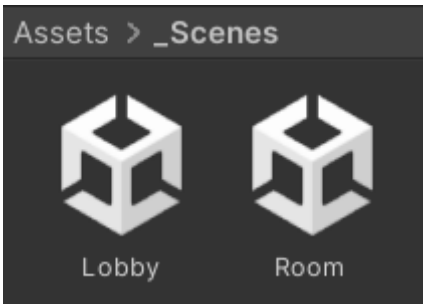
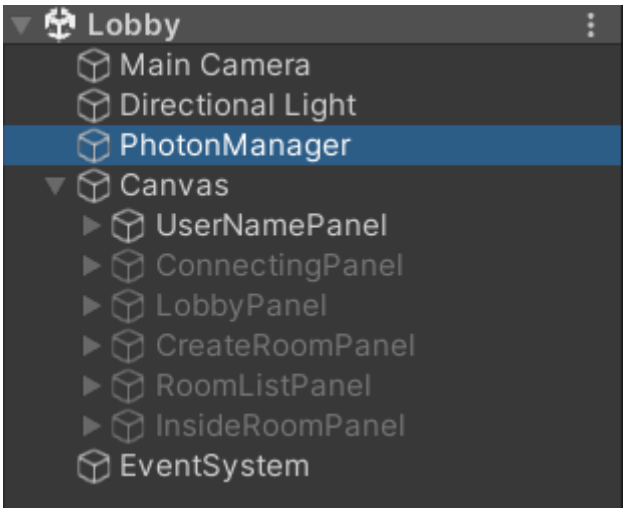
Avance Practico en Unity para el diseño y desarrollo de una plataforma base para la implementación de entornos educativos multiusuarios para realidad virtual.

1. Creación de Escenas

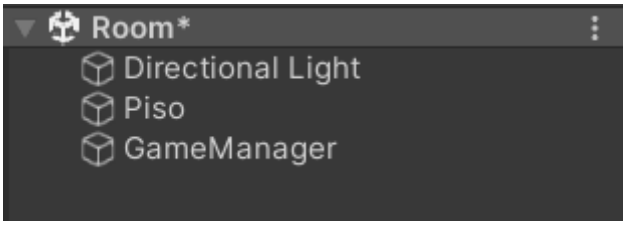


Escena Lobby:



- Escena 2D que contiene solo objetos UI utilizados como canvas para el registro y conexión de usuarios.
- PhotonManager es un objeto vacío que contiene el script que se encarga de la conexión de los usuarios, más abajo hablo específicamente de su contenido.
- Estos canvas no son finales ya que se planea hacer en realidad virtual y por ahora solo funcionan para probar conectividad.

Escena Room:



- Escena 3D que actualmente se encuentra vacía, se la utiliza para spamear los usuarios registrados a una clase, por ahora el ambiente de la escena es simplemente un piso y los usuarios son capsulas.
- GameManager es un objeto vacío que contiene el script que se encarga del spam de los usuarios.

2. Scripts

PhotonManager: [Anexo 1]
Maneja la creación y gestión de rooms en línea utilizando Photon Unity Networking y controla la interfaz de usuario para permitir a los estudiantes unirse a salas, interactuar con otros estudiantes y comenzar la práctica.

1. Importa las bibliotecas necesarias, incluyendo Photon.Pun y Photon.Realtime, para habilitar la funcionalidad de red.
2. Define varios campos públicos para referenciar elementos de la interfaz de usuario (como campos de entrada de texto, paneles y botones) que se utilizarán para interactuar con la plataforma, por ahora.
3. Establece diccionarios para almacenar información sobre los romos de la plataforma y los usuarios en la práctica.
4. En el método Start(), se configuran las inicializaciones iniciales, como configurar el nombre del usuario local y permitir la sincronización automática de las escenas.
5. Hay una serie de métodos públicos llamados OnClickXXX que se ejecutan cuando se hacen clic en botones específicos de la interfaz de usuario. Estos métodos manejan acciones como iniciar sesión, crear una sala, unirse a una sala, salir.
6. El script también incluye una serie de métodos que son llamados automáticamente por Photon en respuesta a eventos de red, como cuando se conecta al servidor, se crea una sala, se unen usuarios y más.
   Estos métodos controlan la lógica de la plataforma y la actualización de la interfaz de usuario.
7. Además, el script define métodos de utilidad para activar y desactivar paneles de la interfaz de usuario y para gestionar las listas de rooms y usuarios.

GameManager: [Anexo 2]
Instanciar el avatar del usuario en el room cuando la practica comienza.

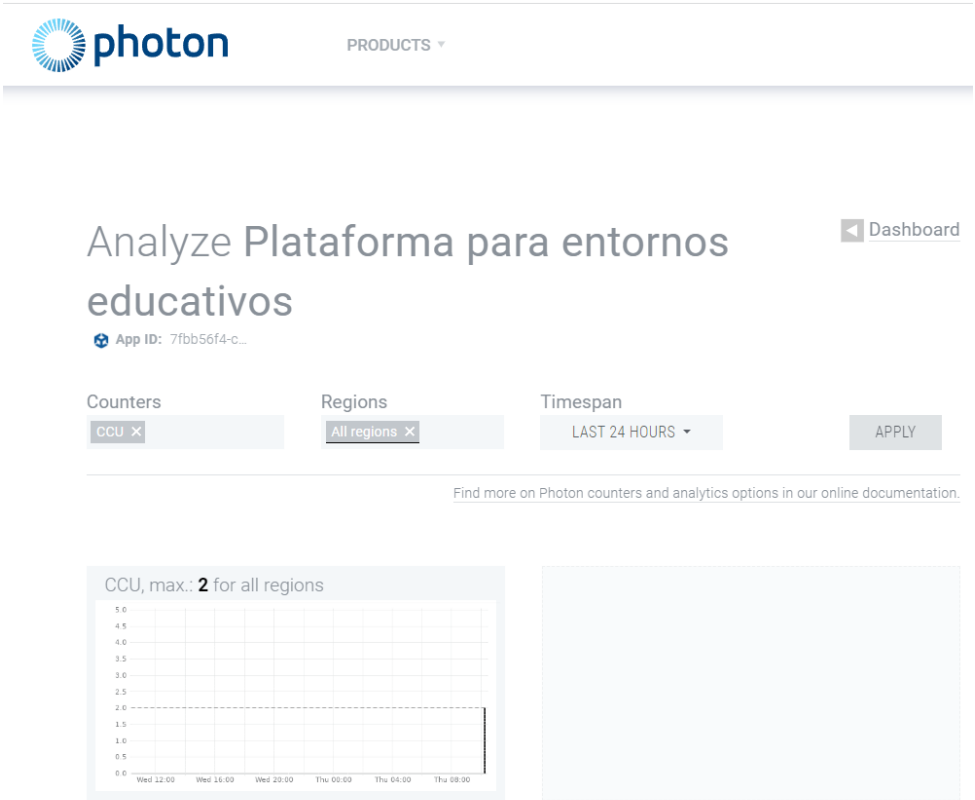
1. Antes de crear el usuario, verifica si PhotonNetwork conectado y listo para usar. Esto garantiza que el programa solo intente crear un usuario en línea cuando esté conectado a Photon.
2. Una vez que se verifica la conexión, el código genera una posición aleatoria para el usuario en el plano XZ (horizontal) y luego utiliza PhotonNetwork.Instantiate para crear una instancia del usuario en esa

posición. Esto significa que cuando los usuarios se unen a la sala, cada uno obtendrá su propio personaje del usuario en la posición especificada.

UserSetUp: [Anexo 3]
Configura la apariencia y el comportamiento de los usuarios locales y remotos en la plataforma multiusuario.

1. Define dos arreglos de GameObjects, localPlayerItems y remotePlayerItems, que contendrá objetos que se deben mostrar u ocultar para el jugador local y remoto respectivamente. Actualmente están vacíos ya que no tienen avatares.
2. También tiene una variable cameraHolder que representa el objeto que contiene la cámara del jugador.
3. En el método Start(), comprueba si el objeto actual es controlado por el jugador local (photonView.IsMine).
4. Si es el jugador local, activa los objetos en localPlayerItems, habilita el componente FirstPersonController para el movimiento y muestra la cámara.
5. Si no es el jugador local, desactiva los objetos en localPlayerItems, deshabilita el componente FirstPersonController y oculta la cámara.

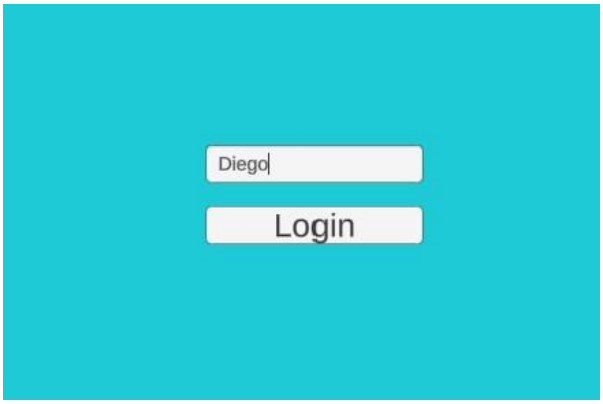
3. Photon Engine

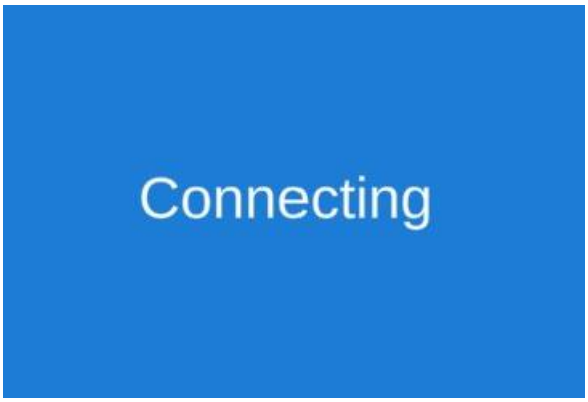


Dashboard propio de Photon Engine que muestra los usuarios conectados en las ultimas 24 horas, en este caso inicialicé la plataforma una sola vez con dos usuarios.

4. **Ejemplo de funcionalidad (por ahora) con imágenes.**

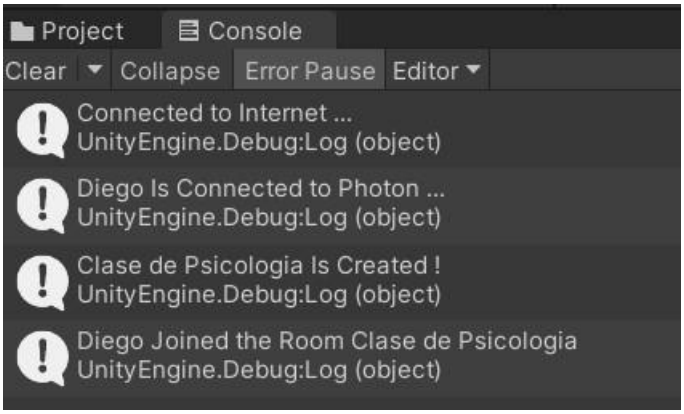
Hay que recordar que los canvas no son finales ya que se realizara más tarde en realidad virtual, solo son usados para pruebas de conectividad y funcionamiento.





Creación de usuario con nombre y pantalla de conexión hasta establecer conexión con internet una vez hecho login.

Creación de un room con nombre del room y cantidad máxima de usuarios.

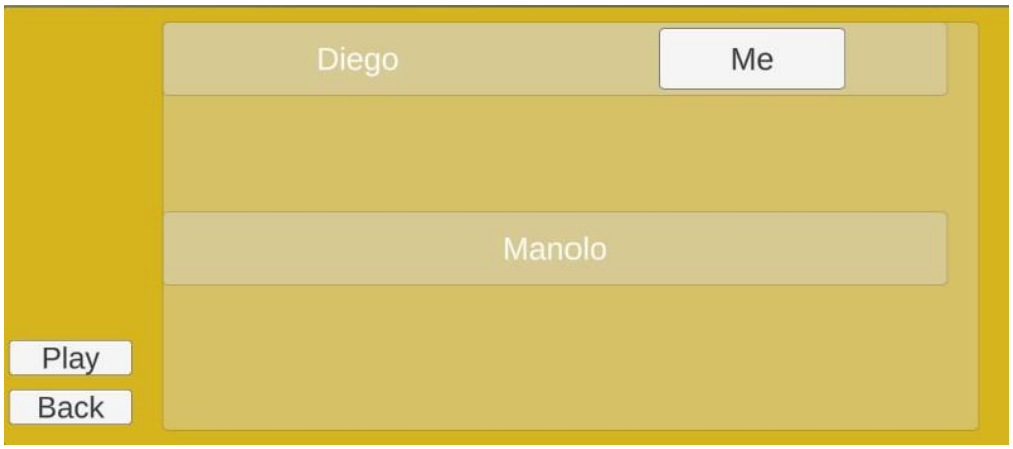


[En este punto, ejecuté la plataforma en otro computador, cree un usuario “Manolo” y presione en el botón “Room List”]

Muestra todas las rooms creadas y la cantidad de usuarios registrados en ese room.

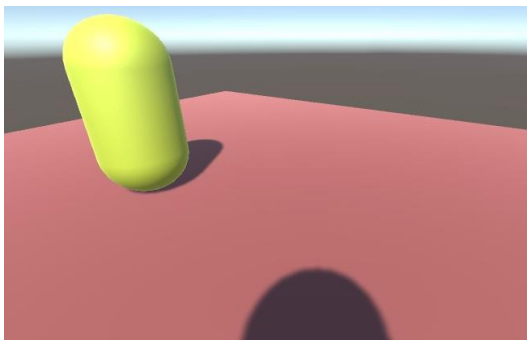
[Presiono “Join Room” desde usuario Manolo]

[Vista desde usuario Diego]



Una vez que Diego (MasterUsuario por crear la room), presione “Play”.

Se spamean los usuarios en el room, cada uno tiene control y vista de su avatar.



ANEXOS

1. **PhotonManager Script:**

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;
using Photon.Pun;
using TMPro;
using Photon.Realtime;

public class PhotonManager : MonoBehaviourPunCallbacks
{

    public TMP_InputField userNameInput;
    public TMP_InputField roomNameInput;
    public TMP_InputField maxUserInput;

    public GameObject userNamePanel;
    public GameObject connectingPanel;
    public GameObject lobbyPanel;
    public GameObject createRoomPanel;
    public GameObject roomListPanel;


    private Dictionary <string, RoomInfo> roomListData;
    private Dictionary<string, GameObject> roomListGameObject;
    private Dictionary<int, GameObject> playerListGameObject;

    public GameObject roomListPrefab;
    public GameObject roomListParent;


    [Header("Inside Room Panel")]
    public GameObject insideRoomPanel;
    public GameObject playerListPrefab;
    public GameObject playerListParent;
    public GameObject playButton;


    #region UnityMethods

    private void Start()
    {
        ActiveMyPanel(userNamePanel.name);
        roomListData = new Dictionary<string, RoomInfo>();
        roomListGameObject = new Dictionary<string, GameObject>();
        PhotonNetwork.AutomaticallySyncScene = true;
    }
    private void Update()
    {
        //Debug.Log("Network state: " + PhotonNetwork.NetworkClientState);
    }

    #endregion


    #region OnClick
    public void OnClickLogin()
    {
        string name  = userNameInput.text;
```

```
    if (!string.IsNullOrEmpty(name))
    {
        PhotonNetwork.LocalPlayer.NickName = name;
        PhotonNetwork.ConnectUsingSettings();
        ActiveMyPanel(connectingPanel.name);
    }
    else
    {
        Debug.Log("--- Empty Name ---");
    }
}


public void OnClickCreateRoom()
{
    string roomName = roomNameInput.text;
    if (string.IsNullOrEmpty(roomName))
    {
        roomName = roomName + Random.Range(0, 20);
    }

    RoomOptions roomOptions = new RoomOptions();
    roomOptions.MaxPlayers = (byte) int.Parse(maxUserInput.text);
    PhotonNetwork.CreateRoom(roomName,roomOptions);
}


public void OnClickCancel()
{
    ActiveMyPanel(lobbyPanel.name);
}

public void OnClickRoomList()
{
    if (!PhotonNetwork.InLobby)
    {
        PhotonNetwork.JoinLobby();
    }

    ActiveMyPanel(roomListPanel.name);
}

public void OnClickBackFromRoomList()
{
    if (PhotonNetwork.InLobby) {
        PhotonNetwork.LeaveLobby();
    }
    ActiveMyPanel(lobbyPanel.name);
}

public void OnClickBackFromPlayerList()
{
    if (PhotonNetwork.InRoom)
    {
        PhotonNetwork.LeaveRoom();
    }
    ActiveMyPanel(lobbyPanel.name);
}

public void OnClickPlayButton()
{
    if (PhotonNetwork.IsMasterClient)
    {
```

```
            PhotonNetwork.LoadLevel("Room");
        }


    }

    #endregion


    #region PhotonServer Callbacks

    public override void OnConnected()
    {
        Debug.Log("Connected to Internet ...");
    }

    public override void OnConnectedToMaster()
    {
        Debug.Log(PhotonNetwork.LocalPlayer.NickName + " Is Connected to
Photon ...");
        ActiveMyPanel(lobbyPanel.name);
    }

    public override void OnCreatedRoom()
    {
        Debug.Log(PhotonNetwork.CurrentRoom.Name + " Is Created !");
    }

    public override void OnJoinedRoom()
    {
        Debug.Log(PhotonNetwork.LocalPlayer.NickName + " Joined the Room " +
PhotonNetwork.CurrentRoom.Name);
        ActiveMyPanel(insideRoomPanel.name);

        if(playerListGameObject == null)
        {
            playerListGameObject = new Dictionary<int, GameObject>();
        }

        //Se activa el boton de play solo para quien creo el room osea el
master
        if (PhotonNetwork.IsMasterClient)
        {
            playButton.SetActive(true);
        }
        else
        {
            playButton.SetActive(false);
        }



        foreach(Player p in PhotonNetwork.PlayerList)
        {
            GameObject playerList = Instantiate(playerListPrefab);
            playerList.transform.SetParent(playerListParent.transform);
            playerList.transform.localScale = Vector3.one;


playerList.transform.GetChild(0).gameObject.GetComponent<TMP_Text>().text =
p.NickName;

            if(p.ActorNumber == PhotonNetwork.LocalPlayer.ActorNumber)
```

```
            {
                playerList.transform.GetChild(1).gameObject.SetActive(true);
            }
            else
            {

playerList.transform.GetChild(1).gameObject.SetActive(false);
            }

            playerListGameObject.Add(p.ActorNumber, playerList);
        }
    }


    public override void OnPlayerEnteredRoom(Player newPlayer)
    {
        GameObject playerList = Instantiate(playerListPrefab);
        playerList.transform.SetParent(playerListParent.transform);
        playerList.transform.localScale = Vector3.one;


playerList.transform.GetChild(0).gameObject.GetComponent<TMP_Text>().text =
newPlayer.NickName;

        if (newPlayer.ActorNumber == PhotonNetwork.LocalPlayer.ActorNumber)
        {
            playerList.transform.GetChild(1).gameObject.SetActive(true);
        }
        else
        {
            playerList.transform.GetChild(1).gameObject.SetActive(false);
        }

        playerListGameObject.Add(newPlayer.ActorNumber, playerList);
    }

    public override void OnPlayerLeftRoom(Player otherPlayer)
    {
        Destroy(playerListGameObject[otherPlayer.ActorNumber]);
        playerListGameObject.Remove(otherPlayer.ActorNumber);
    }


    public override void OnLeftRoom()
    {
        ActiveMyPanel(lobbyPanel.name);
        foreach(GameObject obj in playerListGameObject.Values)
        {
            Destroy(obj);
        }
    }



    public override void OnRoomListUpdate(List<RoomInfo> roomList)
    {
        ClearRoomList();

        foreach(RoomInfo rooms in roomList)
        {
            Debug.Log("Room name: " + rooms.Name);
            if(!rooms.IsOpen || !rooms.IsVisible || rooms.RemovedFromList)
            {
```

```
                if (roomListData.ContainsKey(rooms.Name))
                {
                    roomListData.Remove(rooms.Name);
                }
            }
            else
            {
                if (roomListData.ContainsKey(rooms.Name))
                {
                    //Update List
                    roomListData[rooms.Name] = rooms;
                }
                else
                {
                    roomListData.Add(rooms.Name, rooms);
                }
            }

        }

        foreach(RoomInfo roomItem in roomListData.Values)
        {
            GameObject roomListItemObject = Instantiate(roomListPrefab);

roomListItemObject.transform.SetParent(roomListParent.transform);
            roomListItemObject.transform.localScale = Vector3.one;

roomListItemObject.transform.GetChild(0).gameObject.GetComponent<TMP_Text>()
.text = roomItem.Name;

roomListItemObject.transform.GetChild(1).gameObject.GetComponent<TMP_Text>()
.text = roomItem.PlayerCount + "/" + roomItem.MaxPlayers;

roomListItemObject.transform.GetChild(2).gameObject.GetComponent<Button>().o
nClick.AddListener(() => RoomJoinFromList(roomItem.Name));
            roomListGameObject.Add(roomItem.Name, roomListItemObject);

        }

    }

    public override void OnLeftLobby()
    {
        ClearRoomList();
        roomListData.Clear();
    }


    #endregion


    #region Public Methods


    public void RoomJoinFromList(string roomName)
    {
        if (PhotonNetwork.InLobby)
        {
            PhotonNetwork.LeaveLobby();
        }
        PhotonNetwork.JoinRoom(roomName);
    }
```

```
    public void ClearRoomList()
    {
        if(roomListGameObject.Count > 0)
        {
            foreach (var v in roomListGameObject.Values)
            {
                Destroy(v);
            }
            roomListGameObject.Clear();
        }

    }

    public void ActiveMyPanel(string panelName)
    {
        userNamePanel.SetActive(panelName.Equals(userNamePanel.name));
        lobbyPanel.SetActive(panelName.Equals(lobbyPanel.name));
        createRoomPanel.SetActive(panelName.Equals(createRoomPanel.name));
        connectingPanel.SetActive(panelName.Equals(connectingPanel.name));
        roomListPanel.SetActive(panelName.Equals(roomListPanel.name));
        insideRoomPanel.SetActive(panelName.Equals(insideRoomPanel.name));
    }

    #endregion

}
```

2. **GameManager Script:**

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using Photon.Pun;

public class GameManager : MonoBehaviour
{

    public GameObject playerPrefab;


    void Start()
    {
        if (PhotonNetwork.IsConnectedAndReady)
        {
            int randomNumber = Random.Range(-4, 4);
            PhotonNetwork.Instantiate(playerPrefab.name, new
Vector3(randomNumber, 1f, randomNumber),Quaternion.identity);
        }
    }


}
```

3. **UserSetUp Script:**

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using Photon.Pun;
using UnityStandardAssets.Characters.FirstPerson;

public class UserSetUp : MonoBehaviourPunCallbacks
{
    //Cuando ya tenga avatares, las cosas que quiero mostrar y que no
    public GameObject[] localPlayerItems;
    public GameObject[] remotePlayerItems;
    public GameObject cameraHolder;

    void Start()
    {
        if (photonView.IsMine)
        {
            /*
            foreach (GameObject g in localPlayerItems)
            {
                g.SetActive(true);
            }
            foreach (GameObject g in remotePlayerItems)
            {
                g.SetActive(false);
            }
            */
            GetComponent<FirstPersonController>().enabled = true;
//Movimiento de cada user
            cameraHolder.SetActive(true);
        }
        else
        {
            /*
            foreach (GameObject g in localPlayerItems)
            {
                g.SetActive(false);
            }
            foreach (GameObject g in remotePlayerItems)
            {
                g.SetActive(true);
            }
            */
            GetComponent<FirstPersonController>().enabled = false;
            cameraHolder.SetActive(false);
        }
    }

}
```