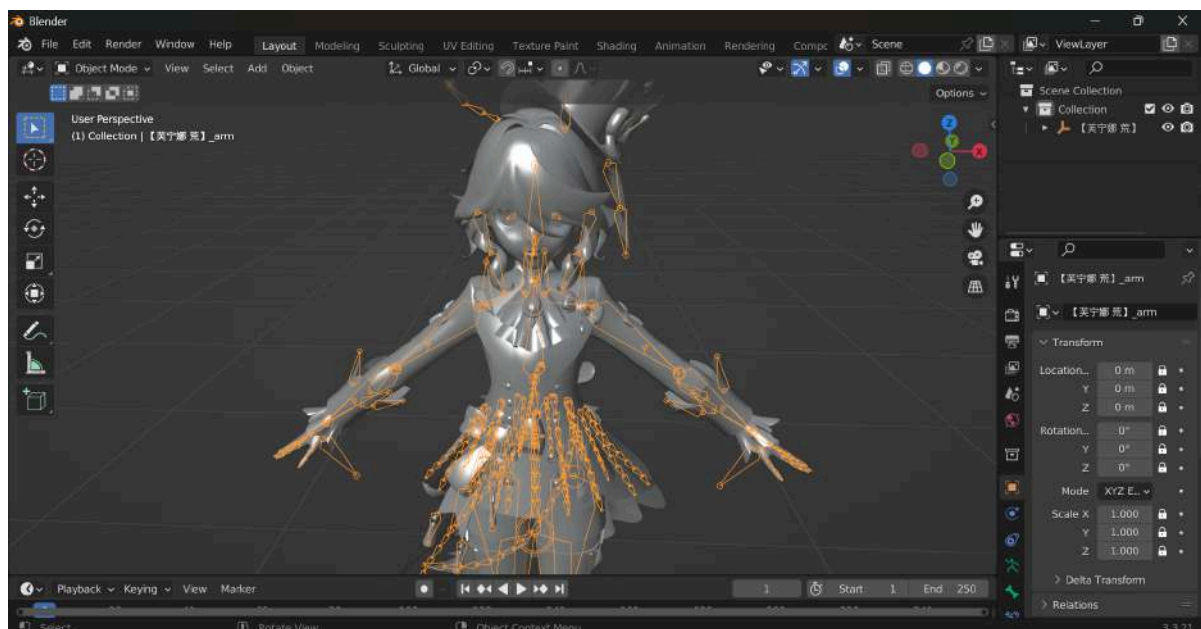
There were some difficulties due to the models, their textures, and components being in Chinese. However, a method to work with them was eventually found.

Instructions from a YouTube video were followed to import them into Blender (version 3.3) using a plugin called MMD.

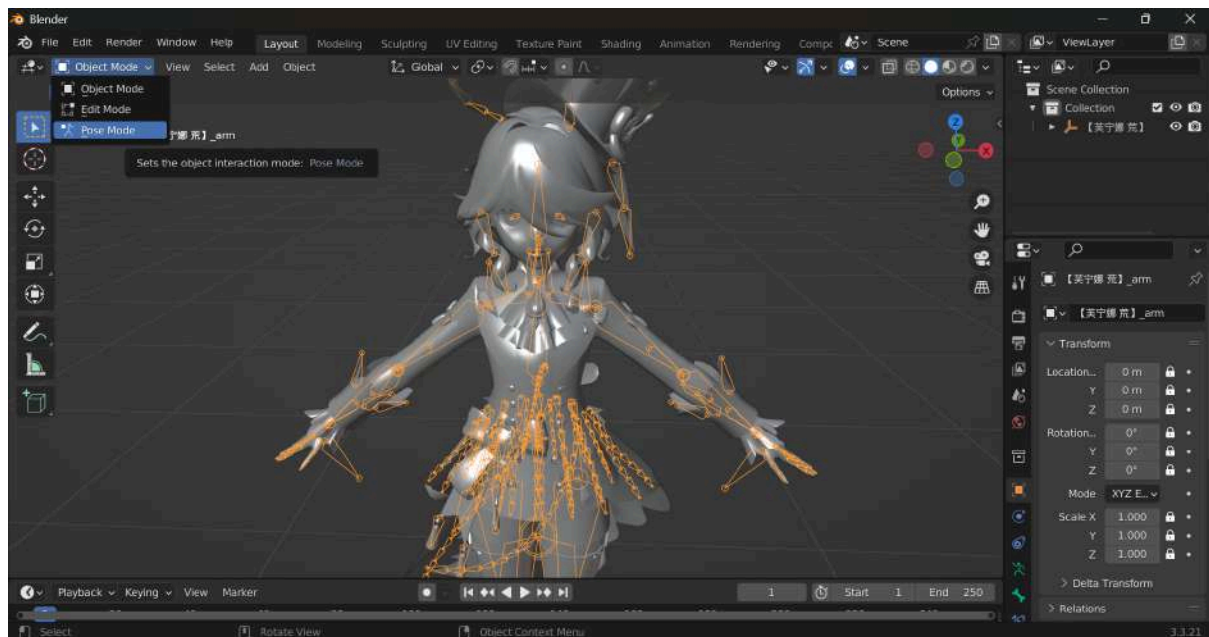
Video link: <https://www.youtube.com/watch?app=desktop&v=OgS31Q9KkDI>

MMD repository link: https://github.com/MMD-Blender/blender_mmd_tools/tree/main

Once opened in Blender, the models can be repositioned using MMD tools, which are designed for this purpose.



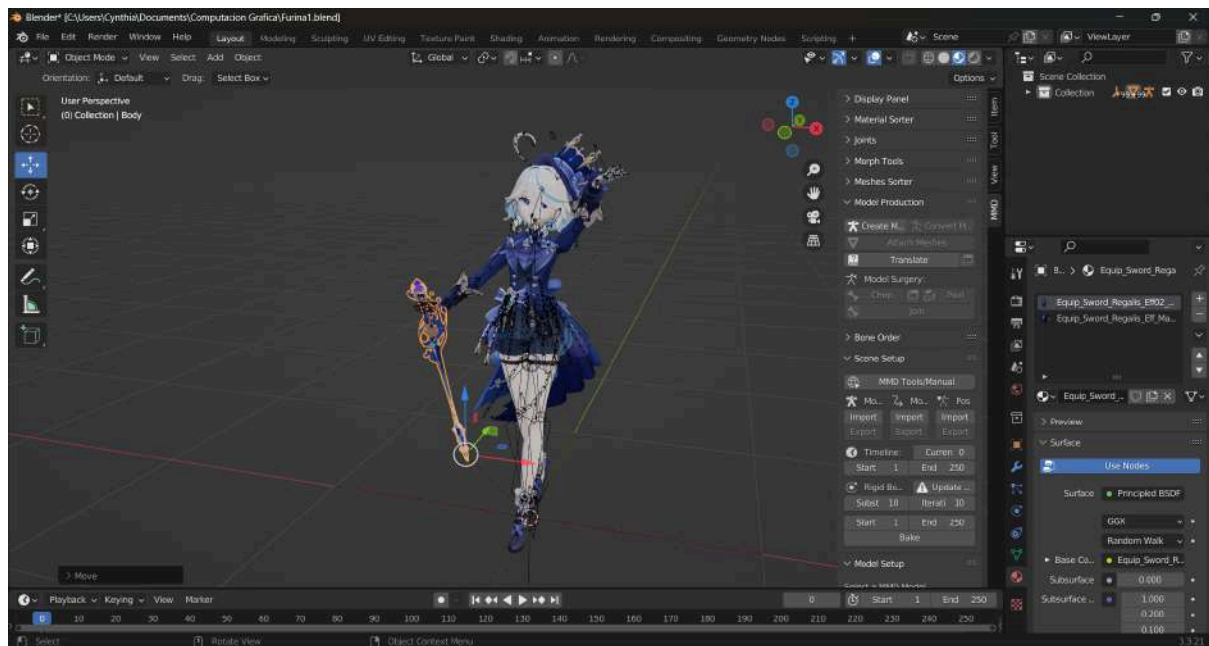
To do this, the skeleton is selected, and the mode is changed to Pose Mode:



In this mode, you can select a bone and rotate it by pressing G (grab), followed by X, Y, or Z to rotate it along that axis. Click outside the bone to finalize the movement.

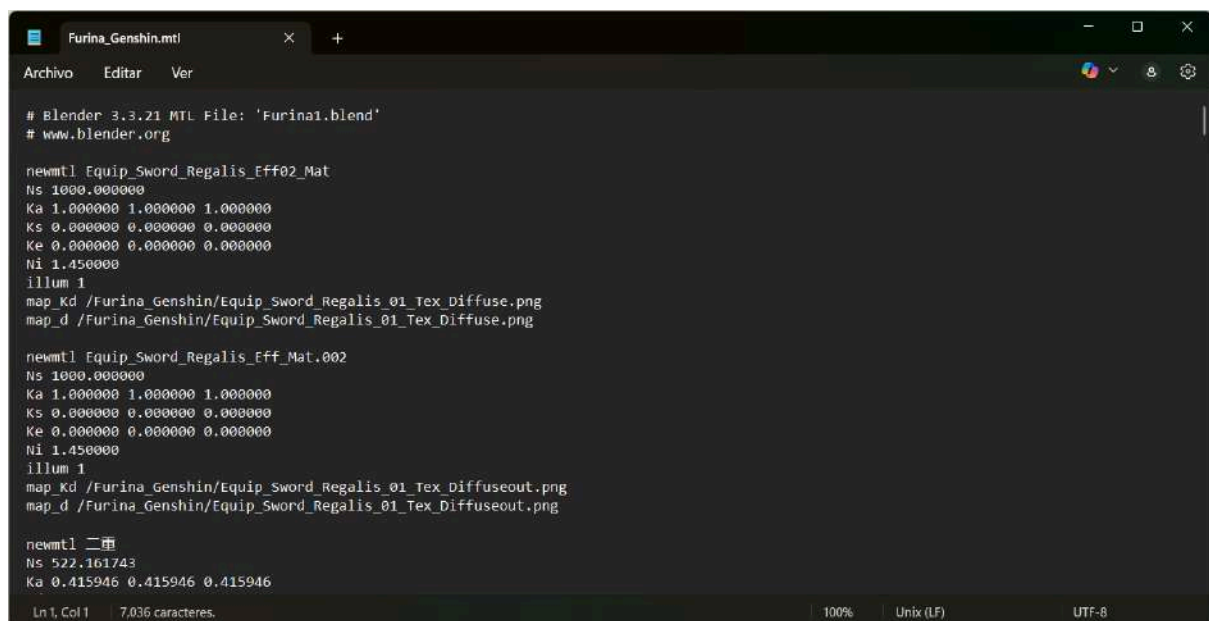


After the pose is defined, extra objects accompanying the model are added.

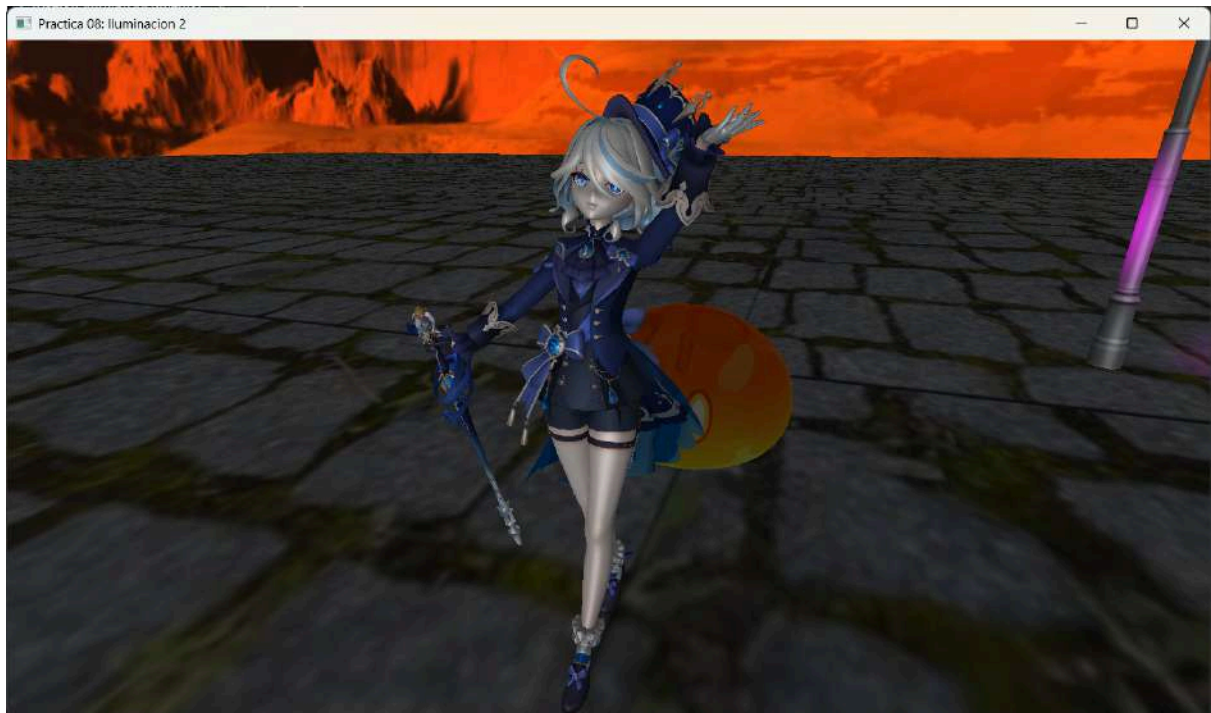


Finally, the model is exported as a .obj file. Since it won't move, it becomes a single unified model. Texture values are added manually because the .pmx to .obj conversion does not preserve textures. This is handled in the final .mtl file.

Note: Some texture names are in Chinese and not properly recognized by the program, so they must be renamed. Another approach is to texture the model in 3ds Max and then export.



The final result is a visualizable model in the program.



Since the characters are based on official and well-aligned fan-made models from the three game universes, they already share a consistent scale. All were scaled up by x5 to match the desired project size, and these became the reference for all other models.

Stands were created using online models and were arranged, retextured when necessary, scaled, and positioned before exporting (especially when the character model and intended stand location were already known to avoid extra transformations).

Skyboxes and Day-Night Cycle:

Day:



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

Dusk:



Night:



<https://gamebanana.com/mods/7921>

Each image was cropped and renamed for clarity. Then, using GIMP, the alpha channel was removed and the images were scaled to 512x512. Finally, they were called in the program.

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

std::vector<std::string> skyboxFaces; //Se crea un vector con las texturas que componen al skybox de día
//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");
```

These are declared before the while loop so they become global variables.


```

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 día

    //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

```

The if statements go inside the while loop. In the code, each action and its function is commented. In short, a counter variable determines when to switch between the day, dusk, or night skybox.

CAMERA SYSTEM (THIRD-PERSON VIEW)

A camera system was implemented that initializes four cameras (including one free camera). Each starts at a predefined position.

The aerial camera was created by adjusting the look direction value to 90 (forward) and -90 (downward).

```

// -----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);

```

Then, within the while loop, there are a series of if statements that determine whether the camera can be changed using a key (controlled by the banderaCamara flag). This ensures that when the stand camera is active, the user cannot exit the animation or switch cameras using any key or the mouse. Inside that if, there are others that determine which camera to use when a certain key is pressed, using joints. For the avatar-following camera and the free camera, the camera is not overwritten as it is with the stands, since these cameras need to keep pointing where they last were and not be reinitialized. On the other hand, the stand camera is a single one whose initial values are overwritten to always look at the same place.

In the stands, the variable camaraAnimacion is changed so that its value can be passed to the lights and animations, allowing them to know which animation to initiate..

```
// -----  
// ----- 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(-83.0f, 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;  
    }  
}
```

Finally, with additional if statements, the window is cleared and "recreated." For the free and avatar cameras, the keyboard movement (for the free camera) and mouse movement (for both) remain enabled.

```
//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.getKeys(), 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.getKeys(), 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);
}
```

For Stand Camera 3, these functionalities were removed so that the user cannot move the camera using the keyboard or the mouse, and it remains fixed in the position it was set.

```
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);
}
```


Avatar camera (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));
```

To solve the problem of having the camera follow the character Blade, the following was done:

First, the direction of the camera is obtained and projected onto the XZ (horizontal) plane to avoid affecting the Y-axis. Pressed keys (e.g., WASD) are detected, and the character is moved in relation to the camera's forward direction (cameraForward) and its right vector.

An angle is calculated using atan2 so that Blade rotates to face the direction the camera is pointing. Transformations are applied to the model to move it to its new position and rotation. A relative position behind and above the character is then defined so the camera always follows.

As a result, the character always moves in the direction the camera is facing. The camera stays behind the character, updating its position each frame. When moving the mouse, the character rotates and the camera follows.

The files camera.cpp and camera.h were also modified to achieve this.

Camera.cpp:

```
22 void Camera::setPosition(glm::vec3 newPos) {  
23     position = newPos;  
24 }  
25  
26 void Camera::setFront(glm::vec3 newFront) {  
27     front = newFront;  
28     update();  
29 }
```

Camera.h:

```
28 void setPosition(glm::vec3 newPos);  
21 void setFront(glm::vec3 newFront);
```

ANIMATION:

Stands:

For the animation of the stands, a variable is used that changes depending on the stand camera, so that when this variable matches the stand number, the animation starts. When the animation ends, the models inside it disappear and are no longer rendered in the world.

First, an if checks which stand was triggered by a key press, then camera changes are locked using the flag banderaCamara = 1 (0 allows switching cameras).

Inside, another if checks the stand timer (tiempoBateo), which is calculated by summing with deltaTime at the end of the if. This time marks the duration of the camera and animation. When the timer ends, the animation finishes, and the camera returns to the avatar's.

First, a coin delivery animation starts, which changes the coin based on the stand's universe (since each stand belongs to a different game universe).

```
//
// -----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) { //Tiempo 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)); //Cx -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
```

Once the coin animation ends, the individual stand animation begins, controlled by specific ifs and 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;
}

}
```

The full animation code is not shown due to length. However, at the end of the timer if, there is an else that resets all used variables to their initial state, so they behave as expected when the key is pressed again. In the image, you can see a final if-else that starts the next stand's animation.

```
    }  
    }  
    ///   
    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) { //Tiempo que va a durar la animación general del stand
```

Sine-based animations:

For the project, both Blade and the trash can use sine functions to simulate the required movements to make them appear as if they are walking, with slight differences.

For Blade full sine values are used for legs and shoulders. Only positive sine values are used for forearms and calves to avoid unnatural movement (e.g., elbows bending the wrong way).

```
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;  
}
```

Blade also includes logic to return to the original pose once walking stops and the animation is not finished, to avoid abrupt transitions.

```
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;
}
```

For the trash can the same logic applies, but since it moves continuously, there's no need for the reset logic. As its animation is infinite, modifications were added for it to move back and forth. A sine function is used, and the program detects whether the sine is rising or falling to assign the direction.

```
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;
```

Main and NPC animations:

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
}

```

In this case, a character moves their forearm to simulate scooping ice cream. The ball (a Genshin Impact slime) spins to appear as if it's being created, then is placed as an ice cream scoop. The animation continues until three scoops are made and then resets.

In general, rotate and translate functions were used.

LIGHTING AND MATERIALS:

Two main light arrays were created: day and night lights. For day lights: Two elements are moved via the keyboard, requiring 4 conditionals. For night lights: A single light is used, requiring 2 conditionals. Additionally, 12 arrays were created—one for each attraction for both day and night.

```

// -----
// -----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];

```

Day/night cycles used skybox conditionals and modified light elements.


```

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

```

To animate the lights in the fair, counters and conditionals were used to modify light values. This applies to the Ferris wheel, carousel, bumper cars, ticket booth, lamps, batting and mole stands.

```

// -----
// -----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;

```

Implemented materials:

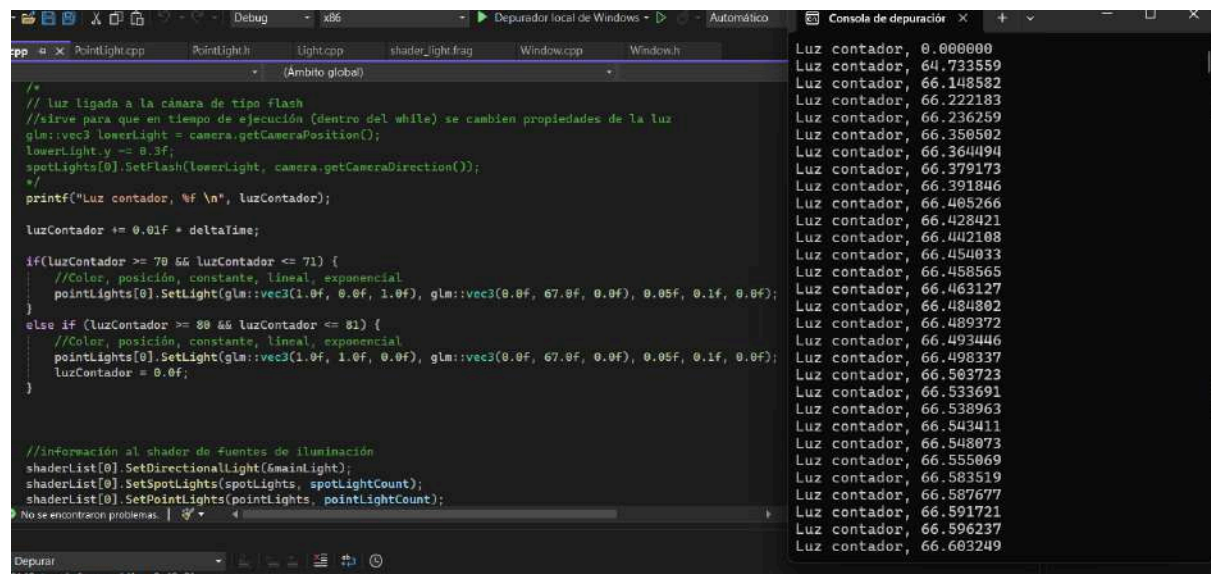
In addition to the two materials provided by the professor, the following were used:

- 3D Cartoon Character (Avatar): Specular Intensity: 0.1, Shininess: 10 — Matte, non-realistic look.
- Wood: Specular Intensity: 0.4, Shininess: 10 — Low reflectivity surface.
- Metal Fence: Specular Intensity: 0.8, Shininess: 50 — Highly reflective.
- Velvet: Specular Intensity: 0.1, Shininess: 5 — Very low reflection.
- Polished Wood: Specular Intensity: 0.7, Shininess: 50 — Noticeable reflections.
- Plastic: Specular Intensity: 0.8, Shininess: 30 — More reflective than regular wood, less than polished wood.

```
// -----  
// -----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);
```

ISSUES ENCOUNTERED:

deltaTime Issues:



```
/*
// luz ligada a la cámara de tipo flash
//sirve para que en tiempo de ejecución (dentro del while) se cambien propiedades de la luz
glm::vec3 lowerLight = camera.getCameraPosition();
lowerLight.y -= 0.3f;
spotlights[0].setFlash(lowerLight, camera.getCameraDirection());
*/
printf("Luz contador, %f \n", luzContador);

luzContador += 0.01f * deltaTime;

if(luzContador >= 70 && luzContador <= 71) {
    //Color, posición, constante, lineal, exponencial
    pointlights[0].SetLight(glm::vec3(1.0f, 0.0f, 1.0f), glm::vec3(0.0f, 67.0f, 0.0f), 0.05f, 0.1f, 0.0f);
}
else if (luzContador >= 80 && luzContador <= 81) {
    //Color, posición, constante, lineal, exponencial
    pointlights[0].SetLight(glm::vec3(1.0f, 1.0f, 0.0f), glm::vec3(0.0f, 67.0f, 0.0f), 0.05f, 0.1f, 0.0f);
    luzContador = 0.0f;
}

//Información al shader de fuentes de iluminación
shaderList[0].SetDirectionalLight(&mainLight);
shaderList[0].SetSpotlights(spotlights, spotLightCount);
shaderList[0].SetPointlights(pointlights, pointLightCount);

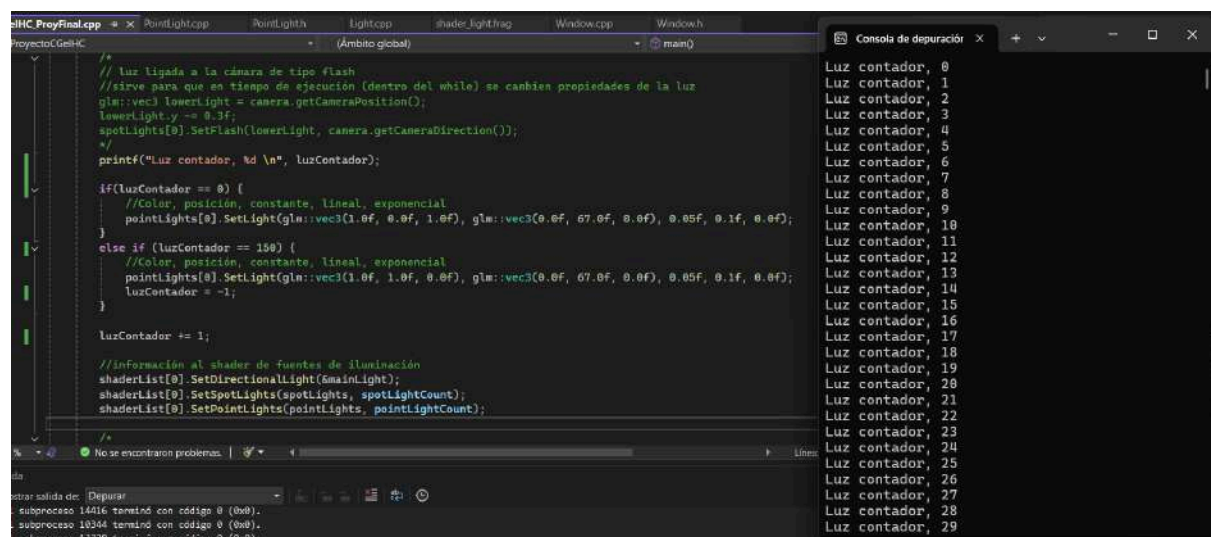
Depurar
```

Consola de depuración

```
Luz contador, 0.000000
Luz contador, 64.733559
Luz contador, 66.148582
Luz contador, 66.222183
Luz contador, 66.236259
Luz contador, 66.358502
Luz contador, 66.364494
Luz contador, 66.379173
Luz contador, 66.391846
Luz contador, 66.405266
Luz contador, 66.428421
Luz contador, 66.442108
Luz contador, 66.454033
Luz contador, 66.458565
Luz contador, 66.463127
Luz contador, 66.484802
Luz contador, 66.489372
Luz contador, 66.493446
Luz contador, 66.498337
Luz contador, 66.503723
Luz contador, 66.533691
Luz contador, 66.538963
Luz contador, 66.543411
Luz contador, 66.548073
Luz contador, 66.555069
Luz contador, 66.583519
Luz contador, 66.587677
Luz contador, 66.591721
Luz contador, 66.596237
Luz contador, 66.603249
```

For initial animations (which require elapsed time tracking), deltaTime showed massive intervals at program start, even when multiplied by 0.01, making it difficult to work with consistent values. In such cases, counters were used instead (e.g., for lights and skybox).

Difference: Counters start at 0 and update consistently.



```
/*
// luz ligada a la cámara de tipo flash
//sirve para que en tiempo de ejecución (dentro del while) se cambien propiedades de la luz
glm::vec3 lowerLight = camera.getCameraPosition();
lowerLight.y -= 0.3f;
spotlights[0].setFlash(lowerLight, camera.getCameraDirection());
*/
printf("Luz contador, %d \n", luzContador);

if(luzContador == 0) {
    //Color, posición, constante, lineal, exponencial
    pointlights[0].SetLight(glm::vec3(1.0f, 0.0f, 1.0f), glm::vec3(0.0f, 67.0f, 0.0f), 0.05f, 0.1f, 0.0f);
}
else if (luzContador == 150) {
    //Color, posición, constante, lineal, exponencial
    pointlights[0].SetLight(glm::vec3(1.0f, 1.0f, 0.0f), glm::vec3(0.0f, 67.0f, 0.0f), 0.05f, 0.1f, 0.0f);
    luzContador = -1;
}

luzContador += 1;

//Información al shader de fuentes de iluminación
shaderList[0].SetDirectionalLight(&mainLight);
shaderList[0].SetSpotlights(spotlights, spotLightCount);
shaderList[0].SetPointlights(pointlights, pointLightCount);

/*
*/

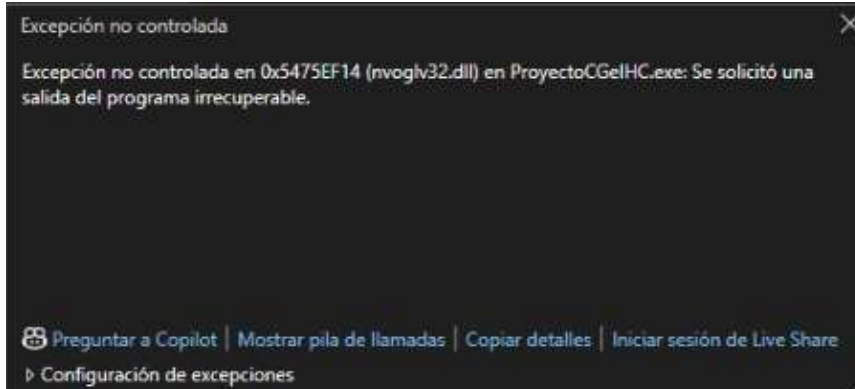
Depurar
```

Consola de depuración

```
Luz contador, 0
Luz contador, 1
Luz contador, 2
Luz contador, 3
Luz contador, 4
Luz contador, 5
Luz contador, 6
Luz contador, 7
Luz contador, 8
Luz contador, 9
Luz contador, 10
Luz contador, 11
Luz contador, 12
Luz contador, 13
Luz contador, 14
Luz contador, 15
Luz contador, 16
Luz contador, 17
Luz contador, 18
Luz contador, 19
Luz contador, 20
Luz contador, 21
Luz contador, 22
Luz contador, 23
Luz contador, 24
Luz contador, 25
Luz contador, 26
Luz contador, 27
Luz contador, 28
Luz contador, 29
```

Duplicate Texture Loading Error:

While loading models, a problem occurred when too many were loaded at once—despite working individually.



Testing on different PCs revealed that, On NVIDIA GPUs, the project crashed. On AMD GPUs, the project loaded but only a few textures rendered randomly. The issue was inferred to be due to texture duplication, to solve this, a map structure was added to check for existing textures in the code. If a texture exists, it is reused. If not, it is saved. This prevents the same texture from being stored multiple times.

```

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();
        }
    }
}

```


FINAL MAP:



 Tears of Themis
 Honkai: Star Rail
 Genshin Impact

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

KEY BINDINGS:

F → Avatar Camera

Q → Extra “Free” Camera

G → Aerial Camera

H → Axe Stand

J → Bowling Stand

K → Dice Stand

L → Batting Stand

M → Darts Stand

N → Mole Stand

O → Inazuma Stand Light

P → Hot Dog Stand Light

COMPARISON: INITIAL PROPOSAL VS. FINAL RESULT

1. Ticket Booth



2. Cotton Candy Stand



3. Dice Throw Stand



4. Portable Toilets



5. Tables



6. Whack-a-mole



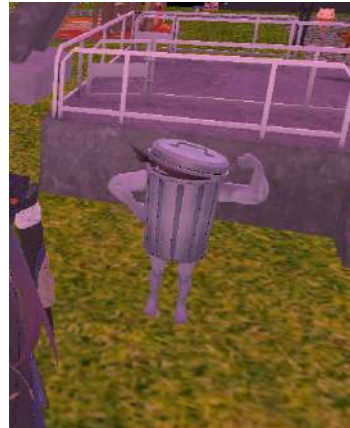
7. Dart Throw Stand



8. Streetlamp



9. Trash Cans from Honkai: Star Rail



10. Bench

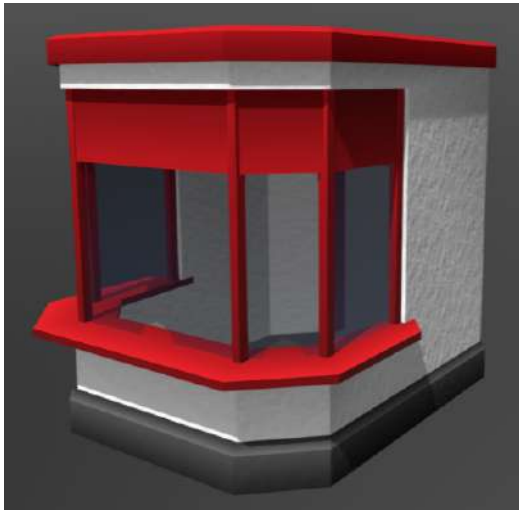


CONCLUSIONS

With the completion of this project, all the topics covered throughout the semester were incorporated, demonstrating my ability to handle both the theoretical and practical aspects. There were many challenges, as it was a very ambitious project; however, they were overcome, and I achieved a result I am satisfied with. I was able to clearly apply the use of models, their texturing, animation, and the correct use of lighting to make the fair look lively. It was also valuable because we learned many things about computer graphics and how ideas are expressed through a screen and a program.

REFERENCES FOR USED ELEMENTS

Proposal models:



Sketchfab. (n.d.-a). Ticket booth [3D model]. Sketchfab. Retrieved from [https://sketchfab.com/3d-models/ticket-booth-7e12559eed534b5699c4262e7e1a6bb](https://sketchfab.com/3d-models/ticket-booth-7e12559eed534b5699c4262e7e1a6bb1)

1



Sketchfab. (n.d.-b). Cotton candy machine [3D model]. Sketchfab. Retrieved from <https://sketchfab.com/3d-models/cotton-candy-machine-971fd0043c594d1b8493f4d0f93e24b7>



thingiverse.com. (n.d.). 8 foot craps table [3D model]. Thingiverse. Retrieved from <https://www.thingiverse.com/thing:3606436>



Berk Gedik. (n.d.). Abandoned Toilet Cabin (Low Poly) [3D model]. Sketchfab. Retrieved from <https://skfb.ly/6Vylo>



Gamedirection. (n.d.-a). Park Table - Low Poly [3D model]. Sketchfab. Retrieved from <https://skfb.ly/6CTuW>



Sketchfab. (n.d.-c). Whack-a-mole machine [3D model]. Sketchfab. Retrieved from <https://sketchfab.com/3d-models/whack-a-mole-machine-ba599fb0184b4d5286ef53f0e8d1bb65>



Keyotone. (n.d.). Stylized Carnival Booth [3D model]. Sketchfab. Retrieved from <https://skfb.ly/olyrq>



avsteir. (n.d.). HW2 10 props Genshin Impact [3D model]. Sketchfab. Retrieved from <https://skfb.ly/oAuUn>



aplaybox.com. (n.d.-a). 【崩坏：星穹铁道】王下一桶 [3D model]. Aplaybox. Retrieved from <https://www.aplaybox.com/details/model/qTXzeDkYaiBu>



avsteir. (n.d.). HW2 10 props Genshin Impact [3D model]. Sketchfab. Retrieved from <https://skfb.ly/oAuUn>

Models used in the other attractions, layout, and references.

- avsteir. (n.d.). *HW2 10 props Genshin Impact* [3D model]. Sketchfab. Retrieved from <https://skfb.ly/oAuUn>
- *Free png.* (n.d.). PNGwing. Retrieved from <https://www.pngwing.com/en/free-png-dcabk/download>
- *Mods.* (n.d.). GameBanana. Retrieved from <https://gamebanana.com/mods/7921>
- aplaybox.com. (n.d.-a). 【崩坏:星穹铁道】帕姆 [3D model]. Aplaybox. Retrieved from <https://www.aplaybox.com/details/model/R50C73gabDfS>
- Sketchfab. (n.d.-a). *Ticket booth* [3D model]. Sketchfab. Retrieved from <https://sketchfab.com/3d-models/ticket-booth-7e12559eed534b5699c4262e7e1a6bb1>
- iStockphoto. (n.d.). *Circo carnaval retro rayas vintage con estrellas patrón sin costuras plantilla gráfica* [Illustration]. iStock. Retrieved from 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=nXI_aY9Ad-hgjyxr7KxCeojMB26gpJA-QugSXEnUg8=
- Sketchfab. (n.d.-b). *Cotton candy machine* [3D model]. Sketchfab. Retrieved from <https://sketchfab.com/3d-models/cotton-candy-machine-971fd0043c594d1b8493f4d0f93e24b7>
- aplaybox.com. (n.d.-b). 【崩坏:星穹铁道】知更鸟 [3D model]. Aplaybox. Retrieved from <https://www.aplaybox.com/details/model/ZBeEOboYGD6z>
- algodoin.es. (2021, March). *ALGODÓN AZÚCAR HEADER* [Image]. Algodoin. Retrieved from <https://algodonin.es/wp-content/uploads/2021/03/ALGODO%CC%81N-AZUCAR-HEADER.png>
- aplaybox.com. (n.d.-c). 【崩坏:星穹铁道】砂金 [3D model]. Aplaybox. Retrieved from <https://www.aplaybox.com/details/model/Njvgh4VhWuWC>
- thingiverse.com. (n.d.). *8 foot craps table* [3D model]. Thingiverse. Retrieved from <https://www.thingiverse.com/thing:3606436>

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

- Sketchfab. (n.d.-f). *Ferris wheel* [3D model]. Sketchfab. Retrieved from <https://sketchfab.com/3d-models/ferris-wheel-675ab80b477b40f280b7311f81fee730>
- Freepik. (n.d.). *Pecat* [Vector]. Freepik. Retrieved from https://img.freepik.com/premium-vector/pecat_824631-2759.jpg
- Sketchfab. (n.d.-g). *Carousel spinning* [3D model]. Sketchfab. Retrieved from <https://sketchfab.com/3d-models/carousel-spinning-be73168bd1b44dc7b42b2c18395eaa26>
- aplaybox.com. (n.d.). [Untitled 3D model]. Aplaybox. Retrieved from <https://www.aplaybox.com/details/model/Yd2F8O36aW9E>
- aplaybox.com. (n.d.). [Untitled 3D model]. Aplaybox. Retrieved from <https://www.aplaybox.com/details/model/Bylr97MuisgW>
- Sketchfab. (n.d.). *Gun targets* [3D model]. Sketchfab. Retrieved from <https://sketchfab.com/3d-models/gun-targets-eb9fbe283faa41ef8685359a0769d303>
- aplaybox.com. (n.d.). [Untitled 3D model]. Aplaybox. Retrieved from <https://www.aplaybox.com/details/model/8INI9RGRm2hW>
- aplaybox.com. (n.d.). [Untitled 3D model]. Aplaybox. Retrieved from <https://www.aplaybox.com/details/model/DWzYm0H7Gztp>
- pngtree.com. (2022, June 16). *Warning ribbon with yellow black stripes png image* [Vector]. PNGtree. Retrieved from https://png.pngtree.com/png-vector/20220616/ourmid/pngtree-warning-ribbon-with-yellow-black-stripes-png-image_5094127.png
- Blender3D. (n.d.). *Simple Axe* [3D model]. Sketchfab. Retrieved from <https://skfb.ly/6Aqus>
- aplaybox.com. (n.d.-n). [Untitled 3D model]. Aplaybox. Retrieved from <https://www.aplaybox.com/details/model/Oh26xPqwZnJe>
- tiunov.se. (n.d.). *Bowling Club* [3D model]. Sketchfab. Retrieved from <https://skfb.ly/oPoAC>
- samixalam. (n.d.). *Bowling Alley* [3D model]. Sketchfab. Retrieved from <https://skfb.ly/psYZz>
- colorate.azurewebsites.net. (n.d.). [Color palette]. Colorate. Retrieved from <https://colorate.azurewebsites.net/SwatchColor/4B4B4B>

- Terizmeh. (n.d.). *Bowling Pin* [3D model]. Sketchfab. Retrieved from <https://skfb.ly/6puNs>
- tijmen_h. (n.d.). *Bowling ball return* [3D model]. Sketchfab. Retrieved from <https://skfb.ly/pqqxr>
- color-hex.com. (n.d.). [Color palette]. Color-Hex. Retrieved from <https://www.color-hex.com/palettes/28679.png>
- aplaybox.com. (n.d.). [Untitled 3D model]. Aplaybox. Retrieved from <https://www.aplaybox.com/details/model/PnyEtfvD7Y5A>
- gstatic.com. (n.d.). [Untitled image]. Googleusercontent. Retrieved from https://encrypted-tbn0.gstatic.com/images?q=tbn:ANd9GcQJT67Bx0xRgA7zg5HP8wFfIDaEDk0uEu7XdC9Mvx-MDHKywqClq4FSEjdP-g_NyBmntu4&usqp=CAU
- pngwing.com. (n.d.). [Light colored wood texture background image]. PNGwing. Retrieved from <https://w7.pngwing.com/pngs/148/189/png-transparent-light-colored-wood-texture-background-wood-texture-wooden-desktop-light-colored-thumbnail.png>
- aplaybox.com. (n.d.). [Untitled 3D model]. Aplaybox. Retrieved from <https://www.aplaybox.com/details/model/oEC8Vg7464Mo>
- LittleWhiteElk. (n.d.). *Cherry Soda Float* [3D model]. Sketchfab. Retrieved from <https://skfb.ly/6X9Jv>
- aplaybox.com. (n.d.). [Untitled 3D model]. Aplaybox. Retrieved from <https://www.aplaybox.com/details/model/gRVXCs6EEtCp>
- kyou.id. (n.d.). Kyou.id. Retrieved from <https://cdn.kyou.id/items/200461-nendoroid-libra-tears-of-themis.jpg.webp>
- amazon.com. (n. d.). [Image of an open book]. Amazon. Retrieved from <https://m.media-amazon.com/images/I/719ApjLMi9L.jpg>
- sebastianesp. (n.d.). *Open book* [3D model]. Sketchfab. Retrieved from <https://skfb.ly/6WP8M>
- aplaybox.com. (n.d.). [Untitled 3D model]. Aplaybox. Retrieved from <https://www.aplaybox.com/details/model/8Fmk7HdtqzkX>
- aplaybox.com. (n.d.). [Untitled 3D model]. Aplaybox. Retrieved from <https://www.aplaybox.com/details/model/RXj6UEI5ZbYk>
- hoyolab.com. (n.d.). [Untitled article]. HoYoLAB. Retrieved from <https://www.hoyolab.com/article/12096914>

- LinjieFan. (n.d.). *electro slime* [3D model]. Sketchfab. Retrieved from <https://skfb.ly/oQDZz>
- LinjieFan. (n.d.). *Water slime* [3D model]. Sketchfab. Retrieved from <https://skfb.ly/oQpHw>
- LinjieFan. (n.d.). *Fire slime* [3D model]. Sketchfab. Retrieved from <https://skfb.ly/oQ6Jt>
- fasteng. (n.d.). *Ice Crteam Scoop* [3D model]. Sketchfab. Retrieved from <https://skfb.ly/oMUOQ>
- Sketchfab. (n.d.). *Ice cream food cart* [3D model]. Sketchfab. Retrieved from <https://sketchfab.com/3d-models/ice-cream-food-cart-60b72bdcd1a445abbef6a3a2cbf7de5e>
- pngtree.com. (2023, October 11). *Vibrant ice cream inspired background texture image* [Image]. PNGtree. Retrieved from https://png.pngtree.com/thumb_back/fw800/background/20231011/pngtree-vibrant-ice-cream-inspired-background-texture-image_13610171.png
- gstatic.com. (n.d.). [Untitled image]. Googleusercontent. Retrieved from <https://encrypted-tbn0.gstatic.com/images?q=tbn:ANd9GcRitdtOu1d32A3S6h-kG3bxrrubTFdGmPldHw&:s>
- aplaybox.com. (n.d.). 【未定事件簿】莫弈/模型配布 [3D model]. Aplaybox. Retrieved from <https://www.aplaybox.com/details/model/eb0IMfxlEsxq>
- aplaybox.com. (n.d.). 【原神】「海风之梦」琴 [3D model]. Aplaybox. Retrieved from <https://www.aplaybox.com/details/model/p6wJAW4O4dar>
- Outlier Spa. (n.d.). *New York Hot Dog Cart* [3D model]. Sketchfab. Retrieved from <https://skfb.ly/osZGy>
- Yacob. (n.d.). *Hotdog* [3D model]. Sketchfab. Retrieved from <https://skfb.ly/6U7KN>
- Pablo Marquez. (n.d.). *Ketchup Bottle* [3D model]. Sketchfab. Retrieved from <https://skfb.ly/6X7DU>
- Citron Legacy. (n.d.). *Genshin Impact Mora* [3D model]. Sketchfab. Retrieved from <https://skfb.ly/o68xX>
- aplaybox.com. (n.d.). 【未定事件簿】柊暗式-左然·燃动潮流夜ver1.0 [3D model]. Aplaybox. Retrieved from <https://www.aplaybox.com/details/model/KGIDMnlhekXY>

- Sketchfab. (n.d.). *Bumper cars game attraction* [3D model]. Sketchfab. Retrieved from <https://sketchfab.com/3d-models/bumper-cars-game-attraction-446af53b3340431182d87ebbf8e48bc5>
- Amir Olphat. (n.d.). *Huawei Y560 4G Mobile Phone* [3D model]. Sketchfab. Retrieved from <https://skfb.ly/MwXW>
- aplaybox.com. (n.d.). [Untitled 3D model]. Aplaybox. Retrieved from <https://www.aplaybox.com/details/model/Zt4NADUTkjRZ>
- MaX3Dd. (n.d.). *Old Wooden Chair Low-poly* [3D model]. Sketchfab. Retrieved from <https://skfb.ly/pqJAY>
- Michael Alexis. (n.d.). *Bate* [3D model]. Sketchfab. Retrieved from <https://skfb.ly/66tJs>
- Anastacio Games. (n.d.). *Grade Grid Game* [3D model]. Sketchfab. Retrieved from <https://skfb.ly/69HqN>
- Spark Games. (n.d.). *Low Poly Balls* [3D model]. Sketchfab. Retrieved from <https://skfb.ly/p8GuZ>
- aplaybox.com. (n.d.). [Untitled 3D model]. Aplaybox. Retrieved from <https://www.aplaybox.com/details/model/zgKs3aPD7EIX>
- yuleeeee. (n.d.). *acient fan* [3D model]. Sketchfab. Retrieved from <https://skfb.ly/oxoOU>
- fulcrumgallery.com. (n.d.). *Bamboo Leaves I* [Image]. Fulcrum Gallery. Retrieved from <https://www.fulcrumgallery.com/product-images/P956317-10/bamboo-leaves-i.jpg>
- Berk Gedik. (n.d.). *Abandoned Toilet Cabin (Low Poly)* [3D model]. Sketchfab. Retrieved from <https://skfb.ly/6Vylo>
- aplaybox.com. (n.d.). 【未定事件簿】莫弈-飞雪 [3D model]. Aplaybox. Retrieved from <https://www.aplaybox.com/details/model/J3Z8Qtv792V1>
- tasha.kosaykina. (n.d.). *3December2020 | Plant orchid flower* [3D model]. Sketchfab. Retrieved from <https://skfb.ly/6Z8tB>
- Voyage. (n.d.). *Simple stylized Cherry blossom* [3D model]. Sketchfab. Retrieved from <https://skfb.ly/oGrxx>
- Voyage. (n.d.). *Cherry blossom petal* [3D model]. Sketchfab. Retrieved from <https://skfb.ly/oGrxy>

- aplaybox.com. (n.d.). [Untitled 3D model]. Aplaybox. Retrieved from <https://www.aplaybox.com/details/model/U7tpl2b7hNei>
- aplaybox.com. (n.d.). [Untitled 3D model]. Aplaybox. Retrieved from <https://www.aplaybox.com/details/model/sKEfNYfXwqFn>
- Keyotine. (n.d.). *Stylized Carnival Booth* [3D model]. Sketchfab. Retrieved from <https://skfb.ly/olyrq>
- ftcn.net. (n.d.). [Carnival background image]. Adobe Stock. Retrieved from https://t3.ftcdn.net/jpg/07/30/73/34/360_F_730733455_DcJ0aOtRxn5xunWoe1Q4AxacEN7HqTRD.jpg
- plaggy. (n.d.). *CC0 - Balloon* [3D model]. Sketchfab. Retrieved from <https://skfb.ly/oUTFo>
- itsdevinci. (n.d.). *Low Poly Dart* [3D model]. Sketchfab. Retrieved from <https://skfb.ly/oHtAD>
- aplaybox.com. (n.d.). [Untitled 3D model]. Aplaybox. Retrieved from <https://www.aplaybox.com/details/model/bnX7ynEPaP1C>
- aplaybox.com. (n.d.). [Untitled 3D model]. Aplaybox. Retrieved from <https://www.aplaybox.com/details/model/F58n5j7y68EQ>
- Elbolillo. (n.d.). *Tacos_ Props* [3D model]. Sketchfab. Retrieved from <https://skfb.ly/oYMJH>
- AVIKA. (n.d.). *Avika Street Food Cart* [3D model]. Sketchfab. Retrieved from <https://skfb.ly/oBTu8>
- swedde. (n.d.). *Plastic Table* [3D model]. Sketchfab. Retrieved from <https://skfb.ly/ossMV>
- chilango.com. (n.d.). Chilango. Retrieved from 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. (n.d.). *Plain neon green solid color* [Image]. Shutterstock. Retrieved from [\[https://www.shutterstock.com/image-illustration/plain-neon-green-solid-color-260nw-1667301652.jpg\]](https://www.shutterstock.com/image-illustration/plain-neon-green-solid-color-260nw-1667301652.jpg)(<https://www.shutterstock.com/image-illustration/plain-neon-green-solid-color-260nw-1667301652.jpg>)
- Nathalie Michel. (n.d.). *Big Knife* [3D model]. Sketchfab. Retrieved from <https://skfb.ly/6sURV>

- aplaybox.com. (n.d.). [Untitled 3D model]. Aplaybox. Retrieved from <https://www.aplaybox.com/details/model/7Ox4GS5NB2MF>
- aplaybox.com. (n.d.). [Untitled 3D model]. Aplaybox. Retrieved from <https://www.aplaybox.com/details/model/ox8L165kTbDq>
- pinterest.com. (n.d.). [Untitled image]. Pinterest. Retrieved from <https://i.pinimg.com/236x/d9/13/48/d91348a831a52b128c5b5aacaca6ee10.jpg>
- istockphoto.com. (n.d.). *Circo carnaval retro rayas vintage con estrellas patrón sin costuras plantilla gráfica* [Illustration]. iStock. Retrieved from 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=nXI_aY9Ad-hgjyxr7KxCeojMB26gpJA-QugSXEnUg8=
- aplaybox.com. (n.d.). [Untitled 3D model]. Aplaybox. Retrieved from <https://www.aplaybox.com/details/model/0ygLuNRGnO1z>
- pngwing.com. (n.d.). [Untitled image]. PNGwing. Retrieved from <https://www.pngwing.com/es/free-png-zyqxw/download>
- pinterest.com. (n.d.). [Untitled image]. Pinterest. Retrieved from <https://i.pinimg.com/736x/5a/74/41/5a7441c392720453d416ce531030d173.jpg>
- Gamedirection. (n.d.). *Park Table - Low Poly* [3D model]. Sketchfab. Retrieved from <https://skfb.ly/6CTuW>
- aplaybox.com. (n.d.). [Untitled 3D model]. Aplaybox. Retrieved from <https://www.aplaybox.com/details/model/qTXzeDkYaiBu>
- aplaybox.com. (n.d.). [Untitled 3D model]. Aplaybox. Retrieved from <https://www.aplaybox.com/details/model/hzLyyqDYqYNp>
- Sketchfab. (n.d.). *Honkai: Star Rail - Ruan Mei Creations Cat Cake* [3D model]. Sketchfab. Retrieved from <https://sketchfab.com/3d-models/honkai-star-rail-ruan-mei-creations-cat-cake-f700f314430043c99bc700185f3ab0a2>
- aplaybox.com. (n.d.). [Untitled 3D model]. Aplaybox. Retrieved from <https://www.aplaybox.com/details/model/e0kNzNem9ZgQ>
- aplaybox.com. (n.d.). [Untitled 3D model]. Aplaybox. Retrieved from <https://www.aplaybox.com/details/model/txXgqWpAVzpr>

- aplaybox.com. (n.d.). [Untitled 3D model]. Aplaybox. Retrieved from <https://www.aplaybox.com/details/model/YquJuZCT6e5w>
- aplaybox.com. (n.d.). [Untitled 3D model]. Aplaybox. Retrieved from <https://www.aplaybox.com/details/model/EZXu8LRkeXY6>
- aplaybox.com. (n.d.). [Untitled 3D model]. Aplaybox. Retrieved from <https://www.aplaybox.com/details/model/j4kat4jyR9bV>
- aplaybox.com. (n.d.). [Untitled 3D model]. Aplaybox. Retrieved from <https://www.aplaybox.com/details/model/ry7ZgsrgLE4Z>
- aplaybox.com. (n.d.). [Untitled 3D model]. Aplaybox. Retrieved from <https://www.aplaybox.com/details/model/nxFIrs3fVI3Q>
- aplaybox.com. (n.d.). [Untitled 3D model]. Aplaybox. Retrieved from <https://www.aplaybox.com/details/model/1zHFnlSmjS2V>
- aplaybox.com. (n.d.). [Untitled 3D model]. Aplaybox. Retrieved from <https://www.aplaybox.com/details/model/gByJebJf5M2r>
- aplaybox.com. (n.d.). [Untitled 3D model]. Aplaybox. Retrieved from <https://www.aplaybox.com/details/model/391cKeeMyed8>
- Gamedirection. (n.d.). *Park Table - Low Poly* [3D model]. Sketchfab. Retrieved from <https://skfb.ly/6CTuW>
- aplaybox.com. (n.d.). [Untitled 3D model]. Aplaybox. Retrieved from <https://www.aplaybox.com/details/model/zAalgNTApJ0a>
- hoyolab.com. (n.d.). [Untitled article]. HoYoLAB. Retrieved from <https://www.hoyolab.com/article/12096914>
- tot.wiki. (n.d.). Tears of Themis Wiki. Retrieved from https://tot.wiki/thumb.php?f=2024_Halloween_Luke.jpg&width=900
- tot.wiki. (n.d.). Tears of Themis Wiki. Retrieved from https://tot.wiki/thumb.php?f=2024_Halloween_Vyn.jpg&width=900
- tot.wiki. (n.d.). Tears of Themis Wiki. Retrieved from https://tot.wiki/thumb.php?f=2024_Halloween_Artem.jpg&width=900
- tot.wiki. (n.d.). Tears of Themis Wiki. Retrieved from https://tot.wiki/thumb.php?f=2024_Halloween_Marius.jpg&width=900
- hoyolab.com. (2024, October 31). [Untitled image]. HoYoLAB. Retrieved from 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, February 12). [Untitled image]. HoYoLAB. Retrieved from 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%2Fquality%2Cq_70
- hoyolab.com. (n.d.). [Untitled article]. HoYoLAB. Retrieved from <https://www.hoyolab.com/article/12096914>
- tot.wiki. (n.d.). *Whac-A-Mole*. Tears of Themis Wiki. Retrieved from <https://tot.wiki/wiki/Whac-A-Mole>