Proyectos de Desarrollo de Aplicaciones e Innovación con Alumnos – 2024

Título del proyecto: Aplicaciones móviles 3D, Realidad Virtual y Realidad Aumentada 2024

Coordinadores: Lic. Sebastián Dapoto, Lic. Federico Cristina.

Alumnos

Nro. de Legajo	Apellido y Nombre
03398/7	Restucha, Tiago
03229/0	Schwindt, Ignacio Andres
01221/0	Barbero, Lautaro

Resumen

Se desarrolló un prototipo en Realidad Virtual para Android utilizando como entorno de Desarrollo Unity + Google CardBoard destinado a dar una recorrida virtual de la facultad de Informática a los futuros ingresantes de la misma, así como la visualización de los horarios de los cursos dados en el momento

Objetivo

El objetivo del proyecto es explorar tecnologías y técnicas de Realidad Virtual (Virtual Reality o VR), aplicadas al diseño y desarrollo de aplicaciones móviles. Tras realizar una investigación sobre el estado del arte, se analizan distintas herramientas y frameworks que permiten utilizar este tipo de tecnologías, con un enfoque principal en la plataforma de desarrollo de videojuegos Unity.

Los conocimientos adquiridos serán aplicados al desarrollo de un prototipo móvil en 3D con VR, destinado a ofrecer un recorrido virtual por la Facultad de Informática. Durante el recorrido, los usuarios podrán explorar las distintas aulas, donde se les mostrará la materia que se dicta en cada una, junto con los horarios correspondientes. Este prototipo servirá como un primer paso para el desarrollo de una aplicación futura con mayores funcionalidades, que permitirá a los estudiantes y visitantes conocer mejor la distribución y actividades de la facultad.

Tareas Realizadas

Las tareas realizadas pueden subdividirse en 3 grandes grupos

A - Modificación del modelo 3D de la Facultad realizado anteriormente

B - Creación del sistema de Colisiones-Movimiento para el entorno de Google Cardboard

C - Agregar al proyecto el manejo de la API de las Aulas/Horarios

A - Creación-Modificación del Modelo 3D de la Facultad

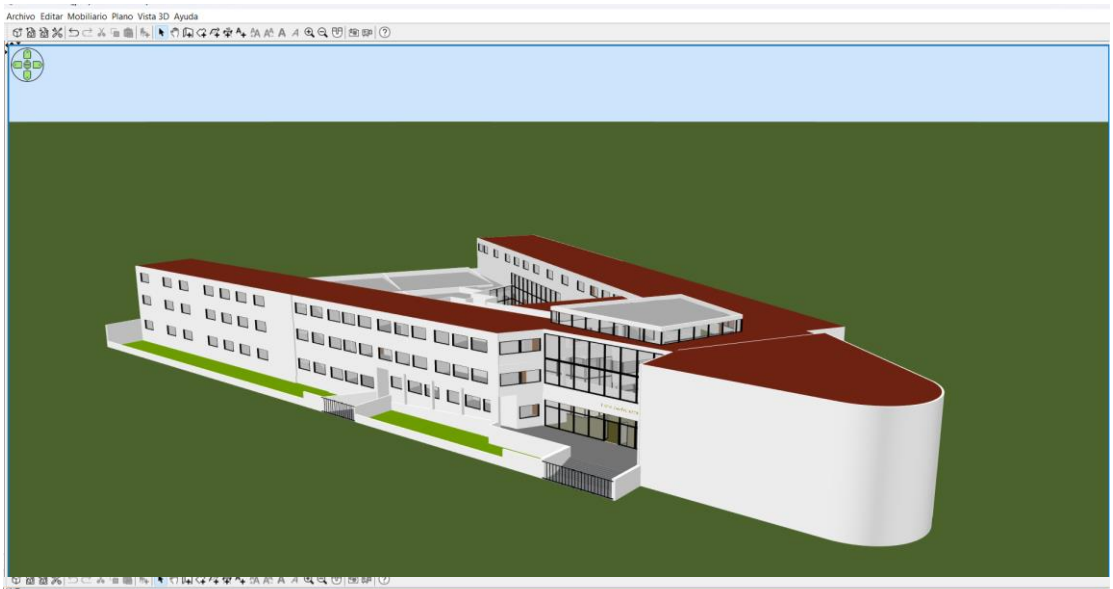
En un principio, se recibe un modelo tridimensional de la Facultad realizado previamente en el programa “Sweet Home 3D” para otra edición de la Expo Ciencia y Tecnología, junto con los planos de la Institución. A partir de esta base, se evalúan las partes faltantes en el proyecto, así como las modificaciones necesarias en aquellas áreas que no representan fielmente los planos originales.

Una vez realizada la evaluación, se enumeran y documentan las partes o características necesarias para completar el modelo, para luego proceder con el siguiente paso.

El siguiente paso consiste en la modificación y mejora del diseño a través del software de modelado 3D previamente mencionado. Este proceso incluyó las siguientes modificaciones principales:

- **Creación de las aulas 14 y 15**, así como del anfiteatro situado sobre estas, y la incorporación de las condiciones de acceso correspondientes.
- **Modificación del área accesible por el personaje**, basada en las limitaciones físicas de las instalaciones que fueron agregadas después de la primera versión del archivo.
- **Correcciones varias de las geometrías**, incluyendo intersecciones de paredes y suelos mal colocados.

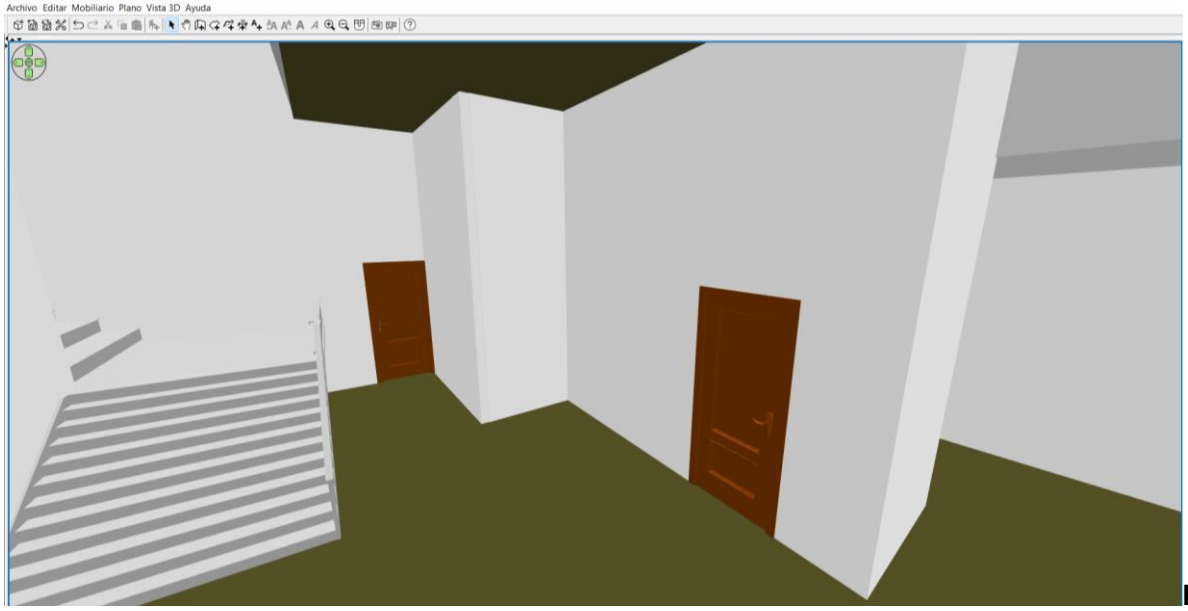
Una vez que las partes faltantes se han añadido y el modelo ha sido optimizado, se realiza una validación del diseño completo para asegurar que cumple con las expectativas del proyecto. Tras la revisión final, se exporta el modelo 3D finalizado en el formato adecuado para su edición en Unity (.OBJ).



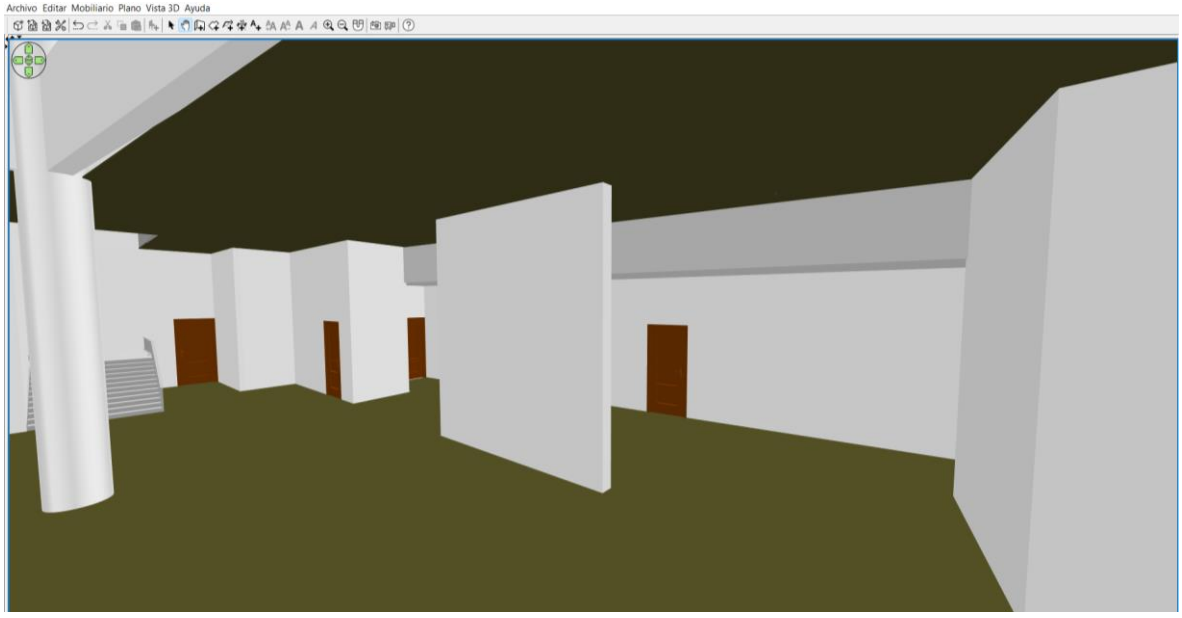
Vista aérea

Anfiteatro

2° Piso



Planta Baja



Planta Baja

Pasillo

B - Creación del sistema de Colisiones-Movimiento

Una vez recibido el modelo actualizado de la facultad, el primer paso es la creación del “personaje principal”, representado por un modelo 3D en forma de cilindro. Sobre este objeto se posicionará la cámara principal y se aplicarán las modificaciones necesarias para implementar el movimiento y las colisiones con el entorno.

En cuanto al movimiento, se desarrolló un script que se integra como componente del “personaje principal”. Este script permite que el jugador avance al inclinar ligeramente la cabeza hacia abajo (aproximadamente 15°) y se detenga al volver a levantarla. Además, el script considera variables que garantizan un desplazamiento adecuado al subir y bajar escaleras.

El código del script es el siguiente:

```
using UnityEngine;

[RequireComponent(typeof(Rigidbody))]
[RequireComponent(typeof(CapsuleCollider))]
public class HeadTiltMovement : MonoBehaviour
{
    public float speed = 3.0f;              // Velocidad de movimiento
    public float tiltThreshold = 15.0f;     // Umbral de inclinación hacia abajo
    public float rayDistance = 1.5f;        // Distancia del SphereCast para detectar el suelo
    public float extraGravity = 50.0f;      // Gravedad extra cuando no está tocando el suelo
    public LayerMask groundLayer;           // Capa que representa el suelo
    public float groundCheckRadius = 0.3f;  // Radio del SphereCast para la detección del suelo
    public float maxVerticalSpeed = 5.0f;   // Máxima velocidad vertical permitida
    public float stepOffset = 0.5f;         // Ajustar para que el personaje suba pequeños escalones
    public float forwardDrag = 1.0f;        // Drag para controlar el movimiento hacia adelante
    public float fallAcceleration = 200.0f; // Fuerza adicional para asegurar la caída inmediata

    private Rigidbody rb;
    private bool isGrounded;

    void Start()
    {
        // Obtener el componente Rigidbody
        rb = GetComponent<Rigidbody>();

        // Evitar rotación debido a la física
        rb.freezeRotation = true;

        // Desactivar la gravedad por defecto para controlarla manualmente
        rb.useGravity = false;

        // Configurar el modo de detección de colisiones
        rb.collisionDetectionMode = CollisionDetectionMode.Continuous;

        // Asegurar que el drag sea cero
        rb.drag = 0;
        rb.angularDrag = 0;
    }

    void FixedUpdate()
    {
        // Verificar si el personaje está tocando el suelo
        CheckGroundStatus();

        // Obtener la rotación de la cámara principal
        Quaternion cameraRotation = Camera.main.transform.rotation;
```

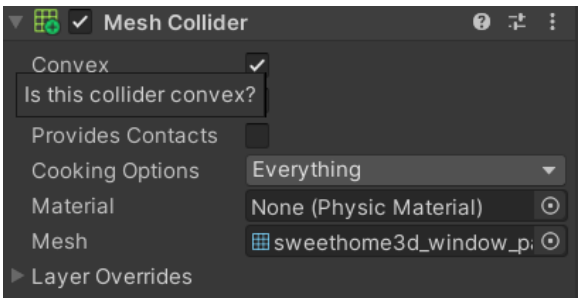
```
        Vector3 eulerAngles = cameraRotation.eulerAngles;

        // Ajustar el ángulo X al rango -180 a 180
        eulerAngles.x = (eulerAngles.x > 180) ? eulerAngles.x - 360 : eulerAngles.x;

        // Controlar el movimiento basado en la inclinación
        if (eulerAngles.x > tiltThreshold)
        {
            // Calcular la dirección hacia adelante en el plano horizontal
            Vector3 forward = new Vector3(Camera.main.transform.forward.x, 0,
Camera.main.transform.forward.z).normalized;

            // Calcular el movimiento hacia adelante
            Vector3 forwardMovement = forward * speed;

            // Establecer la nueva velocidad en X y Z (mantener componente Y)
            rb.velocity = new Vector3(forwardMovement.x, rb.velocity.y, forwardMovement.z);

            // Aplicar un pequeño arrastre en el movimiento hacia adelante para suavizarlo
            rb.velocity = Vector3.Lerp(rb.velocity, new Vector3(forwardMovement.x, rb.velocity.y,
forwardMovement.z), forwardDrag * Time.deltaTime);
        }
        else
        {
            // Detener el movimiento en X y Z cuando no hay inclinación suficiente
            rb.velocity = new Vector3(0, rb.velocity.y, 0);
        }

        // Aplicar caída instantánea si no está en el suelo
        if (!isGrounded)
        {
            rb.AddForce(Vector3.down * fallAcceleration, ForceMode.Acceleration);
        }
    }

    void CheckGroundStatus()
    {
        RaycastHit hit;

        // Usar un SphereCast para detectar el suelo en lugar de rayos individuales
        if (Physics.SphereCast(transform.position, groundCheckRadius, Vector3.down, out hit, rayDistance,
groundLayer))
        {
            isGrounded = true;
            rb.velocity = new Vector3(rb.velocity.x, Mathf.Clamp(rb.velocity.y, -maxVerticalSpeed,
float.MaxValue), rb.velocity.z);

            // Ajustar la posición vertical ligeramente si está cerca de un escalón
            if (hit.distance < stepOffset)
            {
                Vector3 position = rb.position;
                position.y = hit.point.y + stepOffset;
                rb.position = position;
            }
        }
        else
        {
            isGrounded = false;
        }
    }
}
```

Una vez completada la parte del script, se procede a configurar los colisionadores. En primer lugar, se asigna un componente llamado "Capsule Collider" al personaje principal. Posteriormente, dado que el modelo proporcionado por la facultad está compuesto por elementos individuales, se recorre cada uno de estos elementos y se les añade el componente "Mesh Collider".

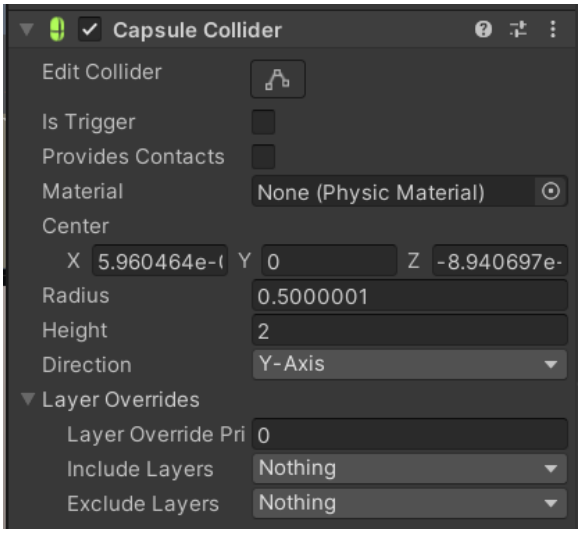
Esta configuración asegura que el personaje colisione correctamente con el entorno, evitando que atraviese paredes o puertas, y permitiendo subir y bajar escaleras de manera adecuada. En las áreas inaccesibles del modelo no se añadieron colisionadores, ya que no eran necesarios para la interacción del personaje.

A continuación se muestran imágenes representativas de lo antes explicado:

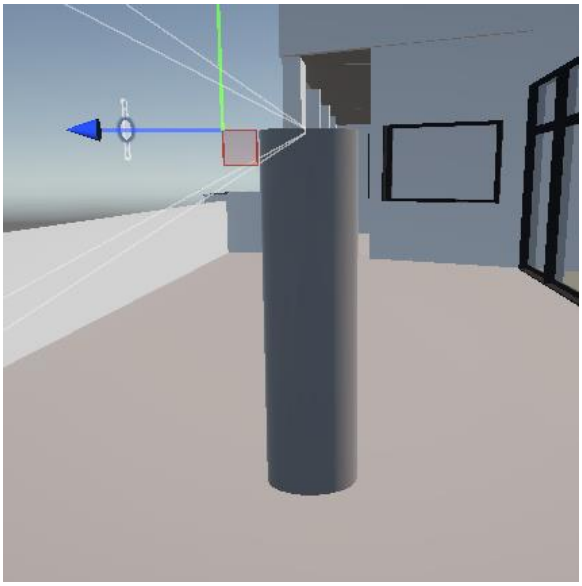
- **Mesh Collider**:



- **Capsule Collider:**



- **Cilindro “Personaje Principal”:**



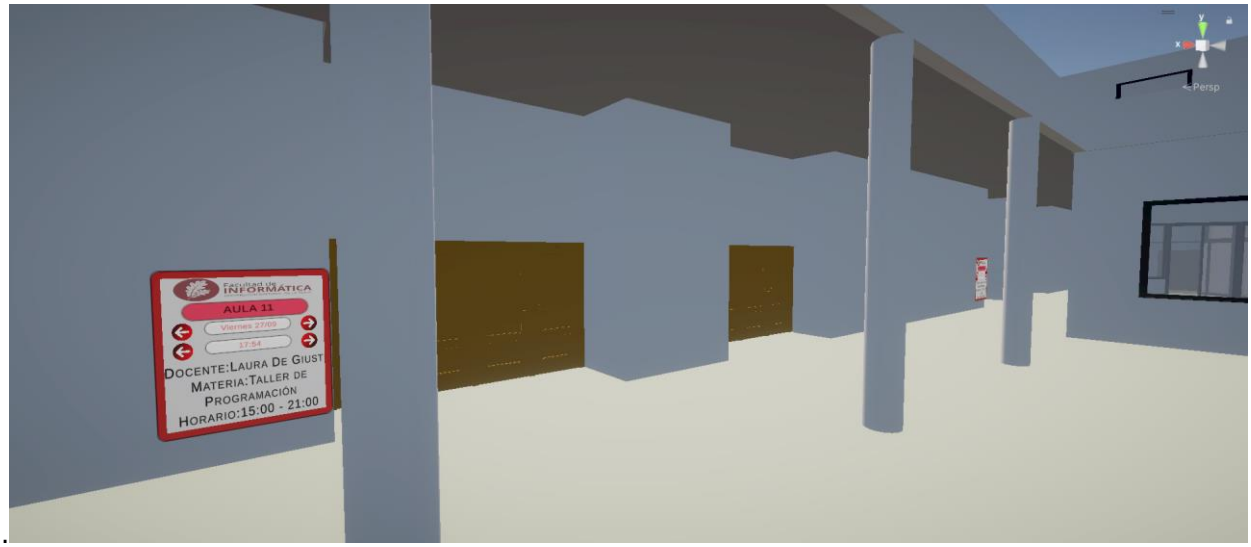
C - Agregar al proyecto el manejo de la API de las Aulas/Horarios

Para mejorar la funcionalidad del proyecto y ofrecer una experiencia más completa, se integrarán carteles en el Modelo de Unity de las aulas, en los cuales se mostrarán los correspondientes horarios e información de las materias que se cursan en el instante solicitado.

La integración de la API implica los siguientes pasos:

- A partir de la información para el manejo de la Api provista por la Institución, se establece una conexión con la API que provee los datos de las aulas y los horarios.
- Los datos obtenidos se procesan para integrarlos al proyecto. Esto incluye la interpretación de la información sobre el aula y la información relevante proporcionada por la API. Estos datos quedarán almacenados en una lista, en donde el nombre del aula será utilizado por un Enum para acceder al índice del vector de carteles del cartel correspondiente (Los cuales son asignados a mano en su ubicación correspondiente)
- Una vez procesados, los datos se incorporan al Prefab “Cartel” realizado específicamente para mostrar la información requerida por la Facultad.
- Para mantener la información actualizada, se implementó un sistema de actualización dinámica que sincroniza regularmente los datos de la API cada 30 minutos con el modelo. Esto garantiza que cualquier cambio en los horarios o la disponibilidad de las aulas se refleje inmediatamente en el proyecto.

Esta integración no solo mejora la funcionalidad del modelo 3D, sino que también ofrece a los usuarios una herramienta interactiva que muestra de manera realista la gestión de aulas y horarios en la Facultad

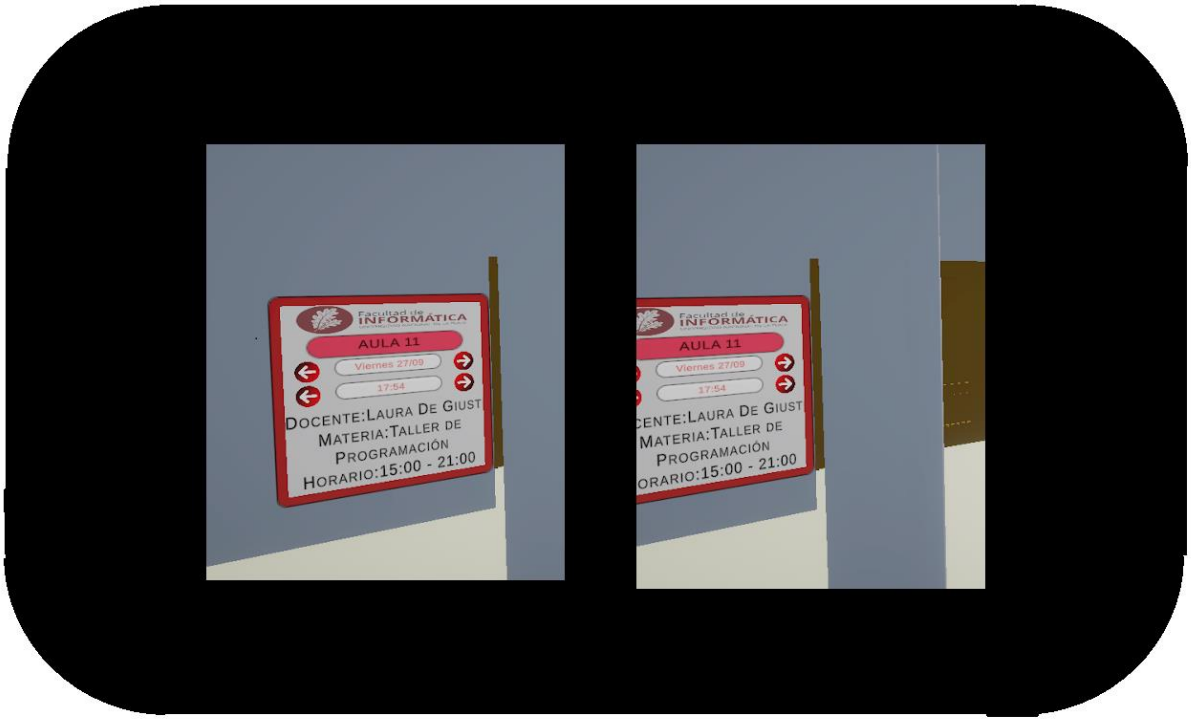


.

<u>Prototipo/Desarrollo a presentar en la 10ma Expo Ciencia y Tecnología el 16/10/2024 (Resultados)</u>

El prototipo desarrollado junta todos estos aspectos desarrollados anteriormente, para realizar un recorrido virtual a la facultad de Informática

A continuación se muestra una imagen a modo representativo del resultado final



Conclusiones

En este trabajo, se logró mejorar el modelo provisto, agregando y corrigiendo elementos clave para alcanzar una representación fiel de los planos y espacios reales que los futuros ingresantes utilizaran en su estadía en la Institución. La implementación de la API permitio añadir una capa de interactividad al proyecto, permitiendo que los usuarios obtengan información actualizada sobre la información de las aulas en tiempo real.

Este proyecto ha sido muy útil para adquirir una base de las capacidades profesionales en los campos de Diseño 3D, desarrollo de videojuegos y gestión de información a través de servidores, que ayuda a formar profesionales que tengan una base amplia de conocimientos en estos campos, junto con el aprendizaje de exponer nuestras visiones a posibles interesados

Además, siendo lo más importante, se logró mostrar una posible aplicación de los contenidos vistos en las cátedras (Manejo de Bases de datos, Programación Orientada a Objetos y Diseño 3D) a los futuros ingresantes, mostrandoles una aplicación práctica de lo que la carrera es capaz de ofrecer

Bibliografía

Documentación oficial de Unity XR:

www.docs.unity3d.com/Manual/XR.html

www.docs.unity3d.com/ScriptReference/XR.XRDevice.html

Configuración de Google Cardboard en Unity:

www.developers.google.com/cardboard/develop/unity/quickstart

Configuración de movimiento en Unity VR:

www.assetstore.unity.com/packages/tools/integration/steamvr-plugin-32647

www.developer.oculus.com/documentation/unity/unity-intro

Varios

UNITY Tutorial en Español para Principiantes

Realidad Virtual Google Cardboard Tutorial Unity - Configuración

Aprende Unity desde cero | Curso Completo (Español)

Unity VR XR Interaction Toolkit - UI Menu and Interactions