

Entrega 3 Proyecto

Created by Andrea

Entregas pasadas y Sustentación

Resumen Entregas Anteriores

El proyecto en su etapa inicial se enfocó en el desarrollo de un oso de goma interactivo en realidad aumentada, donde el principal objetivo fue modelar el oso con textura de goma en Blender e integrarlo al ejemplo GLTF de Diligent. Durante esta fase se enfrentaron desafíos técnicos relacionados con la pérdida de textura y deformaciones del modelo, lo que llevó a intentar modificar el código del Loader para mejorar la carga de texturas.

En la segunda entrega, se lograron avances significativos al corregir los problemas de geometría y textura del oso, permitiendo una visualización completa y correcta del modelo en Diligent. Además, se implementó exitosamente una animación de caminata mediante Rigging, que incluye movimientos naturales de brazos, hombros, cadera, piernas y orejas. También se realizaron modificaciones en el código para optimizar la carga del modelo y se iniciaron investigaciones sobre la implementación de realidad aumentada, estableciendo contactos para acceder a gafas AR y explorando la instalación de ARCore.

Cambios Hechos a la idea principal

La idea principal se desarrollaba mediante una APK de ARCore y generar una aplicación móvil que permitiera la visualización exitosa de el oso en realidad aumentada. Debido a que aun no he visto la materia de desarrollo de aplicaciones móviles y que realizar una aplicación de este estilo en menos de 2 semanas era un trabajo pesado basada en la recomendación del profesor se decidió realizar un cambio a la idea principal haciendo un ejercicio basado en el ejemplo 20 de Ray Tracing. En este se aplicaría una mesa generada en Blender sobre la cual se

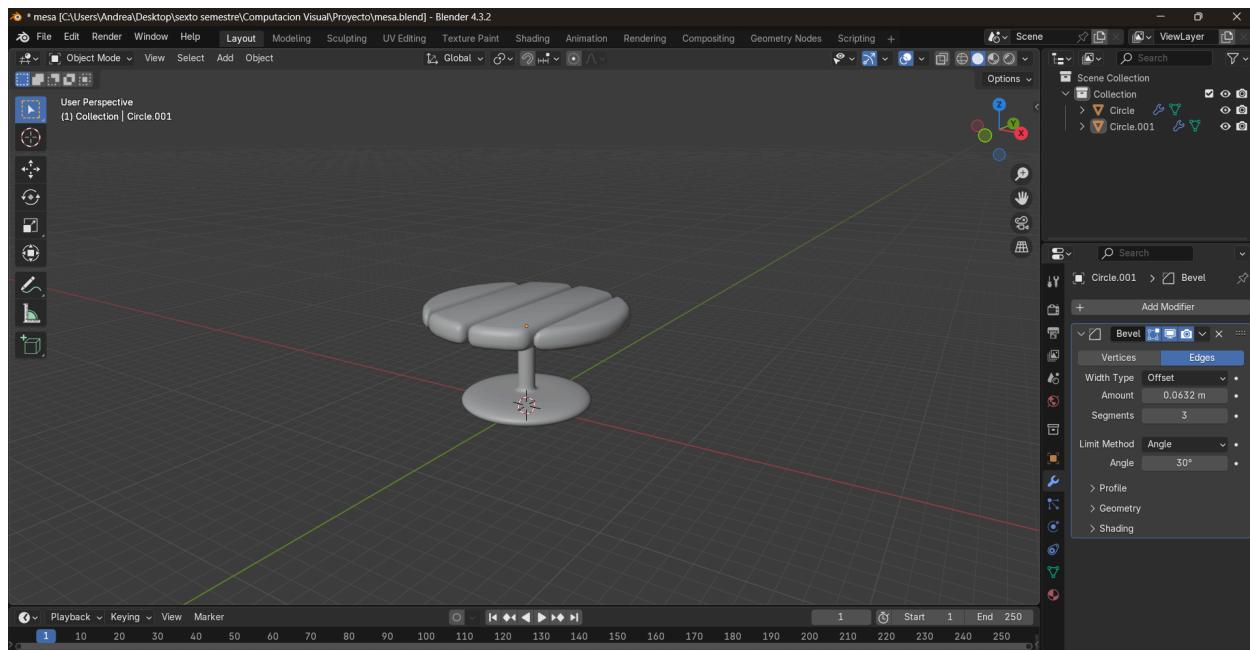
encontrarían los osos de goma y varias cajas. En este cambio de planes la idea central sería mostrar varios osos de goma caminando y que se acerquen a la caja más cercana y al tener contacto con dicha caja que el oso desaparezca al "entrar a la caja".

Avance entrega 3

Para esta entrega se realizaron avances importantes para el desarrollo de la nueva idea central. en este avance se logró generar el modelo de la mesa Blender y la creación de la textura de la caja en Diligent Engine y se realizaron las primeras pruebas de integración con el ejemplo de GLTFViewer.

Mesa Blender

Se generó el modelo de la mesa deseada para poner los osos y las cajas de dulces sobre esta para esto se usó la herramienta de Blender, debido a que es una mesa estática no requirió de animación.



Luego de la finalización del modelado de la mesa, se verificó el correcto procesamiento del gltf dentro de Diligent Engine al cargarla en el Diligent se pudo

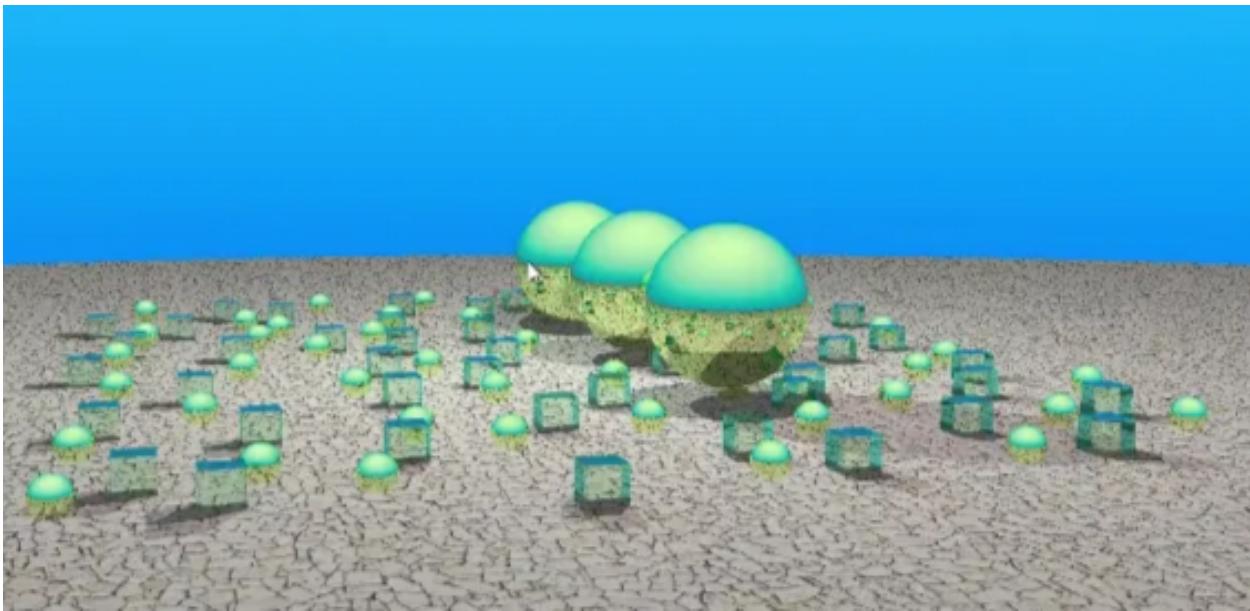
ver el correcto funcionamiento de todos los detalles de la mesa.



Cajas Diligent

descripción general

Para la generacion de las cajas se utilizo la tarea 11, para poder adaptar el ejemplo para que funcionara para el proyecto, se tuvo que hacer varios cambios importantes debido a que el ejemplo hecho para la tarea aun no tenia el funcionamiento el slider para el aumento y la disminución de las entidades. Antes del inicio de la edición el ejemplo se veia de la siguiente manera:



ejemplo tarea 10 antes de edición

Cambios aumento de Entidades

Inicialmente se edito el funcionamiento del ejemplo original y se logro hacer que el slider funcionara correctamente aumentando y disminuyendo las entidades generadas.

en esta parte del código se edita la cantidad de esferas vistas dentro de la vista original

```
for (int i = 0; i < NumSmallSpheres; ++i)
{
    // build the exact same names you used in UpdateTLAS()
    char instanceName[32];
    std::snprintf(instanceName, sizeof(instanceName), "SmallSphere%d", i);

    m_pSBT->BindHitGroupForInstance(
        m_pTLAS,
        instanceName,
        PRIMARY_RAY_INDEX,
        "SpherePrimaryHit");
}
```

Cambios Entidades Mostradas

Debido a que las entidades necesarias para el proyecto solamente son cuadrados/rectangulos se tuvo que hacer cambios en el ejemplo debido a que se tienen mayormente esferas por lo que se cambiaron algunas partes del código ademas de la mostrada en el fragmento anterior se cambiaron las siguientes partes del código

```
static std::string smallSphereNames[MaxSmallSpheres];
int      gridSize = static_cast<int>(std::ceil(std::sqrt(MaxSmallSpheres)));
float   cellSize = 8.0 ; // Puedes ajustar este valor según el espacio que

for (int i = 0; i < MaxSmallSpheres; ++i)
{
    int idx = 4 + i;
    auto& inst = Instances[idx];

    smallSphereNames[i] = "SmallSphere" + std::to_string(i);
    inst.InstanceName = smallSphereNames[i].c_str();

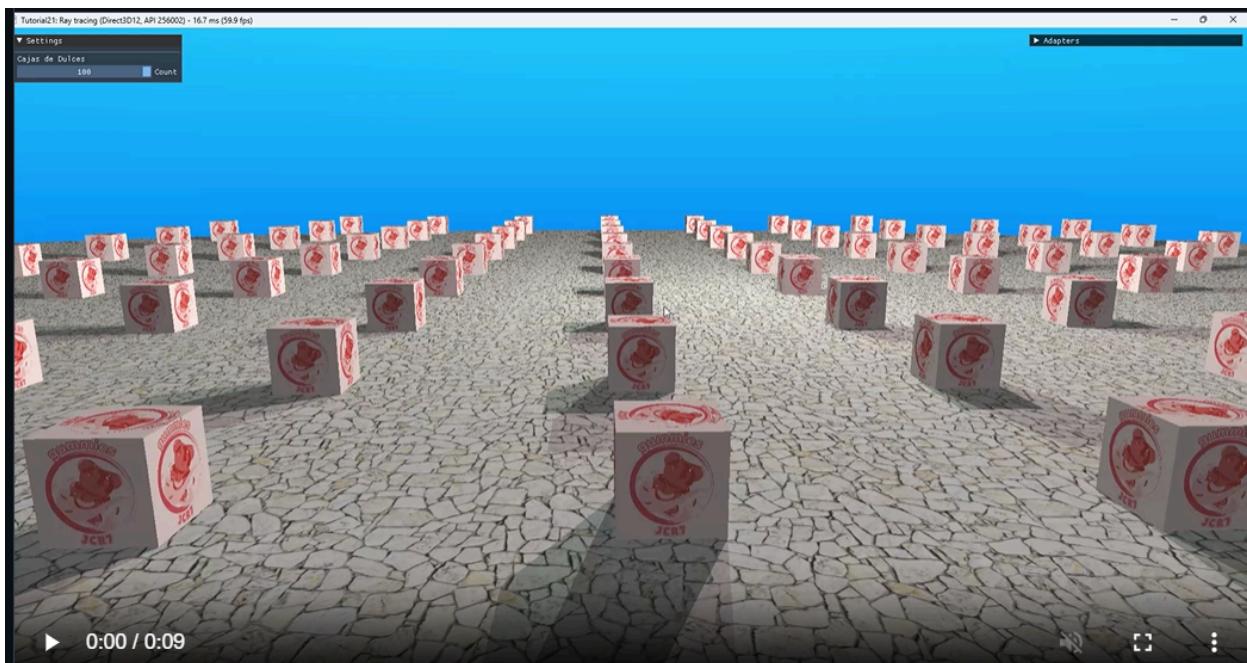
    inst.pBLAS = m_pCubeBLAS;
    inst.Mask = (i < m_NumScatterSpheres ? OPAQUE_GEOM_MASK : 0u);
    inst.CustomId = 3;

    // Cálculo de posición en cuadrícula
    int row = i / gridSize;
    int col = i % gridSize;

    float x = (col - gridSize / 2.0f) * cellSize;
    float z = (row - gridSize / 2.0f) * cellSize;
```

```
    inst.Transform.SetTranslation(x, -4.89f, z);  
}
```

en esta parte del código se realizo un cambio de entidad para que fueran cubos y además se cambiaron las coordenadas para que se ordenaran de manera distinta, homogénea, además de una cuadricula más amplia para permitir que los osos caminen más libremente al rededor de la mesa.



Textura

se genero una textura diferente dentro de Diligent para simular una caja de carton de gomitas similar al paquete de dulces Oka Loka nano los cuales son una cajita de carton pequeña de diferentes colores. Para los colores se espera que para la siguiente implementación se tengan cajas de varios colores.



Inspiracion Caja



Diseño de caja

La textura es ve un poco opaca en el Diligent debido a que se espera que sea similar al cartón.

se utilizo esta textura

```
#include "structures.fch"
#include "RayUtils.fch"

ConstantBuffer<CubeAttribs> g_CubeAttribsCB;

Texture2D g_CubeTextures[NUM_TEXTURES];
SamplerState g_SamLinearWrap;

[shader("closesthit")]
void main(inout PrimaryRayPayload payload, in BuiltInTriangleIntersectionAttribute
{
    // Calculate triangle barycentrics.
    float3 barycentrics = float3(1.0 - attr.barycentrics.x - attr.barycentrics.y, attr.barycentrics.z);

    // Get vertex indices for primitive.
    uint3 primitive = g_CubeAttribsCB.Primitives[PrimitiveIndex()].xyz;

    // Calculate texture coordinates.
    float2 uv = g_CubeAttribsCB.UVs[primitive.x].xy * barycentrics.x +
```

```

g_CubeAttribsCB.UVs[primitive.y].xy * barycentrics.y +
g_CubeAttribsCB.UVs[primitive.z].xy * barycentrics.z;

// Calculate and transform triangle normal.
float3 normal = g_CubeAttribsCB.Normals[primitive.x].xyz * barycentrics.x +
    g_CubeAttribsCB.Normals[primitive.y].xyz * barycentrics.y +
    g_CubeAttribsCB.Normals[primitive.z].xyz * barycentrics.z;
normal      = normalize(mul((float3×3) ObjectToWorld3×4(), normal));

// Sample texturing. Ray tracing shaders don't support LOD calculation, so we
payload.Color = g_CubeTextures[NonUniformResourceIndex(InstanceID())].Sample();
payload.Depth = RayTCurrent();

// Apply lighting.
float3 rayOrigin = WorldRayOrigin() + WorldRayDirection() * RayTCurrent();
LightingPass(payload.Color, rayOrigin, normal, payload.Recursion + 1);
}

```

Integracion GLTFViewer y Tutorial Diligent 21

para realizar la integración con GLTFViewer se trataron de integrar de 3 formas distintas

Realizar la función de cpp donde se carga el modelo dentro del GLTFViewer en el hpp y el cpp del Tutorial

al realizar esta integración se generaron bastantes problemas en los #include, y al arreglar los #includes generaba distintos errores haciendo referencia a que no funcionaban cosas dentro del GLTF que si funcionaban en el ejemplo del GLTF original, empezaron a salir errores que no aparecían a pesar de estar igual y tener los modelos y los assets en los sitios donde debían.

Importar funciones incluyendo el hpp y cpp del GLTFViewer

al tratar de hacer eso genero el siguiente error, se intento de diferentes maneras y genero el mismo error

The screenshot shows a code editor with the following code snippet:

```
>void Tutorial21_RayTracing::WindowResize(UINT32 Width, UINT32 Height) { ... }

void Tutorial21_RayTracing::LoadModel()
{
    const char* camino = "assets/oso/gltf/OSITO.gltf";

    GLTFViewer::LoadModel(camino);
}

>void Tutor
} // names
```

A tooltip is displayed over the line `GLTFViewer::LoadModel(camino);`, containing the following information:

- class Diligent::GLTFViewer
- Tamaño: 960 Bytes
- Alineación: 8 Bytes
- Buscar en línea | Diseño de memoria

Below the tooltip, a message states: "una referencia de miembro no estático debe ser relativa a un objeto específico".

cuando por fin funciono y se elimino el error aparecían errores dentro del gltf viewer a pesar de solo estar en el #include y no realizar ningún cambio dentro de este solamente habían cambios en el tutorial y no en el gltf

Por otro lado se trato de lado contrario importar las funciones de hpp y cpp del Tutorial 21 al GLTFViewer

No se pudo terminar de realizar este intento debido a la generación de varios errores al momento de llamar las funciones