

DOCUMENTACIÓ TÈCNICA

| | |
|--|----------|
| 1. Resum del projecte..... | 4 |
| Resum:..... | 4 |
| Objectiu del projecte:..... | 4 |
| Públic objectiu:..... | 4 |
| 2. Informació general | 5 |
| Nom del projecte: Lethal Run..... | 5 |
| Plataforma: Ordinador/PC..... | 5 |
| Versió: 1.0..... | 5 |
| Autors: Brian Orozco, Izan de la Cruz, David Salvador, Mar Rojano..... | 5 |
| Requisits mínims..... | 5 |
| Lethal Run:..... | 5 |
| 3. Estructura del projecte | 5 |
| Estructura de carpetas:..... | 5 |
| Carpetas principals:..... | 5 |
| 4. Escenes i nivells..... | 7 |
| Flux d'escenes:..... | 7 |
| Title Screen:..... | 8 |
| ◆ Presentació i ambientació:..... | 8 |
| ◆ Logo i menú principal :..... | 8 |
| ◆ Personatges icònics:..... | 8 |
| ◆ Elements visuals i efectes:..... | 8 |
| ◆ Interfície d'usuari (UI):..... | 8 |
| ◆ Objectes Importants:..... | 8 |
| ◆ Experiència i immersió:..... | 9 |
| ◆ Transició a la següent escena:..... | 9 |
| Menú Principal:..... | 9 |
| ◆ Presentació i ambientació:..... | 9 |
| ◆ Opcions del menú:..... | 9 |
| ◆ Objectes importants:..... | 10 |
| ◆ Transicions entre escenes:..... | 10 |
| ◆ Experiència i immersió:..... | 10 |
| Lobby:..... | 11 |
| ◆ Presentació i ambientació:..... | 11 |
| ◆ Funcionalitats principals:..... | 11 |
| ◆ Objectes importants:..... | 12 |
| ◆ Experiència i immersió:..... | 12 |
| Mapa 1:..... | 12 |

| | |
|---------------------------------|-----------|
| ◆ Presentació i ambientació: | 12 |
| ◆ Trampas: | 13 |
| ◆ Experiència i immersió: | 14 |
| Mapa 2: | 15 |
| ◆ Presentació i ambientació: | 15 |
| ◆ Trampas: | 15 |
| ◆ Experiència i immersió: | 15 |
| 5. Prefabs/Interfície | 16 |
| 6. Arquitectura del codi | 16 |
| UI: | 16 |
| TitleScreen Manager: | 16 |
| UISALA: | 26 |
| OptionsMenuControllerSala: | 37 |
| ControlesScriptUI: | 42 |
| CanvasLobbyUIToolkit: | 44 |
| MapSelector: | 46 |
| OpcionesMenu(In Game): | 53 |
| WinCondition: | 61 |
| Personajes: | 65 |
| StarterAssets: | 65 |
| ThirdPersonCharacterController: | 69 |
| PlayerHealth: | 81 |
| Controller(RingEmote): | 84 |
| Ring Menu (Emotes/graffitis): | 86 |
| Trampas: | 93 |
| LavaTrap: | 93 |
| TrampaPinchos: | 96 |
| BolaMovimiento: | 99 |
| Multijugador: | 105 |
| NetworkManager: | 105 |

1. Resum del projecte.

Resum:

Lethal Run és un joc 3D desenvolupat amb Unity que combina acció multijugador i estratègia asimètrica. En cada partida, un jugador assumeix el rol de “trumper”, encarregat d’activar trampes al llarg del recorregut, mentre que la resta de jugadors competeixen entre ells per arribar a la meta en el menor temps possible. El joc destaca per una estètica animada, amb escenaris vius, i per la possibilitat que els jugadors utilitzin emoticons i grafits per interactuar durant la partida. La dinàmica principal gira entorn de la cooperació i la competició, oferint una experiència intensa i divertida tant per als qui esquiven trampes com per a qui les activa.

Objectiu del projecte:

L’objectiu és crear un joc dinàmic i accessible que combini estratègia, habilitat i competició, fomentant la interacció social i la rejugabilitat mitjançant partides ràpides i rols asimètrics.

Públic objectiu:

Lethal Run està adreçat a jugadors casuals i competitius, principalment adolescents i joves adults (de 12 a 30 anys) que gaudeixen d’experiències multijugador ràpides, dinàmiques i amb un fort component social.

2. Informació general

Nom del projecte: Lethal Run

Plataforma: Ordinador/PC

Versió: 1.0

Autors: Brian Orozco, Izan de la Cruz, David Salvador, Mar Rojano

Requisits mínims

Lethal Run:

- ◆ Sistema operatiu: Windows 10 (64 bits)
- ◆ Processador: Intel Core i5 (o equivalent)
- ◆ RAM: 8 GB
- ◆ Gràfics: NVIDIA GTX 960 / AMD Radeon R9 280 (o similar)
- ◆ Emmagatzematge: 1 GB lliure

3. Estructura del projecte

Estructura de carpetas:

Carpetas principales:

- ◆ Scenes: Contiene todas las escenas del juego.

- ◆ Scripts: Almacena todos los scripts de C# utilizados en el proyecto.

- ◆ Prefabs: Contiene los prefab reutilizables.
- ◆ Character: Contiene assets relacionados con los personajes del juego.
- ◆ Environment: Contiene assets para los entornos del juego (modelos, texturas, etc.).
- ◆ Textures: Almacena texturas generales utilizadas en el proyecto.
- ◆ Tree_Textures: Texturas específicas de árboles.
- ◆ InputSystem: Contiene el sistema de input del juego.
- ◆ Settings: Contiene archivos de configuración del proyecto.
- ◆ Resources: Assets cargados dinámicamente en tiempo de ejecución.
- ◆ TutorialInfo: Assets para tutoriales.
- ◆ HDRPDefaultResources: Recursos predeterminados del High Definition Render Pipeline.
- ◆ Mario: Assets específicos de un personaje llamado Mario.

- ◆ Mobile: Assets optimizados para plataformas móviles.

- ◆ RingEmote: Assets relacionados con emotes de anillos.

Assets de terceros (Plugins):

- ◆ NVJOB Water Shaders V2: Shaders para efectos de agua.

- ◆ com.rlabrecque.steamworks.net: Plugin para integrar Steamworks.

- ◆ Mirror: Framework de networking multijugador.

- ◆ simple UI & icons: Assets de interfaz de usuario.

- ◆ TextMesh Pro: Plugin para texto avanzado.

- ◆ UI Toolkit: Sistema de UI de Unity.[Unity](#).

4. Escenes i nivells

Flux d'escenes:

Title Screen:**◆ Presentació i ambientació:**

L'escena d'inici dóna la benvinguda al jugador amb una atmosfera vibrant i plena de personalitat. El fons fosc amb detalls daurats crea una sensació èpica i misteriosa que prepara el terreny per a l'aventura.

◆ Logo i menú principal :

Al centre de la pantalla, el logo imponent destaca el nom del joc, mentre que el menú ofereix opcions clares i accessibles: registrar-se, iniciar sessió o començar a jugar amb el botó "PLAY".

◆ Personatges icònics:

A banda i banda del menú, es mostren figures 3D de personatges emblemàtics dels diferents universos del joc. Aquesta disposició anticipa la varietat i el crossover que el jugador experimentarà.

◆ Elements visuals i efectes:

Estrelles flotants i efectes de llum i reflexos aporten profunditat i realisme a l'escena, fent-la més atractiva i immersiva.

◆ Interfície d'usuari (UI):

La UI és clara, senzilla i està integrada de manera harmoniosa amb l'escena, sense restar protagonisme ni al logo ni als personatges.

◆ Objectes Importants:

- Objectes Decoratius
- UI Document (Boton de Jugar i títol del joc)
- UI Document(1) (Login)
- Cursor

◆ Experiència i immersió:

Tots aquests elements treballen conjuntament per submergir el jugador en l'univers de Lethal Run des del primer moment, convidant-lo a iniciar una experiència única i plena d'acció.

◆ Transició a la següent escena:

Quan el jugador prem **Play** i es logeja et porta a la següent escena que es el **Menú Principal** també té un botón de **Exit** que et surt del joc.

Menú Principal:

◆ Presentació i ambientació:

El Menú Principal de Lethal Run manté la mateixa atmosfera fosca i enèrgica de la pantalla de títol, amb el protagonista en primer pla i el logo dominant la part superior. Aquesta coherència visual reforça la identitat del joc i prepara el jugador per a l'acció.

◆ Opcions del menú:

- **Start Host:** Permet iniciar una partida com a amfitrió. En seleccionar aquesta opció, el joc et porta directament a l'escena de lobby o sala d'espera, on podràs esperar que altres jugadors s'uneixin abans de començar la partida.
- **Join Host:** Dona accés a la llista de sales disponibles. Si no hi ha jugadors actius, la pantalla apareix buida. Un cop hi hagi sales disponibles, es mostraran aquí perquè el jugador pugui unir-se a una partida en curs tenen 3 principals botons el de **Update** per actualitzar la llista de servidors **Join** per entrar a una sala i **Exit** per tornar enrere al menú principal.

- **Options:** Obre una finestra emergent on el jugador pot ajustar el volum, la qualitat gràfica i el mode de pantalla (complet o finestra). Aquesta finestra permet aplicar canvis, sortir sense guardar o tornar al menú principal.
- **Controls:** Mostra una pantalla amb el mapa de controls del joc. Aquí el jugador pot consultar ràpidament com moure's, atacar, acceder al menú d'opcions, etc. Aquesta informació és clau per a nous jugadors i per recordar combinacions importants.
- **Exit:** Tanca el joc i retorna el jugador a la pantalla de títol, permetent sortir del menú principal de forma ràpida i clara.

◆ Objectes importants:

- **Diferents UI Documents (Options, Controls etc..):**
 - Estética del proyecto
- **Steam Manager:**
 - Component que s'encarrega d'inicialitzar i gestionar la connexió del teu joc amb la plataforma Steam mitjançant Steamworks.NET.
- **AudioStart:**
 - Simplement per la implementació de la música al joc

◆ Transicions entre escenes:

- **Start Host:** Porta a l'escena de lobby/sala d'espera.
- **Join Host:** Porta a la pantalla de sales disponibles.
- **Options:** Mostra la finestra de configuració sobreposada i permet tornar al menú principal.
- **Controls:** Mostra la finestra de controls sobreposada
- **Exit:** Retorna a la pantalla de títol.

◆ Experiència i immersió:

El menú principal està dissenyat per ser intuïtiu i ràpid, amb una navegació fluida entre opcions i una presentació visual impactant. Això assegura que el jugador pugui accedir fàcilment a totes les funcionalitats bàsiques abans de començar la partida, mantenint sempre la immersió en l'univers de Lethal Run.

Lobby:

◆ Presentació i ambientació:

El Lobby és l'espai virtual on els jugadors esperen abans de començar la partida. L'ambientació manté el mateix estil visual: parets de maó, sòl verd i un entorn obert que permet moure's lliurement. Aquesta sala serveix com a punt de trobada i socialització abans de l'acció.

◆ Funcionalitats principals:

- **Moviment lliure:**

- Els jugadors poden moure el seu personatge pel lobby i veure en temps real els altres jugadors que es van unint. Això afegeix dinamisme i sensació de comunitat abans de començar la partida.

- **Selecció de mapa:**

- El host pot obrir el menú de selecció de camp de batalla, escollir entre diferents mapes (com "Mario World") i consultar informació sobre cada escenari (dificultat, terreny, mida, etc.).

- **Menú d'emotes i grafitis:**

- Prement el tabulador, s'obre un menú radial on els jugadors poden seleccionar gestos o grafitis per expressar-se dins del lobby.

- **Controls:**

- Si el jugador obre el menú de controls, es mostra una pantalla amb la informació dels controls bàsics del joc.

◆ Botons i accions:

- **Start Host (només host):**
 - Quan el host selecciona un mapa i prem aquest botó, tots els jugadors són transportats a l'escenari seleccionat i comença la partida.
- **Exit (host):**
 - Si el host prem "EXIT", tots els jugadors (host i clients) són expulsats del lobby i tornen al menú principal.
- **Exit (client):**
 - Si un client prem "EXIT", només ell surt del lobby i torna al menú principal; la resta de jugadors es queden a la sala.

◆ Objectes importants:

- **UI Document (Elecció de mapa)**
 - Per poder elegir el mapa
- **Canvas(RingMenu)**
 - Per la ruleta de balls i grafitis
- **MapSelector**
 - Es una classe per guardar el mapa seleccionat
- **AudioSource:**
 - Per implementar música a la lobby

◆ Experiència i immersió:

El lobby està pensat per fomentar la interacció i l'espera activa. Veure altres jugadors en temps real, poder expressar-se amb gestos i tenir una transició fluida cap a l'acció fa que l'espera sigui entretinguda i immersiva. El control clar dels rols (host/client) garanteix una experiència d'usuari intuïtiva i sense confusions.

Mapa 1:

◆ Presentació i ambientació:

El Mapa 1 l'escenari combina elements visuals de l'univers Mario amb un recorregut ple de trampes, obstacles i desafiaments. L'ambientació és acolorida però amb un toc de tensió, ja que cada pas pot ser decisiu per la supervivència dels jugadors.

◆ Trampas:

- **Portes:**
 - Trata de dos portes que simplement si pases per la incorrecte el tepeas a uns punxos
- **Punxos:**
 - Simplement el que fa és que s'activen uns punxos que surten de sota terra.
- **Bolas de Punxos:**
 - El que fa és que solta una bola de punxos que el que fan es arrasar amb qui estigui en la trayectoria
- **Paret que empeny:**
 - El que fa es que empeny al jugador
- **Boles de foc:**
 - El que fa es que s'activen unes boles de foc rotatorias que el que fan es fer mal al jugador
- **Laberint:**
 - El que fa es que es un laberint ple de fantasma que si els mires es quedan quiets i si no et segueixen i l'objectiu és trobar les 3 monedes per sortir de el laberint
- **Plataformes:**
 - El que fa és que simplement el que active trampes li dona el botó i fa desaparèixer la plataforma
- **Cursa final:**
 - Es una cursa entre totss els jugados per veure qui és el guanyador

◆ Objectes Importants:

- **Spawn Points:**
 - Es on si passen per damunt si moren apareixen ahí
- **Spawn Jugadors:**

- Es on comencen tots el jugadors
- **UI Document (Options):**
 - Es la ui per canvia volumen resolucio etc en el mapa
- **Canvas(Vida):**
 - Es la vida del jugador
- **LavaDetector:**
 - Es el que fa que moris per la lava

◆ Experiència i immersió:

El Mapa 1 ofereix una experiència intensa i divertida, plena d'adrenalina i sorpreses. La combinació d'obstacles, trampes i la temàtica Mario genera una atmosfera única que convida a repetir la partida per millorar el temps o desafiar altres jugadors.

Mapa 2:

◆ Presentació i ambientació:

El Mapa 2 l'escenari combina elements visuals de l'univers Mario amb un recorregut ple de trampes, obstacles i desafiaments. L'ambientació és acolorida però amb un toc de tensió, ja que cada pas pot ser decisiu per la supervivència dels jugadors.

◆ Trampas:

- **Terra movable:**
 - Es un terra que es mou i deixa caure a la gent al aigua
- **Càpsula d'aigua:**
 - Tanca els jugados per omplir la càpsula d'aigua i matarlo
- **Paret que empeny:**
 - El que fa es que empeny al jugador a l'aigua
- **Bola de foc Grand :**
 - Es una bola de foc que fa en recorregut i tens que anar de isla en isla per salvarte
- **Paret que empeny:**
 - El que fa es que empeny al jugador a l'aigua

◆ Objectes Importants:

- **Spawn Points:**
 - Es on si passen per damunt si moren apareixen ahí
- **Spawn Jugadors:**
 - Es on comencen tots el jugadors
- **UI Document (Options):**
 - Es la ui per cambia volumen resolucio etc en el mapa
- **Canvas(Vida):**
 - Es la vida del jugador
- **LavaDetector:**
 - Es el que fa que moris per la lava

◆ Experiència i immersió:

El Mapa 2 ofereix una experiència intensa i divertida, plena d'adrenalina i sorpreses. La combinació d'obstacles, trampes i la temàtica Mario genera una atmosfera única que convida a repetir la partida per millorar el temps o desafiar altres jugadors.

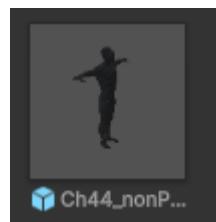
5. Prefabs/Interfície

Skins:

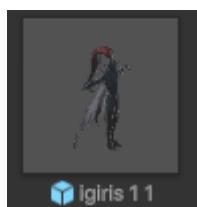
Default2: Skin Comuna és molt comú en les partides



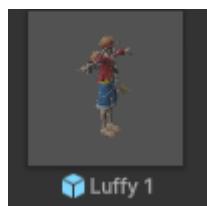
Default: Skin predeterminada per el joc



Igiris: Skin Mítica amb un 1% de que et toqui sortida de el anime Solo Leveling



Luffy: Skin Mítica amb un 1% de que et toqui sortida de el anime One Piece



LETHAL RUN



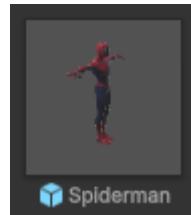
Izan De La Cruz

David Salvador

Marc Rojano

Brian Orozco

Spiderman: Skin Legendaria amb un 4% de que et toqui sortida del mon de Marvel



Spiderman

Megumin: Skin Epic amb un 15% de que et toqui sortida del anime Konosuba



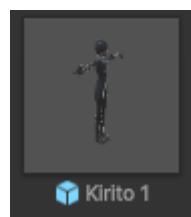
Megumin 1

Asuna: Skin Rara amb un 30% de que et toqui sortida del anime SAO

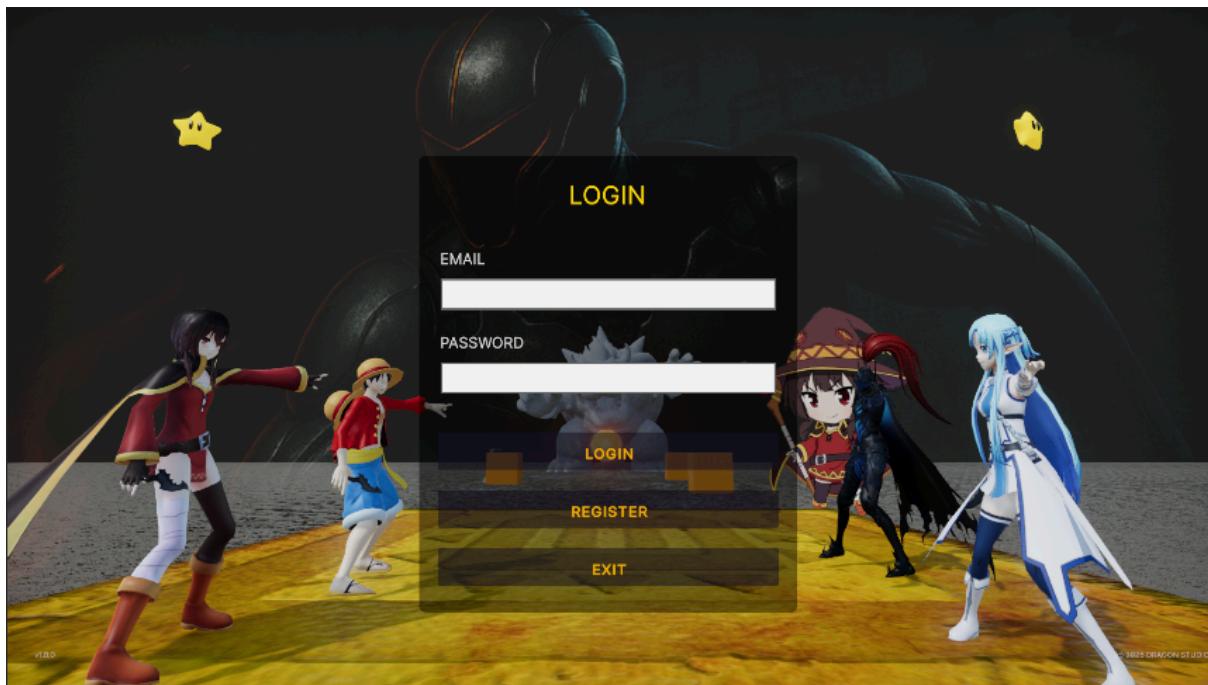


q 1

Kirito: Skin Rara amb un 30% de que et toqui sortida del anime SAO



Kirito 1

Interfície:**Menú inicial i Login:**

Es on t'allotges et registres i sortida

Menú Principal:

Menu on està el start host el join les opciones i el controls.

LETHAL RUN



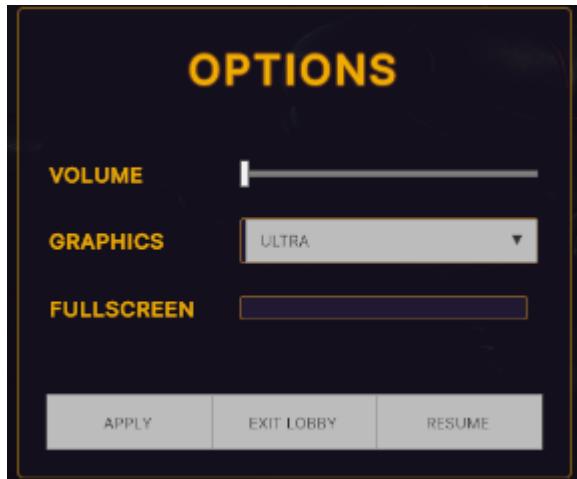
Izan De La Cruz

David Salvador

Marc Rojano

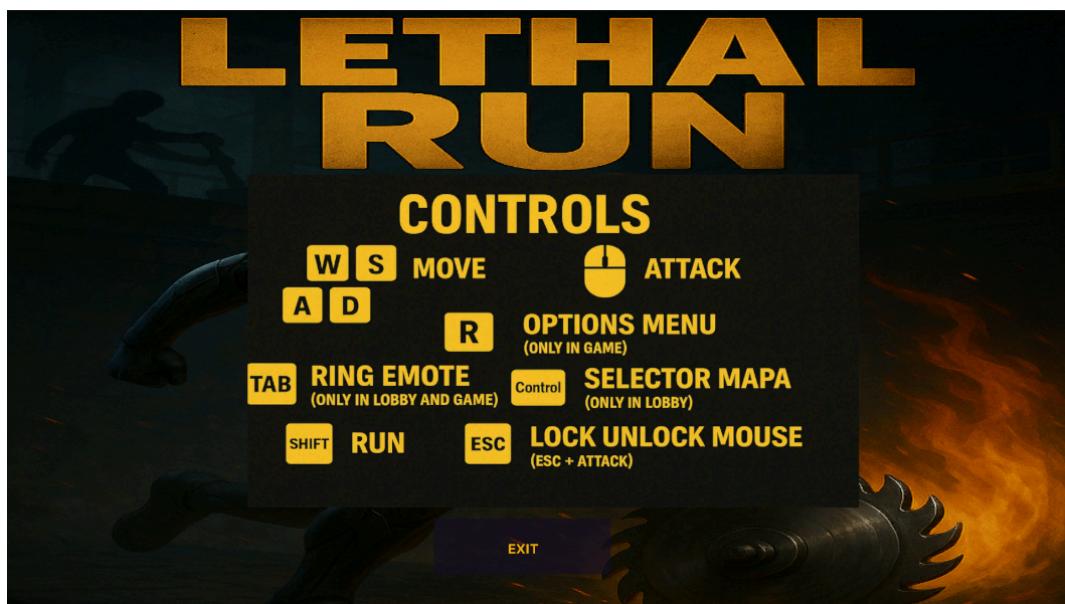
Brian Orozco

Opcions:



Menú per pujar volum millorar graphics i fullScreen

Controls:



LETHAL RUN



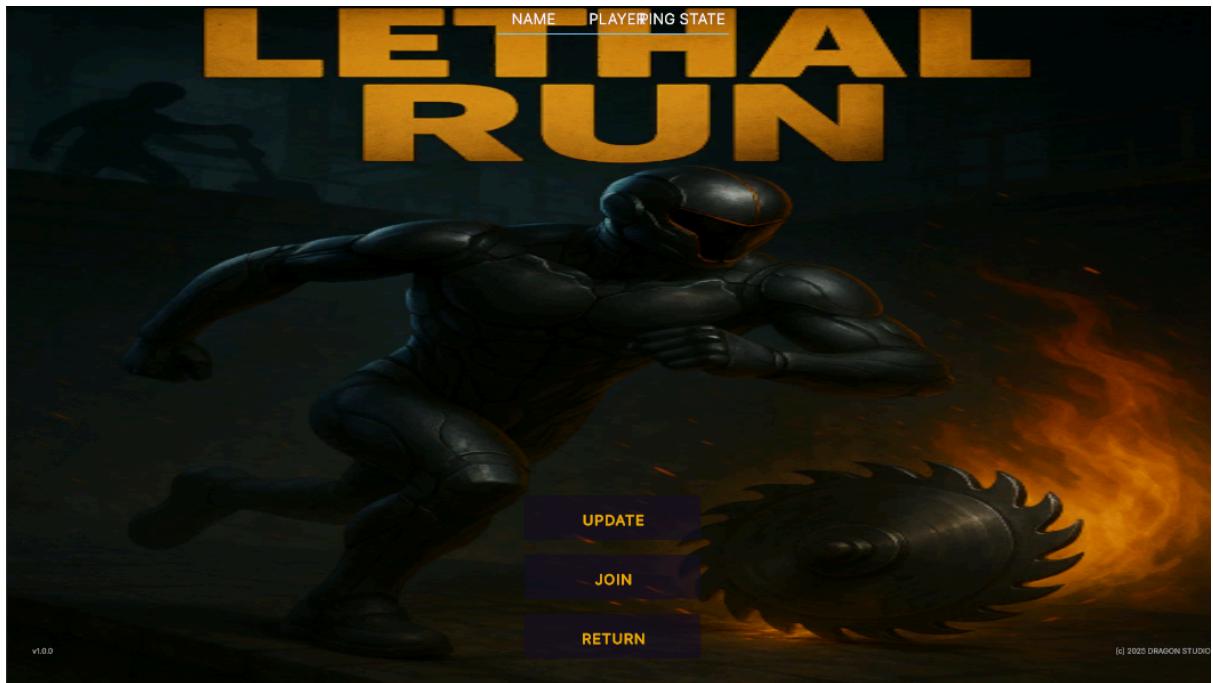
Izan De La Cruz

David Salvador

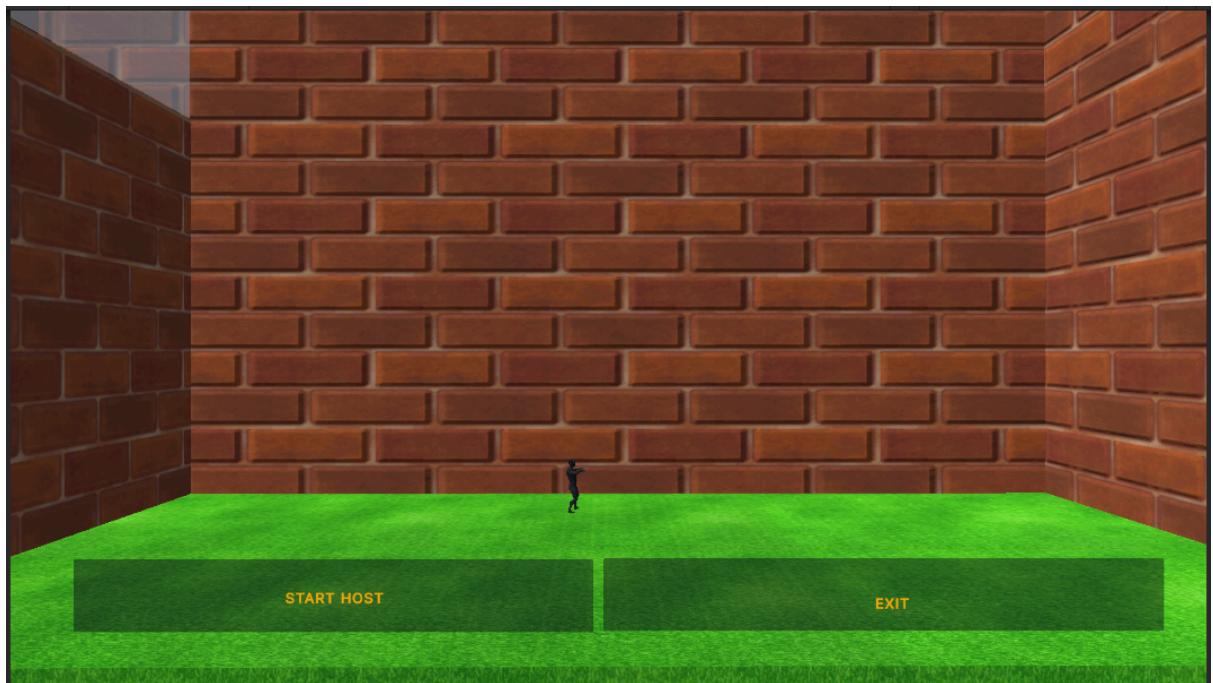
Marc Rojano

Brian Orozco

Menú Sales:



Lobby:



Important:**Funcionalitats principals**

- Autenticació de jugadors:
En connectar-se, cada client envia el seu email i token d'autenticació al servidor, que els desa per gestionar la personalització i la seguretat de la partida.
- Assignació de punts d'aparició (spawns):
Quan comença la partida, el servidor assigna a cada jugador un punt d'aparició. Un jugador és seleccionat aleatoriament com a "trap" (trampa), la resta reben punts normals.
- Selecció dinàmica de l'aspecte (skin):
El servidor contacta amb una API externa per determinar quin aspecte (skin) ha d'utilitzar cada jugador segons el seu compte. El prefab del jugador s'ajusta en funció de la resposta de l'API.
- Gestió de l'estat de la sala i escenes:
El script controla qui pot entrar a la partida segons l'escena actual (només permet nous jugadors al lobby) i actualitza la informació del lobby a Steam.
- Gestió robusta de connexions:
Si un jugador intenta unir-se quan la partida ja ha començat (fora del lobby), la connexió és rebutjada automàticament.

6. Arquitectura del codi

En aquest apartat explicarem com està estructurat el codi font del projecte Lethal Run, detallant els scripts principals. L'objectiu és facilitar la comprensió, el manteniment i l'escalabilitat del codi per a qualsevol membre actual o futur de l'equip de desenvolupament.

UI:**TitleScreen Manager:**

Aquest script gestiona la pantalla de títol i el menú principal d'un joc a Unity. S'encarrega de mostrar el menú, reproduir música de fons, gestionar l'inici de sessió de l'usuari i la transició a l'escena següent després d'un login exitós. A més, administra la interacció amb els botons de la interfície i el registre de nous usuaris.

```
using UnityEngine;
using UnityEngine.UIElements;
using UnityEngine.InputSystem;
using System.Collections;
using UnityEngine.Networking;
using UnityEngine.SceneManagement;
using UnityEngine.Audio;
```

```
/// <summary>
    /// Gestiona la pantalla de título, el menú principal, el login y la
    /// transición de escenas.
/// </summary>
public class TitleScreenManager : MonoBehaviour
{
    private UIDocument uiDocument;
    private VisualElement root;
    private Button playButton;
    private VisualElement mainMenu;

    [Header("UI Toolkit Login Panel")]
    [SerializeField]
    private GameObject loginPanelObject;

    [Header("Registro")]
    [SerializeField]
    private string registerUrl = "https://lethalrun.cat/auth/login";
```

```
[Header("Input & Audio")]
[SerializeField]
private InputActionReference jumpAction;

[SerializeField]
private AudioClip backgroundMusic;

[SerializeField]
private AudioClip buttonClickSound;

[SerializeField]
private float blinkDuration = 0.5f;

[SerializeField]
private int blinkCount = 3;

[SerializeField]
private AudioMixerGroup masterMixerGroup;

[Header("Escena a cargar tras login")]
[SerializeField]
private string nextScene = "NombreDeTuEscena"; // Cambia por el
nombre real

private AudioSource musicPlayer;
private AudioSource sfxPlayer;
private bool isTransitioning = false;

// Referencias a elementos del login
private UIDocument loginPanelDocument;
private VisualElement loginPanel;
private Button loginButton;
private Button registerButton;
private Button exitButton;
private TextField usernameField;
private TextField passwordField;

/// <summary>
/// Inicializa el menú principal y el audio al habilitar el objeto.
```

```
/// </summary>
private void OnEnable()
{
    uiDocument = GetComponent<UIDocument>();
    root = uiDocument.rootVisualElement;

    SetupAudio();
    SetupUI();
    SetupInput();

    if (loginPanelObject != null)
        loginPanelObject.SetActive(false);
}

/// <summary>
/// Configura los reproductores de audio y aplica el volumen
guardado.
/// </summary>
private void SetupAudio()
{
    musicPlayer = gameObject.AddComponent< AudioSource >();
    musicPlayer.clip = backgroundMusic;
    musicPlayer.loop = true;
    musicPlayer.outputAudioMixerGroup = masterMixerGroup;
    musicPlayer.Play();

    sfxPlayer = gameObject.AddComponent< AudioSource >();
    sfxPlayer.outputAudioMixerGroup = masterMixerGroup;

    if (masterMixerGroup != null && masterMixerGroup.audioMixer != null)
    {
        AudioUtils.ApplySavedVolumeToMixer(masterMixerGroup.audioMixer);
    }
}

/// <summary>
/// Configura la interfaz de usuario principal y los callbacks de
botones.
```

```
/// </summary>
private void SetupUI()
{
    playButton = root.Q<Button>("play-button");
    mainMenu = root.Q<VisualElement>("main-menu");

    if (playButton != null)
    {
        playButton.clicked += OnPlayButtonClicked;
        playButton.RegisterCallback<PointerDownEvent>(evt =>
        {
            if ((evt.button == 1 || evt.button == 2) &&
playButton.enabledSelf && playButton.visible && !isTransitioning)
            {
                OnPlayButtonClicked();
            }
        });
    }
}

/// <summary>
/// Configura la acción de salto para activar el botón de jugar.
/// </summary>
private void SetupInput()
{
    if (jumpAction != null && jumpAction.action != null)
    {
        jumpAction.action.Enable();
        jumpAction.action.performed += ctx =>
OnJumpActionPerformed();
    }
}

/// <summary>
/// Deshabilita la acción de salto al desactivar el objeto.
/// </summary>
private void OnDisable()
{
    if (jumpAction != null && jumpAction.action != null)
    {
```

LETHAL RUN



Izan De La Cruz

David Salvador

Marc Rojano

Brian Orozco

```
        jumpAction.action.Disable();
                jumpAction.action.performed -= ctx =>
OnJumpActionPerformed();
    }
}

/// <summary>
/// Maneja la acción de salto para simular el clic en el botón de
jugar.
/// </summary>
private void OnJumpActionPerformed()
{
    if (playButton != null && playButton.enabledSelf &&
playButton.visible && !isTransitioning)
    {
        OnPlayButtonClicked();
    }
}

/// <summary>
/// Callback del botón de jugar: reproduce sonido y muestra el panel
de login.
/// </summary>
private void OnPlayButtonClicked()
{
    if (!isTransitioning)
    {
        isTransitioning = true;
        PlayButtonClickSound();
        StartCoroutine(BlinkButtonAndShowLoginPanel());
    }
}

/// <summary>
/// Reproduce el sonido de clic del botón.
/// </summary>
private void PlayButtonClickSound()
{
    if (sfxPlayer != null && buttonClickSound != null)
        sfxPlayer.PlayOneShot(buttonClickSound);
```

```
}
```

```
/// <summary>
/// Realiza el parpadeo del botón y muestra el panel de login.
/// </summary>
private IEnumerator BlinkButtonAndShowLoginPanel()
{
    if (playButton != null)
    {
        for (int i = 0; i < blinkCount; i++)
        {
            playButton.style.opacity = 0;
            yield return new WaitForSeconds(blinkDuration / 2);
            playButton.style.opacity = 1;
            yield return new WaitForSeconds(blinkDuration / 2);
        }
    }

    ShowLoginPanel();
}

/// <summary>
/// Muestra el panel de login y oculta el menú principal.
/// </summary>
private void ShowLoginPanel()
{
    if (mainMenu != null)
        mainMenu.style.display = DisplayStyle.None;

    if (jumpAction != null && jumpAction.action != null)
        jumpAction.action.Disable();

    if (loginPanelObject != null)
    {
        loginPanelObject.SetActive(true);

        loginPanelDocument = loginPanelObject.GetComponent<UIDocument>();
        if (loginPanelDocument != null)
        {
```

```
var loginRoot = loginPanelDocument.rootVisualElement;
loginPanel = loginRoot.Q<VisualElement>("login-panel");
loginButton = loginRoot.Q<Button>("login-button");
registerButton = loginRoot.Q<Button>("register-button");
exitButton = loginRoot.Q<Button>("exit-button");
usernameField = loginRoot.Q<TextField>("username-field");
passwordField = loginRoot.Q<TextField>("password-field");

if (loginButton != null)
{
    loginButton.clicked -= OnLoginButtonClicked;
    loginButton.clicked += OnLoginButtonClicked;
}
if (registerButton != null)
{
    registerButton.clicked -= OnRegisterButtonClicked;
    registerButton.clicked += OnRegisterButtonClicked;
}
if (exitButton != null)
{
    exitButton.clicked -= OnExitButtonClicked;
    exitButton.clicked += OnExitButtonClicked;
}

isTransitioning = false;
}

/// <summary>
/// Vuelve al menú principal desde el panel de login.
/// </summary>
public void ReturnToMainMenu()
{
    if (loginPanelObject != null)
        loginPanelObject.SetActive(false);

    if (mainMenu != null)
        mainMenu.style.display = DisplayStyle.Flex;
```

```
if (jumpAction != null && jumpAction.action != null)
    jumpAction.action.Enable();
}

/// <summary>
/// Callback del botón de login: inicia la corrutina de login.
/// </summary>
private void OnLoginButtonClicked()
{
    string email = usernameField != null ? usernameField.value : "";
    string password = passwordField != null ? passwordField.value :
    "";
    StartCoroutine(LoginCoroutine(email, password));
}

/// <summary>
/// Corrutina para enviar la petición de login y procesar la respuesta.
/// </summary>
private IEnumerator LoginCoroutine(string email, string password)
{
    LoginData loginData = new LoginData { email = email, password =
password };
    string jsonBody = JsonUtility.ToJson(loginData);

        using (UnityWebRequest request = new
UnityWebRequest(Rutas.Instance.LoginUrl, "POST"))
    {
        byte[] bodyRaw =
System.Text.Encoding.UTF8.GetBytes(jsonBody);
        request.uploadHandler = new UploadHandlerRaw(bodyRaw);
        request.downloadHandler = new DownloadHandlerBuffer();
        request.SetRequestHeader("Content-Type",
"application/json");

        yield return request.SendWebRequest();

        if (request.result == UnityWebRequest.Result.Success ||
request.responseCode == 200)
    {
```

LETHAL RUN



Izan De La Cruz

David Salvador

Marc Rojano

Brian Orozco

```
TokenResponse response =
JsonUtility.FromJson<TokenResponse>(request.downloadHandler.text);
if (!string.IsNullOrEmpty(response.token) && response.user
!= null)
{
    PlayerPrefs.SetString("user.token", response.token);
    PlayerPrefs.SetInt("user.id", response.user.id);
    PlayerPrefs.SetString("user.email", response.user.email);
    PlayerPrefs.SetString("user.username",
response.user.username);
    PlayerPrefs.Save();

    SceneManager.LoadScene(nextScene);
}
}
}
}

/// <summary>
/// Datos del login para el envío al servidor.
/// </summary>
[System.Serializable]
public class LoginData
{
    public string email;
    public string password;
}

/// <summary>
/// Respuesta del servidor tras el login.
/// </summary>
[System.Serializable]
public class TokenResponse
{
    public string message;
    public string token;
    public UserData user;
}

/// <summary>
```

```
/// Datos del usuario autenticado.  
/// </summary>  
[System.Serializable]  
public class UserData  
{  
    public int id;  
    public string email;  
    public string username;  
}  
  
/// <summary>  
/// Callback del botón de registro: abre la URL de registro.  
/// </summary>  
private void OnRegisterButtonClicked()  
{  
    if (!string.IsNullOrEmpty(registerUrl))  
    {  
        Application.OpenURL(registerUrl);  
    }  
}  
  
/// <summary>  
/// Callback del botón de salir: cierra la aplicación.  
/// </summary>  
private void OnExitButtonClicked()  
{  
    Application.Quit();  
  
#if UNITY_EDITOR  
    UnityEditor.EditorApplication.isPlaying = false;  
#endif  
}
```

UISALA:

Aquest script gestiona la interfície d'usuari d'una sala multijugador a Unity, integrant Steamworks i Mirror per crear i unir-se a vestíbuls. Permet mostrar la llista de servidors disponibles, gestionar la creació i unió a partides, i coordina la transició entre escenes. A més, administra la interacció amb els botons del menú principal i l'actualització dinàmica de la llista de servidors.

```
using UnityEngine;
using UnityEngine.UIElements;
using Mirror;
using Steamworks;
using System.Collections;
using System.Collections.Generic;
using UnityEngine.SceneManagement;

/// <summary>
/// Gestiona la interfaz de usuario para la sala multijugador, integrando
/// Steamworks y Mirror.
/// Permite crear y unirse a lobbies, mostrar la lista de servidores y manejar la
/// transición entre escenas.
/// </summary>
public class UISALA : MonoBehaviour
{
    /// <summary>
    /// Documento UI Toolkit del menú principal.
    /// </summary>
    [SerializeField] UIDocument mainMenuDocument;

    /// <summary>
    /// Documento UI Toolkit de la lista de servidores.
    /// </summary>
    [SerializeField] UIDocument serverListDocument;

    /// <summary>
    /// Indica si se debe usar Steam para el multijugador.
    /// </summary>
    [SerializeField] bool useSteam = true;

    /// <summary>
    /// Prefab del NetworkManager a instanciar si es necesario.
    /// </summary>
```

LETHAL RUN



Izan De La Cruz

David Salvador

Marc Rojano

Brian Orozco

```
/// </summary>
public GameObject networkManagerPrefab;

private Button startHostBtn;
private Button joinGameBtn;
private VisualElement serverListContainer;
private List<CSteamID> lobbyIDs = new List<CSteamID>();
private HashSet<CSteamID> displayedLobbies = new HashSet<CSteamID>();

// Callbacks de Steam
protected Callback<LobbyCreated_t> lobbyCreated;
protected Callback<GameLobbyJoinRequested_t>
gameLobbyJoinRequested;
protected Callback<LobbyEnter_t> lobbyEntered;
protected Callback<LobbyMatchList_t> lobbyMatchList;
protected Callback<LobbyDataUpdate_t> lobbyDataUpdated;

private bool callbacksInitialized = false;

/// <summary>
/// Suscribe el método de carga de escena al habilitar el objeto.
/// </summary>
void OnEnable()
{
    SceneManager.sceneLoaded += OnSceneLoaded;
}

/// <summary>
/// Quita la suscripción del método de carga de escena al deshabilitar el
objeto.
/// </summary>
void OnDisable()
{
    SceneManager.sceneLoaded -= OnSceneLoaded;
}

/// <summary>
/// Inicializa la interfaz y los callbacks de Steam al iniciar.
/// </summary>
void Start()
{
    InicializarUI();

    if (useSteam && !SteamManager.Initialized)
```

LETHAL RUN



Izan De La Cruz

David Salvador

Marc Rojano

Brian Orozco

```
{  
    useSteam = false;  
    return;  
}  
  
if (useSteam)  
    InitializeSteamCallbacks();  
  
CheckIfReturningFromGame();  
}  
  
/// <summary>  
/// Verifica si se está regresando al menú principal desde otra escena.  
/// </summary>  
private void CheckIfReturningFromGame()  
{  
    if (PlayerPrefs.GetInt("ReturningToMainMenu", 0) == 1)  
    {  
        PlayerPrefs.SetInt("ReturningToMainMenu", 0);  
        PlayerPrefs.Save();  
        StartCoroutine(ResetUIAfterDelay());  
    }  
}  
  
/// <summary>  
/// Corrutina para reinicializar la UI después de un pequeño retraso.  
/// </summary>  
private IEnumerator ResetUIAfterDelay()  
{  
    yield return new WaitForSeconds(0.2f);  
    InicializarUI();  
    if (useSteam && SteamManager.Initialized)  
        RefreshLobbies();  
}  
  
/// <summary>  
/// Inicializa los callbacks de Steam para eventos de lobby.  
/// </summary>  
private void InitializeSteamCallbacks()  
{  
    if (callbacksInitialized) return;  
    if (!SteamManager.Initialized)  
    {  
        useSteam = false;  
    }  
}
```

```
        return;
    }
    lobbyCreated = Callback<LobbyCreated_t>.Create(OnLobbyCreated);
    gameLobbyJoinRequested =
Callback<GameLobbyJoinRequested_t>.Create(OnGameLobbyJoinRequested)
;
    lobbyEntered = Callback<LobbyEnter_t>.Create(OnLobbyEntered);
    lobbyMatchList =
Callback<LobbyMatchList_t>.Create(OnLobbyMatchList);
    lobbyDataUpdated =
Callback<LobbyDataUpdate_t>.Create(OnLobbyDataUpdated);
    callbacksInitialized = true;
}

/// <summary>
/// Inicializa la UI cuando se carga la escena correspondiente.
/// </summary>
/// <param name="scene">Escena cargada</param>
/// <param name="mode">Modo de carga</param>
private void OnSceneLoaded(Scene scene, LoadSceneMode mode)
{
    if (scene.name == "OFFLINE MAPA 1")
    {
        InicializarUI();
        CheckIfReturningFromGame();
    }
}

/// <summary>
/// Configura los botones y contenedores de la UI.
/// </summary>
private void InicializarUI()
{
    if (mainMenuDocument == null || serverListDocument == null)
        return;

    var mainRoot = mainMenuDocument.rootVisualElement;
    var serverRoot = serverListDocument.rootVisualElement;

    startHostBtn = mainRoot.Q<Button>("start-host-button");
    joinGameBtn = mainRoot.Q<Button>("join-host-button");

    startHostBtn.clicked -= StartLobby;
    joinGameBtn.clicked -= ShowServerList;
```

```
startHostBtn.clicked += StartLobby;
joinGameBtn.clicked += ShowServerList;

serverListContainer = serverRoot.Q<ScrollView>("server-list-scroll");
if (serverListContainer != null)
    serverListContainer.Clear();

serverRoot.Q<Button>("back-button").clicked -= HideServerList;
serverRoot.Q<Button>("refresh-button").clicked -= RefreshLobbies;
serverRoot.Q<Button>("back-button").clicked += HideServerList;
serverRoot.Q<Button>("refresh-button").clicked += RefreshLobbies;

serverListDocument.rootVisualElement.visible = false;
mainMenuDocument.rootVisualElement.visible = true;

displayedLobbies.Clear();

if (useSteam && SteamManager.Initialized)
    RefreshLobbies();
}

/// <summary>
/// Muestra la lista de servidores y actualiza los lobbies.
/// </summary>
private void ShowServerList()
{
    mainMenuDocument.rootVisualElement.visible = false;
    serverListDocument.rootVisualElement.visible = true;
    if (useSteam && SteamManager.Initialized)
        RefreshLobbies();
}

/// <summary>
/// Oculta la lista de servidores y muestra el menú principal.
/// </summary>
private void HideServerList()
{
    mainMenuDocument.rootVisualElement.visible = true;
    serverListDocument.rootVisualElement.visible = false;
}

/// <summary>
/// Refresca la lista de lobbies disponibles en Steam.
```

LETHAL RUN



Izan De La Cruz
David Salvador
Marc Rojano
Brian Orozco

```
/// </summary>
private void RefreshLobbies()
{
    if (serverListContainer != null)
        serverListContainer.Clear();

    displayedLobbies.Clear();

    if (!useSteam)
        return;

    if (!SteamManager.Initialized)
    {
        useSteam = false;
        return;
    }

    SteamMatchmaking.AddRequestLobbyListDistanceFilter(ELobbyDistanceFilter
.k_ELobbyDistanceFilterWorldwide);
    SteamMatchmaking.AddRequestLobbyListStringFilter("game", "mijuego",
ELobbyComparison.k_ELobbyComparisonEqual);
    SteamMatchmaking.RequestLobbyList();
}

/// <summary>
/// Callback cuando Steam devuelve la lista de lobbies.
/// </summary>
/// <param name="callback">Datos del evento LobbyMatchList_t</param>
private void OnLobbyMatchList(LobbyMatchList_t callback)
{
    if (!useSteam || !SteamManager.Initialized)
        return;

    lobbyIDs.Clear();
    if (serverListContainer != null)
        serverListContainer.Clear();

    displayedLobbies.Clear();

    for (int i = 0; i < callback.m_nLobbiesMatching; i++)
    {
        CSteamID lobbyID = SteamMatchmaking.GetLobbyByIndex(i);
        lobbyIDs.Add(lobbyID);
```

LETHAL RUN



Izan De La Cruz
David Salvador
Marc Rojano
Brian Orozco

```
        SteamMatchmaking.RequestLobbyData(lobbyID);
    }
}

/// <summary>
/// Callback cuando se actualizan los datos de un lobby.
/// </summary>
/// <param name="callback">Datos del evento LobbyDataUpdate_t</param>
private void OnLobbyDataUpdated(LobbyDataUpdate_t callback)
{
    if (!useSteam || !SteamManager.Initialized)
        return;
    StartCoroutine(UpdateLobbyUIWithDelay(new
CSteamID(callback.m_ulSteamIDLobby)));
}

/// <summary>
/// Corrutina para actualizar la UI con los datos del lobby tras un pequeño
retraso.
/// </summary>
/// <param name="lobbyID">ID del lobby</param>
IEnumerator UpdateLobbyUIWithDelay(CSteamID lobbyID)
{
    yield return new WaitForSeconds(0.3f);

    if (!useSteam || !SteamManager.Initialized)
        yield break;

    // FILTRO: No duplicados
    if (displayedLobbies.Contains(lobbyID))
        yield break;

    string hostAddress = SteamMatchmaking.GetLobbyData(lobbyID,
"HostAddress");
    string hostSteamID = SteamMatchmaking.GetLobbyData(lobbyID,
"HostSteamID");
    string lobbyName = SteamMatchmaking.GetLobbyData(lobbyID, "name");
    string gameTag = SteamMatchmaking.GetLobbyData(lobbyID, "game");
    string sceneName = SteamMatchmaking.GetLobbyData(lobbyID, "scene");

    // FILTRO: Solo lobbies de este juego
    if (gameTag != "mijuego") yield break;

    // FILTRO: SOLO mostrar lobbies en la escena "Lobby"
```

```
if (string.IsNullOrEmpty(sceneName) || sceneName != "Lobby")
    yield break;

// FILTRO: Lobby válido
if (string.IsNullOrEmpty(hostAddress) ||
string.IsNullOrEmpty(hostSteamID))
    yield break;

// FILTRO: No mostrar tu propio lobby SOLO si el hostSteamID está
definido
if (!string.IsNullOrEmpty(hostSteamID) && hostSteamID ==
SteamUser.GetSteamID().ToString())
    yield break;

// FILTRO: No mostrar lobbies vacíos
int playerCount = SteamMatchmaking.GetNumLobbyMembers(lobbyID);
if (playerCount <= 0)
    yield break;

displayedLobbies.Add(lobbyID);

CSteamID hostID = new CSteamID(ulong.Parse(hostSteamID));
string displayName = SteamFriends.GetFriendPersonaName(hostID);
int maxPlayers = 4; // Valor por defecto
if (NetworkManager.singleton != null)
    maxPlayers = NetworkManager.singleton.maxConnections;

var serverEntry = new Button();
serverEntry.AddToClassList("server-item");
serverEntry.style.flexDirection = FlexDirection.Row;
serverEntry.Add(new Label(lobbyName) { style = { width = 400 } });
serverEntry.Add(new Label($"{playerCount}/{maxPlayers}") { style = {
width = 200 } });
serverEntry.Add(new Label(displayName) { style = { width = 300 } });

serverEntry.clicked += () => ConnectToLobby(lobbyID);
serverListContainer.Add(serverEntry);
}

/// <summary>
/// Conecta al lobby seleccionado.
/// </summary>
/// <param name="lobbyID">ID del lobby</param>
private void ConnectToLobby(CSteamID lobbyID)
```

LETHAL RUN



Izan De La Cruz
David Salvador
Marc Rojano
Brian Orozco

```
{  
    // Instanciar NetworkManager si no existe  
    if (NetworkManager.singleton == null)  
        Instantiate(networkManagerPrefab);  
  
    if (useSteam && SteamManager.Initialized)  
        SteamMatchmaking.JoinLobby(lobbyID);  
    else  
        NetworkManager.singleton.StartClient();  
}  
  
/// <summary>  
/// Inicia la creación de un lobby como host.  
/// </summary>  
private void StartLobby()  
{  
    if (useSteam)  
    {  
        if (!SteamManager.Initialized)  
        {  
            useSteam = false;  
            return;  
        }  
        // Instanciar NetworkManager si no existe  
        if (NetworkManager.singleton == null)  
            Instantiate(networkManagerPrefab);  
  
        SteamMatchmaking.CreateLobby(  
            ELobbyType.k_ELobbyTypePublic,  
            NetworkManager.singleton.maxConnections  
        );  
        return;  
    }  
    // Instanciar NetworkManager si no existe  
    if (NetworkManager.singleton == null)  
        Instantiate(networkManagerPrefab);  
  
    NetworkManager.singleton.StartHost();  
}  
  
/// <summary>  
/// Callback cuando se crea un lobby.  
/// </summary>  
/// <param name="callback">Datos del evento LobbyCreated_t</param>
```

```

private void OnLobbyCreated(LobbyCreated_t callback)
{
    if (!useSteam || !SteamManager.Initialized)
        return;

    if (callback.m_eResult != EResult.k_EResultOK)
    {
        return;
    }

    // Guarda el lobby ID antes de arrancar el host
    LobbyData.currentLobbyID = new CSteamID(callback.m_ulSteamIDLobby);

    NetworkManager.singleton.StartHost();

    // Pone los datos básicos del lobby en Steam
    CSteamID lobbyID = LobbyData.currentLobbyID;
    SteamMatchmaking.SetLobbyData(lobbyID, "HostAddress",
        SteamUser.GetSteamID().ToString());
    SteamMatchmaking.SetLobbyData(lobbyID, "HostSteamID",
        SteamUser.GetSteamID().ToString());
    SteamMatchmaking.SetLobbyData(lobbyID, "name",
        SteamFriends.GetPersonaName() + "'s Lobby");
    SteamMatchmaking.SetLobbyData(lobbyID, "game", "mijuego");
    // El dato "scene" se actualizará automáticamente en
    OnServerSceneChanged del NetworkManager
}

/// <summary>
/// Callback cuando se recibe una invitación para unirse a un lobby.
/// </summary>
/// <param name="callback">Datos del evento
GameLobbyJoinRequested_t</param>
private void OnGameLobbyJoinRequested(GameLobbyJoinRequested_t
callback)
{
    if (useSteam && SteamManager.Initialized)
        SteamMatchmaking.JoinLobby(callback.m_steamIDLobby);
}

/// <summary>
/// Callback cuando se entra a un lobby.
/// </summary>
/// <param name="callback">Datos del evento LobbyEnter_t</param>

```

```
private void OnLobbyEntered(LobbyEnter_t callback)
{
    if (!useSteam || !SteamManager.Initialized)
        return;

    if (NetworkServer.active) return;

    CSteamID lobbyID = new CSteamID(callback.m_ulSteamIDLobby);

    // Guarda el lobbyID globalmente
    LobbyData.currentLobbyID = lobbyID;
    string hostAddress = SteamMatchmaking.GetLobbyData(lobbyID,
    "HostAddress");
    NetworkManager.singleton.networkAddress = hostAddress;
    NetworkManager.singleton.StartClient();
}

/// <summary>
/// Limpia las suscripciones y callbacks al destruir el objeto.
/// </summary>
void OnDestroy()
{
    if (startHostBtn != null)
        startHostBtn.clicked -= StartLobby;
    if (joinGameBtn != null)
        joinGameBtn.clicked -= ShowServerList;

    lobbyCreated = null;
    gameLobbyJoinRequested = null;
    lobbyEntered = null;
    lobbyMatchList = null;
    lobbyDataUpdated = null;
    callbacksInitialized = false;
}
}
```

OptionsMenuControllerSala:

Aquest script controla el menú d'opcions. Permet ajustar el volum, la qualitat gràfica i el mode de pantalla completa, desant les preferències de l'usuari. A més, gestiona la navegació entre el menú principal, el menú d'opcions i la

transició a la pantalla de títol. Facilita la interacció del jugador amb els ajustos d'àudio i gràfics de manera intuïtiva.

```
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.Audio;
using UnityEngine.SceneManagement;
using UnityEngine.UIElements;

/// <summary>
/// Controla el menú de opciones y el menú principal de la sala multijugador.
/// Permite modificar el volumen, la calidad gráfica, el modo de pantalla y
/// navegar entre menús.
/// </summary>
public class OptionsMenuControllerSala : MonoBehaviour
{
    /// <summary>
    /// Documento UI Toolkit del menú principal.
    /// </summary>
    [Header("UI Documents")]
    [SerializeField] private UIDocument mainMenuUIDocument;

    /// <summary>
    /// Documento UI Toolkit del menú de opciones.
    /// </summary>
    [SerializeField] private UIDocument optionsUIDocument;

    /// <summary>
    /// Mezclador de audio principal del juego.
    /// </summary>
    [Header("Audio")]
    [SerializeField] private AudioMixer audioMixer;

    // Elementos del menú principal
    private VisualElement mainMenuContainer;
    private Button optionsButton;
    private Button exitButton;

    // Elementos del menú de opciones
    private VisualElement optionsOverlay;
    private Button resumeButton;
    private Button applyButton;
    private Button exitLobbyButton;
```

```
private Slider volumeSlider;
private DropdownField graphicsDropdown;
private Toggle fullscreenToggle;

private const float DEFAULT_VOLUME = 0.7f;

/// <summary>
/// Aplica el volumen guardado al AudioMixer al cargar la escena.
/// </summary>
void Awake()
{
    if (audioMixer != null)
        AudioUtils.ApplySavedVolumeToMixer(audioMixer);
}

/// <summary>
/// Inicializa los elementos de la UI y sus eventos al activar el objeto.
/// </summary>
private void OnEnable()
{
    if (audioMixer != null)
        AudioUtils.ApplySavedVolumeToMixer(audioMixer);

    // Inicializa el menú principal
    if (mainMenuUIDocument != null)
    {
        var mainRoot = mainMenuUIDocument.rootVisualElement;
        mainMenuContainer =
mainRoot.Q<VisualElement>("main-menu-container");
        optionsButton = mainRoot.Q<Button>("options-button");
        exitButton = mainRoot.Q<Button>("exit-button");

        if (optionsButton != null)
            optionsButton.clicked += ShowOptionsMenu;

        if (exitButton != null)
            exitButton.clicked += GoToTitleScreen;
    }

    // Inicializa el menú de opciones
    if (optionsUIDocument != null)
    {
        var optionsRoot = optionsUIDocument.rootVisualElement;
        optionsOverlay = optionsRoot.Q<VisualElement>("options-overlay");
```

```
resumeButton = optionsRoot.Q<Button>("resume-button");
applyButton = optionsRoot.Q<Button>("apply-button");
exitLobbyButton = optionsRoot.Q<Button>("exit-lobby-button");
volumeSlider = optionsRoot.Q<Slider>("volume-slider");
graphicsDropdown =
optionsRoot.Q<DropdownField>("graphics-dropdown");
fullscreenToggle = optionsRoot.Q<Toggle>("fullscreen-toggle");

if (optionsOverlay != null) optionsOverlay.style.display =
DisplayStyle.None;
if (mainMenuContainer != null) mainMenuContainer.style.display =
DisplayStyle.Flex;

if (resumeButton != null)
    resumeButton.clicked += HideOptionsMenu;
if (exitLobbyButton != null)
    exitLobbyButton.clicked += HideOptionsMenu;
if (applyButton != null)
    applyButton.clicked += ApplySettings;

// Configura el slider de volumen
if (volumeSlider != null)
{
    volumeSlider.lowValue = 0f;
    volumeSlider.highValue = 1f;
    volumeSlider.value = PlayerPrefs.GetFloat("MasterVolume",
DEFAULT_VOLUME);
    SetVolume(volumeSlider.value);
    volumeSlider.RegisterValueChangedCallback(evt =>
SetVolume(evt.newValue));
}

// Configura la calidad gráfica
if (graphicsDropdown != null)
{
    graphicsDropdown.choices = new List<string> { "LOW", "MEDIUM",
"HIGH", "ULTRA" };
    int qualityLevel = PlayerPrefs.GetInt("QualityLevel",
QualitySettings.GetQualityLevel());
    graphicsDropdown.index = Mathf.Clamp(qualityLevel, 0,
graphicsDropdown.choices.Count - 1);
}

// Configura el modo de pantalla completa
```

LETHAL RUN



Izan De La Cruz

David Salvador

Marc Rojano

Brian Orozco

```
        if (fullscreenToggle != null)
        {
            fullscreenToggle.value = PlayerPrefs.GetInt("Fullscreen",
Screen.fullScreen ? 1 : 0) == 1;
        }
    }

/// <summary>
/// Muestra el menú de opciones y oculta el menú principal.
/// </summary>
private void ShowOptionsMenu()
{
    if (mainMenuContainer != null) mainMenuContainer.style.display =
DisplayStyle.None;
    if (optionsOverlay != null) optionsOverlay.style.display = DisplayStyle.Flex;
}

/// <summary>
/// Oculta el menú de opciones y muestra el menú principal.
/// </summary>
private void HideOptionsMenu()
{
    if (optionsOverlay != null) optionsOverlay.style.display =
DisplayStyle.None;
    if (mainMenuContainer != null) mainMenuContainer.style.display =
DisplayStyle.Flex;
}

/// <summary>
/// Establece el volumen general y lo guarda en PlayerPrefs.
/// </summary>
/// <param name="value">Nuevo valor de volumen (0 a 1).</param>
private void SetVolume(float value)
{
    float volume = Mathf.Clamp(value, 0f, 1f);
    PlayerPrefs.SetFloat("MasterVolume", volume);
    PlayerPrefs.Save();

    if (audioMixer != null)
    {
        if (volume <= 0.0001f)
            audioMixer.SetFloat("MasterVolume", -80f);
        else
```

```

        audioMixer.SetFloat("MasterVolume", Mathf.Log10(volume) * 20f);
    }
}

/// <summary>
/// Aplica y guarda la configuración de calidad gráfica y pantalla completa.
/// </summary>
private void ApplySettings()
{
    // Calidad gráfica
    if (graphicsDropdown != null)
    {
        int qualityLevel = graphicsDropdown.index;
        PlayerPrefs.SetInt("QualityLevel", qualityLevel);
        QualitySettings.SetQualityLevel(qualityLevel, true);
    }

    // Pantalla completa
    if (fullscreenToggle != null)
    {
        bool isFullscreen = fullscreenToggle.value;
        Screen.fullScreen = isFullscreen;
        PlayerPrefs.SetInt("Fullscreen", isFullscreen ? 1 : 0);
    }

    PlayerPrefs.Save();
}

/// <summary>
/// Cambia a la escena de la pantalla de título.
/// </summary>
private void GoToTitleScreen()
{
    SceneManager.LoadScene("Title Screen");
}
}

```

ControlesScriptUI:

Aquest script controla la navegació entre el menú principal i el panell de controls en una interfície feta amb UI Toolkit a Unity. Permet mostrar o ocultar

el panell de controls quan l'usuari prem els botons corresponents. Gestiona les referències als botons i assegura que els esdeveniments estiguin correctament subscrits i alliberats.

```
using UnityEngine;
using UnityEngine.UIElements;

/// <summary>
/// Controla la navegación entre el menú principal y el panel de controles
/// usando UI Toolkit.
/// </summary>
public class controles : MonoBehaviour
{
    /// <summary>
    /// Referencia al documento UI del menú principal.
    /// </summary>
    [Header("UIDocuments")]
    [SerializeField] private UIDocument mainMenuUIDocument;

    /// <summary>
    /// Referencia al documento UI del panel de controles.
    /// </summary>
    [SerializeField] private UIDocument controlsUIDocument;

    private VisualElement mainMenuRoot;
    private VisualElement controlsRoot;

    private Button controlsButton; // Botón para abrir el panel de controles
    private Button backButton; // Botón para volver al menú principal

    /// <summary>
    /// Inicializa las referencias de UI y configura los eventos de los botones al
    /// habilitar el objeto.
    /// </summary>
    private void OnEnable()
    {
        if (mainMenuUIDocument == null || controlsUIDocument == null)
        {
            return;
        }

        // Obtiene el elemento raíz de cada panel
        mainMenuRoot = mainMenuUIDocument.rootVisualElement;
        controlsRoot = controlsUIDocument.rootVisualElement;
    }
}
```

```
// Oculta el panel de controles al inicio
controlsRoot.style.display = DisplayStyle.None;

// Obtiene las referencias a los botones
controlsButton = mainMenuRoot.Q<Button>("controls-button");
backButton = controlsRoot.Q<Button>("back-button");

if (controlsButton != null)
    controlsButton.clicked += ShowControlsPanel;

if (backButton != null)
    backButton.clicked += ShowMainMenu;
}

/// <summary>
/// Muestra el panel de controles y oculta el menú principal.
/// </summary>
private void ShowControlsPanel()
{
    mainMenuRoot.style.display = DisplayStyle.None;
    controlsRoot.style.display = DisplayStyle.Flex;
}

/// <summary>
/// Muestra el menú principal y oculta el panel de controles.
/// </summary>
private void ShowMainMenu()
{
    controlsRoot.style.display = DisplayStyle.None;
    mainMenuRoot.style.display = DisplayStyle.Flex;
}

/// <summary>
/// Elimina las suscripciones a los eventos de los botones al deshabilitar el
/// objeto.
/// </summary>
private void OnDisable()
{
    if (controlsButton != null)
        controlsButton.clicked -= ShowControlsPanel;

    if (backButton != null)
        backButton.clicked -= ShowMainMenu;
```

```

    }
}
```

CanvasLobbyUIToolkit:

Aquest script gestiona la interfície d'usuari del vestíbul multijugador a Unity utilitzant UI Toolkit, Mirror i Steamworks. Permet a l'amfitrió iniciar la partida i a qualsevol jugador sortir del vestíbul, assegurant que es tanquin correctament les connexions i s'abandoni el vestíbul de Steam si és necessari. A més, gestiona la transició entre escenes en iniciar el joc o sortir del vestíbul. Facilita la interacció de l'usuari amb les opcions principals del vestíbul abans de començar la partida.

```

using UnityEngine;
using Mirror;
using UnityEngine.UIElements;
using UnityEngine.SceneManagement;
using Steamworks;

/// <summary>
/// Controla la interfaz de usuario del lobby multijugador usando UI Toolkit,
/// permitiendo iniciar la partida y salir del lobby, integrando Mirror y
/// Steamworks.
/// </summary>
public class CanvasLobbyUIToolkit : MonoBehaviour
{
    /// <summary>
    /// Documento UI Toolkit asociado al lobby.
    /// </summary>
    [SerializeField] private UIDocument uiDocument;

    private Button startGameButton;
    private Button exitLobbyButton;

    /// <summary>
    /// Inicializa los botones y sus eventos al iniciar el objeto.
    /// </summary>
    void Start()
    {
        VisualElement root = uiDocument.rootVisualElement;

        startGameButton = root.Q<Button>("start-host-button");
        exitLobbyButton = root.Q<Button>("exit-button");
    }
}
```

LETHAL RUN



Izan De La Cruz

David Salvador

Marc Rojano

Brian Orozco

```
startGameButton.clicked -= StartGame;
exitLobbyButton.clicked -= ExitLobby;

if (NetworkServer.active)
{
    startGameButton.style.display = DisplayStyle.Flex;
    startGameButton.clicked += StartGame;
}
else
{
    startGameButton.style.display = DisplayStyle.None;
}

exitLobbyButton.clicked += ExitLobby;
}

/// <summary>
/// Sale del lobby, detiene la conexión y vuelve a la escena offline.
/// </summary>
private void ExitLobby()
{
    // Salir del lobby de Steam si existe
    if (SteamManager.Initialized && LobbyData.currentLobbyID.IsValid())
    {
        SteamMatchmaking.LeaveLobby(LobbyData.currentLobbyID);
        LobbyData.currentLobbyID = CSteamID.Nil;
    }

    // Detener la conexión de Mirror
    if (NetworkServer.active)
    {
        NetworkManager.singleton.StopHost();
    }
    else if (NetworkClient.isConnected)
    {
        NetworkManager.singleton.StopClient();
    }

    // Cambiar a la escena offline
    SceneManager.LoadScene("OFFLINE MAPA 1");

    // Destruir el NetworkManager persistente
    if (NetworkManager.singleton != null)
```

```
        Destroy(NetworkManager.singleton.gameObject);
    }

/// <summary>
/// Inicia la partida cambiando a la escena seleccionada.
/// </summary>
private void StartGame()
{
    string sceneToLoad = "Mapa1";
    if (SelectedMap.Instance != null &&
!string.IsNullOrEmpty(SelectedMap.Instance.SelectedMapSceneName))
    {
        sceneToLoad = SelectedMap.Instance.SelectedMapSceneName;
    }
    NetworkManager.singleton.ServerChangeScene(sceneToLoad);
}
}
```

MapSelector:

Aquest script permet a l'amfitrió d'una partida multijugador seleccionar el mapa abans de començar el joc, mostrant una interfície visual amb detalls i una previsualització de cada mapa. Sincronitza la selecció entre tots els clients utilitzant Mirror i actualitza dinàmicament la interfície d'usuari. A més, gestiona la visibilitat del panell de selecció i tradueix el nom del mapa seleccionat al nom de l'escena corresponent. D'aquesta manera, facilita una experiència de selecció de mapa clara i centralitzada per a tots els jugadors.

```
using System;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UIElements;
using Mirror;

/// <summary>
/// Permite al host seleccionar el mapa antes de iniciar la partida multijugador.
/// Sincroniza la selección entre todos los clientes y muestra detalles visuales
/// del mapa.
/// </summary>
public class MapSelector : NetworkBehaviour
{
```

```
/// <summary>
/// Documento UI Toolkit asociado a la selección de mapas.
/// </summary>
[SerializeField] private UIDocument uiDocument;

/// <summary>
/// Tecla para mostrar/ocultar la UI de selección de mapas.
/// </summary>
[SerializeField] private KeyCode toggleUIKey = KeyCode.LeftControl;

private VisualElement root;
private VisualElement mapSelectionContainer;
private VisualElement mapDisplay;
private VisualElement mapPreviewImage;
private Label mapNameLabel;
private Label mapDescriptionLabel;
private Button map1Button, map2Button;
private Button selectButton, backButton;

/// <summary>
/// Indica si la UI de selección de mapas está activa (sincronizado en red).
/// </summary>
[SyncVar(hook = nameof(OnUIActiveChanged))]
private bool isUIActive = false;

/// <summary>
/// Nombre del mapa actualmente seleccionado (sincronizado en red).
/// </summary>
[SyncVar(hook = nameof(OnMapSelected))]
private string currentMapName = "";

/// <summary>
/// Base de datos local de mapas disponibles.
/// </summary>
private Dictionary<string, MapData> mapDatabase = new Dictionary<string,
MapData>();

/// <summary>
/// Inicializa la base de datos de mapas al despertar el objeto.
/// </summary>
void Awake()
{
    InitializeMapDatabase();
}
```

LETHAL RUN



Izan De La Cruz

David Salvador

Marc Rojano

Brian Orozco

```
/// <summary>
/// Configura la UI y su estado inicial al conectar el cliente.
/// </summary>
public override void OnStartClient()
{
    base.OnStartClient();
    SetupUI();
    UpdateUIVisibility(isUIActive);
    if (!string.IsNullOrEmpty(currentMapName))
    {
        UpdateMapSelectionUI(currentMapName);
        DisplayMapDetails(currentMapName);
    }
}

/// <summary>
/// Permite al host mostrar u ocultar la UI de selección de mapas con una
tecla.
/// </summary>
void Update()
{
    if (!isServer) return;

    if (Input.GetKeyDown(toggleUIKey))
    {
        isUIActive = !isUIActive;
    }
}

/// <summary>
/// Configura los elementos y eventos de la UI.
/// </summary>
void SetupUI()
{
    root = uiDocument.rootVisualElement;
    mapSelectionContainer =
root.Q<VisualElement>("map-selection-container");
    mapDisplay = root.Q<VisualElement>("map-display");
    mapPreviewImage = root.Q<VisualElement>("map-preview");
    mapNameLabel = root.Q<Label>("map-name");
    mapDescriptionLabel = root.Q<Label>("map-description");

    map1Button = root.Q<Button>("map-1");
```

```
map2Button = root.Q<Button>("map-2");

selectButton = root.Q<Button>("select-button");
backButton = root.Q<Button>("back-button");

if (isServer)
{
    map1Button.clicked += () => CmdSelectMap("Mario World");
    map2Button.clicked += () => CmdSelectMap("Parkour Extremo");

    if (selectButton != null)
        selectButton.clicked += OnSelectButtonClicked;
    if (backButton != null)
        backButton.clicked += OnBackButtonClicked;
}
else
{
    map1Button.SetEnabled(false);
    map2Button.SetEnabled(false);
}

/// <summary>
/// Muestra u oculta el panel de selección de mapas.
/// </summary>
/// <param name="visible">Si es true, muestra la UI; si es false, la
oculta.</param>
void UpdateUIVisibility(bool visible)
{
    if (mapSelectionContainer != null)
        mapSelectionContainer.style.display = visible ? DisplayStyle.Flex :
DisplayStyle.None;
}

/// <summary>
/// Hook llamado cuando cambia el estado de visibilidad de la UI.
/// </summary>
/// <param name="oldVal">Valor anterior.</param>
/// <param name="newVal">Nuevo valor.</param>
void OnUIActiveChanged(bool oldVal, bool newVal)
{
    UpdateUIVisibility(newVal);
    if (!newVal)
        HideMapDetails();
```

```
}

/// <summary>
/// Comando para seleccionar un mapa desde el host.
/// </summary>
/// <param name="mapName">Nombre del mapa a seleccionar.</param>
[Command(requiresAuthority = false)]
void CmdSelectMap(string mapName)
{
    if (mapDatabase.ContainsKey(mapName))
    {
        currentMapName = mapName;
    }
}

/// <summary>
/// Hook llamado cuando cambia el mapa seleccionado.
/// </summary>
/// <param name="oldMap">Mapa anterior.</param>
/// <param name="newMap">Nuevo mapa seleccionado.</param>
void OnMapSelected(string oldMap, string newMap)
{
    if (!string.IsNullOrEmpty(newMap))
    {
        UpdateMapSelectionUI(newMap);
        DisplayMapDetails(newMap);

        // Traducción de nombre de mapa a nombre de escena
        string sceneName = MapNameToSceneName(newMap);
        if (SelectedMap.Instance != null)
            SelectedMap.Instance.SelectedMapSceneName = sceneName;
    }
    else
    {
        HideMapDetails();
        if (SelectedMap.Instance != null)
            SelectedMap.Instance.SelectedMapSceneName = "Mapa 1";
    }
}

/// <summary>
/// Traduce el nombre del mapa al nombre de la escena.
/// </summary>
/// <param name="mapName">Nombre del mapa.</param>
```

```
/// <returns>Nombre de la escena correspondiente.</returns>
string MapNameToSceneName(string mapName)
{
    switch (mapName)
    {
        case "Mario World": return "Mapa1";
        case "Parkour Extremo": return "Mapa 2";
        default: return "Mapa 1";
    }
}

/// <summary>
/// Actualiza visualmente la selección del mapa en la UI.
/// </summary>
/// <param name="mapName">Nombre del mapa seleccionado.</param>
void UpdateMapSelectionUI(string mapName)
{
    // Aquí puedes actualizar la selección visual de botones si lo deseas
}

/// <summary>
/// Muestra los detalles del mapa seleccionado, incluyendo nombre,
descripción e imagen.
/// </summary>
/// <param name="mapName">Nombre del mapa.</param>
void DisplayMapDetails(string mapName)
{
    if (mapDatabase.TryGetValue(mapName, out MapData data))
    {
        mapDisplay.style.display = DisplayStyle.Flex;
        mapNameLabel.text = data.Name;
        mapDescriptionLabel.text = data.Description;
        if (mapPreviewImage != null && data.PreviewTexture != null)
        {
            mapPreviewImage.style.backgroundImage = new
StyleBackground(data.PreviewTexture);
        }
    }
}

/// <summary>
/// Oculta los detalles del mapa.
/// </summary>
void HideMapDetails()
```

LETHAL RUN



Izan De La Cruz

David Salvador

Marc Rojano

Brian Orozco

```
{  
    if (mapDisplay != null)  
        mapDisplay.style.display = DisplayStyle.None;  
}  
  
/// <summary>  
/// Acción al pulsar el botón de seleccionar mapa.  
/// </summary>  
void OnSelectButtonClicked()  
{  
    if (!isServer) return;  
    isUIActive = false;  
    // Aquí puedes iniciar el juego o cambiar de escena.  
}  
  
/// <summary>  
/// Acción al pulsar el botón de volver atrás.  
/// </summary>  
void OnBackButtonClicked()  
{  
    if (!isServer) return;  
    isUIActive = false;  
}  
  
/// <summary>  
/// Inicializa la base de datos de mapas disponibles.  
/// </summary>  
void InitializeMapDatabase()  
{  
    mapDatabase = new Dictionary<string, MapData>()  
    {  
        { "Mario World", new MapData(  
            "Mario World",  
            "Mapa inspirado en el mundo icónico de Mario",  
            Resources.Load<Texture2D>("MarioWorld")  
        )),  
        { "Parkour Extremo", new MapData(  
            "Parkour Extremo",  
            "Parkour agotador (Nivel DIOS)",  
            Resources.Load<Texture2D>("ParkourExtremo")  
        )},  
    };  
}
```

LETHAL RUN



Izan De La Cruz

David Salvador

Marc Rojano

Brian Orozco

```
/// <summary>
/// Clase que almacena los datos de cada mapa.
/// </summary>
public class MapData
{
    public string Name;
    public string Description;
    public Texture2D PreviewTexture;

    /// <summary>
    /// Constructor para los datos del mapa.
    /// </summary>
    /// <param name="name">Nombre del mapa.</param>
    /// <param name="description">Descripción del mapa.</param>
    /// <param name="previewTexture">Textura de previsualización.</param>
    public MapData(string name, string description, Texture2D
previewTexture)
    {
        Name = name;
        Description = description;
        PreviewTexture = previewTexture;
    }
}
```

OpcionesMenu(In Game):

Aquest script controla el menú d'opcions dins de la sala multijugador de Unity, permetent als jugadors ajustar el volum, la qualitat gràfica i el mode de pantalla completa, tot desant les seves preferències. A més, gestiona la sortida del vestíbul, tancant les connexions de xarxa i tornant al menú principal. També facilita la navegació i la visibilitat del menú d'opcions durant la partida. La seva integració amb Mirror i Steamworks assegura una transició fluida entre escenes i el correcte maneig de la xarxa.

```
using UnityEngine;
using UnityEngine.UIElements;
using UnityEngine.SceneManagement;
using System.Collections.Generic;
using Mirror;
using Steamworks;
```

LETHAL RUN



Izan De La Cruz

David Salvador

Marc Rojano

Brian Orozco

```
/// <summary>
/// Controla el menú de opciones en el lobby multijugador, permitiendo ajustar
volumen, calidad gráfica, pantalla completa y salir del lobby.
/// Gestiona la persistencia de preferencias y la transición entre escenas,
integrando Mirror y Steamworks.
/// </summary>
public class OptionsMenuController : MonoBehaviour
{
    /// <summary>
    /// Instancia singleton del controlador de menú de opciones.
    /// </summary>
    public static OptionsMenuController Instance { get; private set; }

    /// <summary>
    /// Referencia al UIDocument que contiene el menú de opciones.
    /// </summary>
    [Tooltip("Referencia al UIDocument que contiene el menú de opciones")]
    [SerializeField] private UIDocument uiDocument;

    /// <summary>
    /// Nombre de la escena principal a cargar al salir del lobby.
    /// </summary>
    [Tooltip("Escena a cargar cuando se presiona Exit Lobby")]
    [SerializeField] private string mainMenuSceneName = "OFFLINE MAPA 1";

    // Referencias a los elementos de UI
    private VisualElement optionsOverlay;
    private Button resumeButton;
    private Button applyButton;
    private Button exitLobbyButton;
    private Slider volumeSlider;
    private DropdownField graphicsDropdown;
    private Toggle fullscreenToggle;

    private AudioSource musicSource;

    // Referencia al jugador local
    private StarterAssets.ThirdPersonController _localPlayer;

    /// <summary>
    /// Inicializa el singleton y referencias de audio y UI.
    /// </summary>
    private void Awake()
    {
```

LETHAL RUN



Izan De La Cruz

David Salvador

Marc Rojano

Brian Orozco

```
// Singleton pattern
if (Instance != null && Instance != this)
{
    Destroy(this.gameObject);
    return;
}
Instance = this;

musicSource = FindObjectOfType< AudioSource>();
if (uiDocument == null)
    uiDocument = GetComponent< UIDocument>();
}

/// <summary>
/// Inicializa la UI al iniciar el objeto.
/// </summary>
private void Start()
{
    if (uiDocument != null)
        InitializeUI();
}

/// <summary>
/// Prepara los elementos del menú de opciones y sus eventos.
/// </summary>
private void InitializeUI()
{
    VisualElement root = uiDocument.rootVisualElement;
    if (root == null)
    {
        return;
    }

    optionsOverlay = root.Q< VisualElement>("options-overlay");
    if (optionsOverlay == null)
    {
        return;
    }

    resumeButton = root.Q< Button>("resume-button");
    applyButton = root.Q< Button>("apply-button");
    exitLobbyButton = root.Q< Button>("exit-lobby-button");
    volumeSlider = root.Q< Slider>("volume-slider");
    graphicsDropdown = root.Q< DropdownField>("graphics-dropdown");
```

```
fullscreenToggle = root.Q<Toggle>("fullscreen-toggle");

SetupInitialValues();
RegisterCallbacks();
HideOptionsMenu();
}

/// <summary>
/// Configura los valores iniciales de los controles del menú de opciones.
/// </summary>
private void SetupInitialValues()
{
    // Volumen
    if (volumeSlider != null)
    {
        volumeSlider.lowValue = 0f;
        volumeSlider.highValue = 1f;
        volumeSlider.value = PlayerPrefs.GetFloat("MasterVolume", 0.75f);
    }

    // Calidad gráfica
    if (graphicsDropdown != null)
    {
        graphicsDropdown.choices = new List<string> { "LOW", "MEDIUM",
    "HIGH", "ULTRA" };
        int qualityLevel = PlayerPrefs.GetInt("QualityLevel",
    QualitySettings.GetQualityLevel());
        graphicsDropdown.index = Mathf.Clamp(qualityLevel, 0,
    graphicsDropdown.choices.Count - 1);
    }

    // Pantalla completa
    if (fullscreenToggle != null)
    {
        fullscreenToggle.value = Screen.fullScreen;
    }
}

/// <summary>
/// Registra los eventos de los botones del menú de opciones.
/// </summary>
private void RegisterCallbacks()
{
    if (resumeButton != null)
```

LETHAL RUN



Izan De La Cruz

David Salvador

Marc Rojano

Brian Orozco

```
resumeButton.clicked += HideOptionsMenu;

if (applyButton != null)
    applyButton.clicked += ApplySettings;

if (exitLobbyButton != null)
    exitLobbyButton.clicked += ExitLobby;
}

/// <summary>
/// Aplica y guarda la configuración de volumen, calidad gráfica y pantalla completa.
/// </summary>
private void ApplySettings()
{
    // Volumen
    if (volumeSlider != null)
    {
        float volume = volumeSlider.value;
        PlayerPrefs.SetFloat("MasterVolume", volume);
        AudioListener.volume = volume;
        if (musicSource != null)
            musicSource.volume = volume;
    }

    // Calidad gráfica
    if (graphicsDropdown != null)
    {
        int qualityLevel = graphicsDropdown.index;
        PlayerPrefs.SetInt("QualityLevel", qualityLevel);

        switch (qualityLevel)
        {
            case 0: // LOW
                QualitySettings.SetQualityLevel(0, true);
                QualitySettings.shadows = ShadowQuality.Disable;
                QualitySettings.globalTextureMipmapLimit = 2;
                QualitySettings.particleRaycastBudget = 4;
                QualitySettings.pixelLightCount = 0;
                QualitySettings.antiAliasing = 0;
                QualitySettings.realtimeReflectionProbes = false;
                break;

            case 1: // MEDIUM
                QualitySettings.SetQualityLevel(1, true);
                QualitySettings.shadows = ShadowQuality.Enable;
                QualitySettings.globalTextureMipmapLimit = 4;
                QualitySettings.particleRaycastBudget = 8;
                QualitySettings.pixelLightCount = 4;
                QualitySettings.antiAliasing = 1;
                QualitySettings.realtimeReflectionProbes = true;
                break;

            case 2: // HIGH
                QualitySettings.SetQualityLevel(2, true);
                QualitySettings.shadows = ShadowQuality.Enable;
                QualitySettings.globalTextureMipmapLimit = 8;
                QualitySettings.particleRaycastBudget = 16;
                QualitySettings.pixelLightCount = 8;
                QualitySettings.antiAliasing = 2;
                QualitySettings.realtimeReflectionProbes = true;
                break;

            case 3: // ULTRA
                QualitySettings.SetQualityLevel(3, true);
                QualitySettings.shadows = ShadowQuality.Enable;
                QualitySettings.globalTextureMipmapLimit = 16;
                QualitySettings.particleRaycastBudget = 32;
                QualitySettings.pixelLightCount = 16;
                QualitySettings.antiAliasing = 4;
                QualitySettings.realtimeReflectionProbes = true;
                break;
        }
    }
}
```

```

QualitySettings.SetQualityLevel(1, true);
QualitySettings.shadows = ShadowQuality.HardOnly;
QualitySettings.globalTextureMipmapLimit = 1;
QualitySettings.particleRaycastBudget = 64;
QualitySettings.pixelLightCount = 1;
QualitySettings.antiAliasing = 0;
QualitySettings.realtimeReflectionProbes = false;
break;

case 2: // HIGH
QualitySettings.SetQualityLevel(2, true);
QualitySettings.shadows = ShadowQuality.All;
QualitySettings.globalTextureMipmapLimit = 0;
QualitySettings.particleRaycastBudget = 256;
QualitySettings.pixelLightCount = 2;
QualitySettings.antiAliasing = 2;
QualitySettings.realtimeReflectionProbes = true;
break;

case 3: // ULTRA
QualitySettings.SetQualityLevel(3, true);
QualitySettings.shadows = ShadowQuality.All;
QualitySettings.shadowResolution = ShadowResolution.VeryHigh;
QualitySettings.globalTextureMipmapLimit = 0;
QualitySettings.particleRaycastBudget = 4096;
QualitySettings.pixelLightCount = 4;
QualitySettings.antiAliasing = 8;
QualitySettings.realtimeReflectionProbes = true;
QualitySettings.softParticles = true;
break;
}

// Ajustar cámara principal si existe
Camera mainCamera = Camera.main;
if (mainCamera != null)
{
    mainCamera.farClipPlane = 100f + (qualityLevel * 200f);

    var postProcess = mainCamera.GetComponent<MonoBehaviour>();
    if (postProcess != null &&
postProcess.GetType().Name.Contains("PostProcess"))
    {
        postProcess.enabled = (qualityLevel >= 2);
    }
}

```

```
        }

    }

// Pantalla completa
if (fullscreenToggle != null)
{
    bool isFullscreen = fullscreenToggle.value;
    Screen.fullScreen = isFullscreen;
    PlayerPrefs.SetInt("Fullscreen", isFullscreen ? 1 : 0);
}

PlayerPrefs.Save();
}

/// <summary>
/// Sale del lobby, detiene la conexión de red y vuelve a la escena principal.
/// </summary>
private void ExitLobby()
{
    // Ocultar menú y restaurar el tiempo
    HideOptionsMenu();

    // Salir del lobby de Steam si existe
    if (SteamManager.Initialized && LobbyData.currentLobbyID.IsValid())
    {
        SteamMatchmaking.LeaveLobby(LobbyData.currentLobbyID);
        LobbyData.currentLobbyID = CSteamID.Nil;
    }

    // Detener la conexión de Mirror
    if (NetworkServer.active)
    {
        NetworkManager.singleton.StopHost();
    }
    else if (NetworkClient.isConnected)
    {
        NetworkManager.singleton.StopClient();
    }

    // Marcar que estamos volviendo al menú principal
    PlayerPrefs.SetInt("ReturningToMainMenu", 1);
    PlayerPrefs.Save();

    // Cambiar a la escena offline
```

```
SceneManager.LoadScene(mainMenuSceneName);

// Destruir el NetworkManager persistente
if (NetworkManager.singleton != null)
    Destroy(NetworkManager.singleton.gameObject);
}

/// <summary>
/// Muestra el menú de opciones.
/// </summary>
public void ShowOptionsMenu()
{
    if (optionsOverlay != null)
    {
        optionsOverlay.style.display = DisplayStyle.Flex;
    }
}

/// <summary>
/// Oculta el menú de opciones.
/// </summary>
public void HideOptionsMenu()
{
    if (optionsOverlay != null)
    {
        optionsOverlay.style.display = DisplayStyle.None;
    }
}

/// <summary>
/// Alterna la visibilidad del menú de opciones.
/// </summary>
public void ToggleOptionsMenu()
{
    if (optionsOverlay == null) return;

    if (optionsOverlay.style.display == DisplayStyle.None)
        ShowOptionsMenu();
    else
        HideOptionsMenu();
}

/// <summary>
```

```
/// Busca el jugador local y permite abrir/cerrar el menú de opciones con la tecla R.  
/// </summary>  
private void Update()  
{  
    // Buscar y cachear el jugador local si no está asignado  
    if (_localPlayer == null)  
    {  
        foreach (var player in  
FindObjectsOfType<StarterAssets.ThirdPersonController>())  
        {  
            if (player.isLocalPlayer)  
            {  
                _localPlayer = player;  
                break;  
            }  
        }  
    }  
  
    // Solo el jugador local puede abrir/cerrar el menú  
    if (_localPlayer == null || !_localPlayer.isLocalPlayer)  
        return;  
  
    if (Input.GetKeyDown(KeyCode.R))  
    {  
        ToggleOptionsMenu();  
    }  
}
```

WinCondition:

Aquest script gestiona la condició de victòria en una partida multijugador amb Mirror. Detecta quan un jugador arriba a l'objectiu, determina i sincronitza el guanyador, i mostra a la interfície si el jugador local ha guanyat o percut. A més, després d'un breu retard, coordina el canvi d'escena per a tots els jugadors cap al vestíbul. D'aquesta manera, facilita el flux final de la partida i la comunicació visual del resultat.

using UnityEngine;

LETHAL RUN



Izan De La Cruz

David Salvador

Marc Rojano

Brian Orozco

```
using Mirror;
using TMPro;
using UnityEngine.UI;
using System.Collections;

/// <summary>
/// Gestiona la condición de victoria en una partida multijugador con Mirror.
/// Detecta al jugador ganador, muestra el resultado en la UI y coordina la
/// transición de escena.
/// </summary>
public class WinCondition : NetworkBehaviour
{
    /// <summary>
    /// NetId del jugador ganador, sincronizado en red.
    /// </summary>
    [SyncVar(hook = nameof(OnWinnerChanged))]
    private uint winnerNetId = 0;

    [Header("UI Panels")]
    [Tooltip("Panel que se muestra al ganar.")]
    [SerializeField] private GameObject winPanel;

    [Tooltip("Panel que se muestra al perder.")]
    [SerializeField] private GameObject losePanel;

    [Header("UI Text")]
    [Tooltip("Texto de victoria.")]
    [SerializeField] private TextMeshProUGUI winText;

    [Tooltip("Texto de derrota.")]
    [SerializeField] private TextMeshProUGUI loseText;

    [Header("Canvas a desactivar")]
    [Tooltip("Canvas de gameplay que se desactiva al finalizar la partida.")]
    [SerializeField] private Canvas gameplayCanvas;

    private bool hasStartedLobbyCountdown = false;

    /// <summary>
    /// Inicializa el estado de los paneles de UI al iniciar.
    /// </summary>
    private void Start()
    {
        if (winPanel != null)
```

```
winPanel.SetActive(false);

if (losePanel != null)
    losePanel.SetActive(false);
}

/// <summary>
/// Detecta la colisión de un jugador con el trigger de victoria.
/// </summary>
/// <param name="other">Collider que entra en el trigger.</param>
private void OnTriggerEnter(Collider other)
{
    if (winnerNetId != 0) return;

    if (other.CompareTag("Player"))
    {
        NetworkIdentity playerIdentity =
other.GetComponent<NetworkIdentity>();
        if (playerIdentity != null)
        {
            CmdSetWinner(playerIdentity.netId);
        }
    }
}

/// <summary>
/// Comando que establece el jugador ganador en el servidor.
/// </summary>
/// <param name="playerNetId">NetId del jugador ganador.</param>
[Command(requiresAuthority = false)]
private void CmdSetWinner(uint playerNetId)
{
    if (winnerNetId == 0)
    {
        winnerNetId = playerNetId;

        // Iniciar la corutina para cambiar de escena después de 5 segundos
        // (solo en servidor)
        if (!hasStartedLobbyCountdown)
        {
            hasStartedLobbyCountdown = true;
            StartCoroutine(WaitAndLoadLobby());
        }
    }
}
```

```
}

/// <summary>
/// Espera unos segundos y luego cambia a la escena del lobby para todos
/// los jugadores.
/// </summary>
private IEnumerator WaitAndLoadLobby()
{
    yield return new WaitForSeconds(5f);

    // Cambiar la escena para todos los jugadores
    NetworkManager.singleton.ServerChangeScene("Lobby");
}

/// <summary>
/// Hook que se ejecuta cuando cambia el ganador en la red.
/// </summary>
/// <param name="oldWinnerNetId">Valor anterior del ganador.</param>
/// <param name="newWinnerNetId">Nuevo valor del ganador.</param>
private void OnWinnerChanged(uint oldWinnerNetId, uint newWinnerNetId)
{
    if (newWinnerNetId != 0)
    {
        ShowGameResult(newWinnerNetId);
    }
}

/// <summary>
/// Muestra el resultado de la partida (victoria o derrota) en la UI local.
/// </summary>
/// <param name="winningPlayerId">NetId del jugador ganador.</param>
private void ShowGameResult(uint winningPlayerId)
{
    NetworkIdentity localPlayer = NetworkClient.localPlayer;
    if (localPlayer != null)
    {
        if (gameplayCanvas != null)
            gameplayCanvas.gameObject.SetActive(false);

        bool isWinner = localPlayer.netId == winningPlayerId;

        if (isWinner)
        {
            if (winPanel != null)
```

```
{  
    winPanel.SetActive(true);  
    if (winText != null)  
        winText.text = "¡VICTORIA!";  
    }  
}  
else  
{  
    if (losePanel != null)  
    {  
        losePanel.SetActive(true);  
        if (loseText != null)  
            loseText.text = "DERROTA";  
    }  
}  
}  
}  
}
```

Personajes:

StarterAssets:

Aquest script gestiona i emmagatzema les entrades del jugador per al moviment, la càmera, el salt i l'sprint en un joc desenvolupat amb Unity. A més, controla la configuració del cursor i permet adaptar el comportament del personatge segons l'entrada rebuda. És compatible amb el nou sistema d'entrada de Unity i facilita la integració de controls tant per a teclat com per a comandament. La seva funció principal és servir de pont entre el sistema d'entrada i la lògica de moviment del personatge.

```
using UnityEngine;  
#if ENABLE_INPUT_SYSTEM && STARTER_ASSETS_PACKAGES_CHECKED  
using UnityEngine.InputSystem;  
#endif  
  
namespace StarterAssets  
{  
    /// <summary>
```

```
/// Gestiona y almacena los valores de entrada del jugador para movimiento,  
cámara, salto y sprint.  
/// Además, controla el estado del cursor y la configuración de entrada.  
/// </summary>  
public class StarterAssetsInputs : MonoBehaviour  
{  
    [Header("Character Input Values")]  
    /// <summary>  
    /// Vector de movimiento recibido del jugador.  
    /// </summary>  
    public Vector2 move;  
  
    /// <summary>  
    /// Vector de rotación de la cámara recibido del jugador.  
    /// </summary>  
    public Vector2 look;  
  
    /// <summary>  
    /// Indica si el jugador está intentando saltar.  
    /// </summary>  
    public bool jump;  
  
    /// <summary>  
    /// Indica si el jugador está intentando esprintar.  
    /// </summary>  
    public bool sprint;  
  
    [Header("Movement Settings")]  
    /// <summary>  
    /// Indica si el movimiento es analógico.  
    /// </summary>  
    public bool analogMovement;  
  
    [Header("Mouse Cursor Settings")]  
    /// <summary>  
    /// Determina si el cursor debe estar bloqueado.  
    /// </summary>  
    public bool cursorLocked = true;  
  
    /// <summary>  
    /// Determina si la entrada del cursor afecta a la rotación de la cámara.  
    /// </summary>  
    public bool cursorInputForLook = true;
```

```
#if ENABLE_INPUT_SYSTEM && STARTER_ASSETS_PACKAGES_CHECKED
    /// <summary>
    /// Callback del sistema de entrada para el movimiento.
    /// </summary>
    /// <param name="value">Valor de entrada del movimiento.</param>
    public void OnMove(InputValue value)
    {
        MoveInput(value.Get<Vector2>());
    }

    /// <summary>
    /// Callback del sistema de entrada para la cámara.
    /// </summary>
    /// <param name="value">Valor de entrada de la cámara.</param>
    public void OnLook(InputValue value)
    {
        if(cursorInputForLook)
        {
            LookInput(value.Get<Vector2>());
        }
    }

    /// <summary>
    /// Callback del sistema de entrada para el salto.
    /// </summary>
    /// <param name="value">Valor de entrada del salto.</param>
    public void OnJump(InputValue value)
    {
        JumpInput(value.isPressed);
    }

    /// <summary>
    /// Callback del sistema de entrada para el sprint.
    /// </summary>
    /// <param name="value">Valor de entrada del sprint.</param>
    public void OnSprint(InputValue value)
    {
        SprintInput(value.isPressed);
    }
#endif

    /// <summary>
    /// Asigna el vector de movimiento recibido.
    /// </summary>
```

```
/// <param name="newMoveDirection">Nuevo vector de
movimiento.</param>
public void MoveInput(Vector2 newMoveDirection)
{
    move = newMoveDirection;
}

/// <summary>
/// Asigna el vector de rotación de la cámara recibido.
/// </summary>
/// <param name="newLookDirection">Nuevo vector de cámara.</param>
public void LookInput(Vector2 newLookDirection)
{
    look = newLookDirection;
}

/// <summary>
/// Asigna el estado de salto recibido.
/// </summary>
/// <param name="newJumpState">Nuevo estado de salto.</param>
public void JumpInput(bool newJumpState)
{
    jump = newJumpState;
}

/// <summary>
/// Asigna el estado de sprint recibido.
/// </summary>
/// <param name="newSprintState">Nuevo estado de sprint.</param>
public void SprintInput(bool newSprintState)
{
    sprint = newSprintState;
}

/// <summary>
/// Ajusta el estado del cursor al cambiar el foco de la aplicación.
/// </summary>
/// <param name="hasFocus">Indica si la aplicación tiene el foco.</param>
private void OnApplicationFocus(bool hasFocus)
{
    SetCursorState(cursorLocked);
}

/// <summary>
```

```
/// Establece el estado de bloqueo del cursor.  
/// </summary>  
/// <param name="newState">True para bloquear el cursor, false para  
liberarlo.</param>  
private void SetCursorState(bool newState)  
{  
    Cursor.lockState = newState ? CursorLockMode.Locked :  
CursorLockMode.None;  
}  
}  
}
```

ThirdPersonCharacterController:

Aquest script controla el comportament d'un personatge en tercera persona en un entorn multijugador utilitzant Mirror i Unity. Gestiona el moviment, el salt, l'atac, la sincronització d'animacions i la posició del jugador en xarxa. A més, administra la càmera, els punts de control i la transició entre modes de càmera. El seu objectiu és oferir una experiència fluida i sincronitzada per a tots els jugadors en partides multijugador.

```
using UnityEngine;
using Cinemachine;
using System.Collections;
using Mirror;
using UnityEngine.UI;
using UnityEngine.Audio;

#if ENABLE_INPUT_SYSTEM && STARTER_ASSETS_PACKAGES_CHECKED
using UnityEngine.InputSystem;
#endif

namespace StarterAssets
{
    /// <summary>
    /// Controlador de personaje en tercera persona para juegos multijugador con Mirror.
    /// Gestiona el movimiento, animaciones, cámaras, ataques, checkpoints y sincronización de estado en red.
    /// </summary>
    [RequireComponent(typeof(CharacterController))]
#if ENABLE_INPUT_SYSTEM && STARTER_ASSETS_PACKAGES_CHECKED
```

LETHAL RUN



Izan De La Cruz
David Salvador
Marc Rojano
Brian Orozco

```
[RequireComponent(typeof(PlayerInput))]  
#endif  
public class ThirdPersonController : NetworkBehaviour  
{  
    [Header("Network Animator")]  
    [SerializeField] private NetworkAnimator _networkAnimator;  
  
    [Header("Camera Root Position")]  
    [SerializeField] public Vector3 desiredCameraRootPosition = new  
Vector3(-0.0015f, 1.579f, 0.141f);  
    private Vector3 _currentCameraRootPosition;  
    private Vector3 initialRootPosition;  
  
    private GameObject ringMenuObject;  
    private RingMenu ringMenu;  
  
    [Header("Checkpoint")]  
    /// <summary>  
    /// Última posición de checkpoint alcanzada por el jugador.  
    /// </summary>  
    public Vector3 lastCheckpointPosition;  
    /// <summary>  
    /// Indica si el jugador tiene un checkpoint registrado.  
    /// </summary>  
    public bool hasCheckpoint = false;  
  
    [Header("Player")]  
    public float MoveSpeed = 2.0f;  
    public float SprintSpeed = 5.335f;  
    [Range(0.0f, 0.3f)]  
    public float RotationSmoothTime = 0.12f;  
    public float SpeedChangeRate = 10.0f;  
  
    public AudioClip LandingAudioClip;  
    public AudioClip[] FootstepAudioClips;  
    [Range(0, 1)] public float FootstepAudioVolume = 0.5f;  
  
    [Header("Audio Mixer Group")]  
    [SerializeField] private AudioMixerGroup sfxMixerGroup;  
  
    [Space(10)]  
    public float JumpHeight = 1.2f;  
    public float Gravity = -15.0f;  
    public float JumpTimeout = 0.50f;
```

LETHAL RUN



Izan De La Cruz
David Salvador
Marc Rojano
Brian Orozco

```
public float FallTimeout = 0.15f;

[Header("Player Grounded")]
public bool Grounded = true;
public float GroundedOffset = -0.14f;
public float GroundedRadius = 0.28f;
public LayerMask GroundLayers;

[Header("Cinemachine")]
public GameObject CinemachineCameraTarget;
public float TopClamp = 70.0f;
public float BottomClamp = -30.0f;
public float CameraAngleOverride = 0.0f;
public bool LockCameraPosition = false;

private float _cinemachineTargetYaw;
private float _cinemachineTargetPitch;
private float _speed;
private float _animationBlend;
private float _targetRotation = 0.0f;
private float _rotationVelocity;
private float _verticalVelocity;
private float _terminalVelocity = 53.0f;
private float _jumpTimeoutDelta;
private float _fallTimeoutDelta;

private int _animIDSpeed;
private int _animIDGrounded;
private int _animIDJump;
private int _animIDFreeFall;
private int _animIDMotionSpeed;
private int _animIDAttack;

private Vector3 _initialCameraPosition;
[SerializeField] private Vector3 CameraPositionReal = new
Vector3(-0.0015f, 1.579f, 0.141f);

[SyncVar(hook = nameof(OnPositionChanged))]
private Vector3 syncPosition;

[SyncVar(hook = nameof(OnRotationChanged))]
private Quaternion syncRotation;

#if ENABLE_INPUT_SYSTEM && STARTER_ASSETS_PACKAGES_CHECKED
```

LETHAL RUN



Izan De La Cruz
David Salvador
Marc Rojano
Brian Orozco

```
private PlayerInput _playerInput;
#endif
private Animator _animator;
private CharacterController _controller;
private StarterAssetsInputs _input;
private GameObject _mainCamera;

private const float _threshold = 0.01f;
private bool _hasAnimator;

public CinemachineVirtualCamera virtualCamera1;
public CinemachineVirtualCamera virtualCamera2;
public bool isFirstPerson = false;

[SyncVar(hook = nameof(OnAnimationChanged))]
private AnimationState syncAnimationState;

public Transform spineTransform;
private bool _isAttacking = false;

private bool IsCurrentDeviceMouse =>
_playerInput.currentControlScheme == "KeyboardMouse";

/// <summary>
/// Estructura para sincronizar el estado de animación en red.
/// </summary>
public struct AnimationState
{
    public float Speed;
    public bool Grounded;
    public bool Jump;
    public bool FreeFall;
    public bool IsAttacking;
}

private void Awake()
{
    if (_mainCamera == null)
    {
        _mainCamera =
GameObject.FindGameObjectWithTag("MainCamera");
    }
    _hasAnimator = TryGetComponent(out _animator);
    _controller = GetComponent<CharacterController>();
```

```

    _input = GetComponent<StarterAssetsInputs>();
#if ENABLE_INPUT_SYSTEM && STARTER_ASSETS_PACKAGES_CHECKED
    _playerInput = GetComponent<PlayerInput>();
#endif
    AssignAnimationIDs();
}

/// <summary>
/// Comando para sincronizar el estado de animación en red.
/// </summary>
/// <param name="state">Estado de animación a sincronizar.</param>
[Command]
void CmdSyncAnimationState(AnimationState state)
{
    syncAnimationState = state;
}

/// <summary>
/// Hook que actualiza la animación local cuando cambia el estado
/// sincronizado.
/// </summary>
void OnAnimationChanged(AnimationState oldState, AnimationState
newState)
{
    if (_animator == null) return;

    _animator.SetFloat(_animIDSpeed, newState.Speed);
    _animator.SetBool(_animIDGrounded, newState.Grounded);
    _animator.SetBool(_animIDJump, newState.Jump);
    _animator.SetBool(_animIDFreeFall, newState.FreeFall);

    if (newState.IsAttacking)
        _animator.SetTrigger(_animIDAttack);
}

/// <summary>
/// Inicialización y configuración de cámaras virtuales y controles para el
/// jugador local.
/// </summary>
public override void OnStartLocalPlayer()
{
    base.OnStartLocalPlayer();

    _currentCameraRootPosition = desiredCameraRootPosition;
}

```

LETHAL RUN



Izan De La Cruz

David Salvador

Marc Rojano

Brian Orozco

```
SetCameraRootPosition();

#if ENABLE_INPUT_SYSTEM && STARTER_ASSETS_PACKAGES_CHECKED
    if (_playerInput != null)
    {
        _playerInput.enabled = true;
        _playerInput.SwitchCurrentControlScheme("KeyboardMouse",
Keyboard.current, Mouse.current);
    }
    else
    {
        return;
    }
#endif
    if (_input != null) _input.enabled = true;
    if (_controller != null) _controller.enabled = true;
    if (CinemachineCameraTarget != null)
    {
        CinemachineCameraTarget.transform.localPosition =
desiredCameraRootPosition;
    }

    GameObject camera1Obj =
GameObject.FindGameObjectWithTag("VirtualCamera1");
    GameObject camera2Obj =
GameObject.FindGameObjectWithTag("VirtualCamera2");

    SetupVirtualCamera(camera1Obj, ref virtualCamera1, true);
    SetupVirtualCamera(camera2Obj, ref virtualCamera2, false);
}

/// <summary>
/// Establece la posición raíz de la cámara.
/// </summary>
private void SetCameraRootPosition()
{
    if (CinemachineCameraTarget != null)
    {
        CinemachineCameraTarget.transform.localPosition =
desiredCameraRootPosition;
    }
}

/// <summary>
```

LETHAL RUN



Izan De La Cruz

David Salvador

Marc Rojano

Brian Orozco

```
/// Configura las cámaras virtuales de la escena.  
/// </summary>  
private void SetupVirtualCameras()  
{  
    GameObject camera1Obj =  
GameObject.FindGameObjectWithTag("VirtualCamera1");  
    GameObject camera2Obj =  
GameObject.FindGameObjectWithTag("VirtualCamera2");  
  
    SetupVirtualCamera(camera1Obj, ref virtualCamera1, true);  
    SetupVirtualCamera(camera2Obj, ref virtualCamera2, false);  
}  
  
/// <summary>  
/// Devuelve el estado actual de animación.  
/// </summary>  
private AnimationState GetCurrentAnimationState()  
{  
    return new AnimationState  
{  
        Speed = _animationBlend,  
        Grounded = Grounded,  
        Jump = !_controller.isGrounded && _input.jump,  
        FreeFall = !_controller.isGrounded && !_input.jump,  
        IsAttacking = _isAttacking  
    };  
}  
  
/// <summary>  
/// Configura una cámara virtual específica.  
/// </summary>  
private void SetupVirtualCamera(GameObject cameraObj, ref  
CinemachineVirtualCamera virtualCamera, bool isActive)  
{  
    if (cameraObj != null)  
    {  
        virtualCamera =  
cameraObj.GetComponent<CinemachineVirtualCamera>();  
        if (virtualCamera != null)  
        {  
            virtualCamera.Follow = CinemachineCameraTarget.transform;  
            virtualCamera.LookAt = null;  
            virtualCamera.gameObject.SetActive(isActive);  
        }  
    }  
}
```

LETHAL RUN



Izan De La Cruz
David Salvador
Marc Rojano
Brian Orozco

```
        }

    }

private void Start()
{
    _hasAnimator = TryGetComponent(out _animator);
    _controller = GetComponent<CharacterController>();
    _input = GetComponent<StarterAssetsInputs>();
#ifndef ENABLE_INPUT_SYSTEM && STARTER_ASSETS_PACKAGES_CHECKED
    _playerInput = GetComponent<PlayerInput>();
#endif
    initialRootPosition = transform.position;
    AssignAnimationIDs();
    _jumpTimeoutDelta = JumpTimeout;
    _fallTimeoutDelta = FallTimeout;
    Cursor.visible = true;
    Cursor.lockState = CursorLockMode.None;

    if (_networkAnimator == null)
        _networkAnimator = GetComponent<NetworkAnimator>();

    if (!isLocalPlayer)
    {
        if (_input != null) _input.enabled = false;
        if (_controller != null) _controller.enabled = false;
        Transform cameraRoot = transform.Find("PlayerCameraRoot");
        if (cameraRoot != null) cameraRoot.gameObject.SetActive(false);
        return;
    }

    if (CinemachineCameraTarget != null)
    {
        _initialCameraPosition = CameraPositionReal;
        CinemachineCameraTarget.transform.localPosition =
        _initialCameraPosition;
    }

    LoadCheckpoint();
}

/// <summary>
/// Sincroniza la posición del jugador en red.
/// </summary>
void OnPositionChanged(Vector3 oldPosition, Vector3 newPosition)
```

LETHAL RUN



Izan De La Cruz

David Salvador

Marc Rojano

Brian Orozco

```
{  
    if (!isLocalPlayer)  
    {  
        transform.position = newPosition;  
    }  
}  
  
/// <summary>  
/// Sincroniza la rotación del jugador en red.  
/// </summary>  
void OnRotationChanged(Quaternion oldRotation, Quaternion  
newRotation)  
{  
    if (!isLocalPlayer)  
    {  
        transform.rotation = newRotation;  
    }  
}  
  
private void Update()  
{  
#if ENABLE_INPUT_SYSTEM && STARTER_ASSETS_PACKAGES_CHECKED  
    if (Keyboard.current.escapeKey.wasPressedThisFrame)  
    {  
        if (Cursor.visible)  
        {  
            Cursor.visible = false;  
            Cursor.lockState = CursorLockMode.Locked;  
        }  
        else  
        {  
            Cursor.visible = true;  
            Cursor.lockState = CursorLockMode.None;  
        }  
    }  
#endif  
  
    if (!isLocalPlayer) return;  
  
    _hasAnimator = TryGetComponent(out _animator);  
  
    JumpAndGravity();  
    GroundedCheck();  
    Move();
```

LETHAL RUN



Izan De La Cruz
David Salvador
Marc Rojano
Brian Orozco

```
#if ENABLE_INPUT_SYSTEM && STARTER_ASSETS_PACKAGES_CHECKED
    if (Keyboard.current.pKey.wasPressedThisFrame)
    {
        SwitchCamera();
    }
    if (Mouse.current.leftButton.wasPressedThisFrame)
    {
        if (RingMenu.Instance == null ||
!RingMenu.Instance.gameObject.activeSelf)
        {
            Attack();
        }
    }
#endif

        CmdSyncTransform(transform.position, transform.rotation);
    }

/// <summary>
/// Comando para sincronizar posición y rotación en red.
/// </summary>
[Command]
void CmdSyncTransform(Vector3 position, Quaternion rotation)
{
    syncPosition = position;
    syncRotation = rotation;
}

private void LateUpdate()
{
    if (!isLocalPlayer) return;
    CameraRotation();
    if (CinemachineCameraTarget != null)
    {
        Vector3 targetPosition = _initialCameraPosition;
        if (isFirstPerson)
        {
            // Ajustes adicionales para primera persona si es necesario
        }
        CinemachineCameraTarget.transform.localPosition = Vector3.Lerp(
            CinemachineCameraTarget.transform.localPosition,
            targetPosition,
            Time.deltaTime * 10f
    }
}
```

LETHAL RUN



Izan De La Cruz

David Salvador

Marc Rojano

Brian Orozco

```
        );
    }
}

/// <summary>
/// Asigna los IDs de los parámetros de animación.
/// </summary>
private void AssignAnimationIDs()
{
    _animIDSpeed = Animator.StringToHash("Speed");
    _animIDGrounded = Animator.StringToHash("Grounded");
    _animIDJump = Animator.StringToHash("Jump");
    _animIDFreeFall = Animator.StringToHash("FreeFall");
    _animIDMotionSpeed = Animator.StringToHash("MotionSpeed");
    _animIDAttack = Animator.StringToHash("Attack");
}

/// <summary>
/// Comprueba si el jugador está en el suelo.
/// </summary>
private void GroundedCheck()
{
    Vector3 spherePosition = new Vector3(transform.position.x,
transform.position.y - GroundedOffset, transform.position.z);
    Grounded = Physics.CheckSphere(spherePosition, GroundedRadius,
GroundLayers, QueryTriggerInteraction.Ignore);

    if (_hasAnimator)
    {
        _animator.SetBool(_animIDGrounded, Grounded);
    }
}

/// <summary>
/// Ejecuta el ataque del jugador y lo sincroniza en red.
/// </summary>
private void Attack()
{
    if (!_isAttacking)
    {
        _isAttacking = true;
        CmdPerformAttack();
        StartCoroutine(ResetAttackState());
    }
}
```

LETHAL RUN



Izan De La Cruz
David Salvador
Marc Rojano
Brian Orozco

```
}
```

```
/// <summary>  
/// Comando para ejecutar el ataque en el servidor.  
/// </summary>  
[Command]  
void CmdPerformAttack()  
{  
    RpcPerformAttack();  
}
```

```
/// <summary>  
/// RPC para ejecutar el ataque en todos los clientes.  
/// </summary>  
[ClientRpc]  
void RpcPerformAttack()  
{  
    if (_animator != null)  
        _animator.SetTrigger(_animIDAttack);  
}
```

```
/// <summary>  
/// Corrutina que reinicia el estado de ataque tras un breve tiempo.  
/// </summary>  
private IEnumerator ResetAttackState()  
{  
    yield return new WaitForSeconds(0.5f);  
    _isAttacking = false;  
}
```

```
/// <summary>  
/// Actualiza la rotación de la cámara según la entrada del jugador.  
/// </summary>  
private void CameraRotation()  
{  
    if (!isLocalPlayer) return;  
    if (_input.look.sqrMagnitude >= _threshold && !LockCameraPosition)  
    {  
        float deltaTimeMultiplier = IsCurrentDeviceMouse ? 1.0f :  
Time.deltaTime;  
        _cinemachineTargetYaw += _input.look.x * deltaTimeMultiplier;  
        _cinemachineTargetPitch = Mathf.Clamp(_cinemachineTargetPitch +  
_input.look.y * deltaTimeMultiplier, BottomClamp, TopClamp);  
    }  
}
```

```
    CinemachineCameraTarget.transform.rotation =
Quaternion.Euler(_cinemachineTargetPitch + CameraAngleOverride,
_cinemachineTargetYaw, 0.0f);
}
}
```

PlayerHealth:

Aquest script gestiona la salut del jugador en un joc, permetent rebre mal, mostrar la barra de vida i atorgar invencibilitat temporal després de ser ferit. Quan la salut arriba a zero, desactiva el control del personatge, reproduceix l'animació de mort i, després d'un temps, el reviu a l'últim punt de control si existeix. Així, controla tot el cicle de dany, mort i suport del jugador.

```
using UnityEngine;
using UnityEngine.UI;
using StarterAssets;
using System.Collections;

/// <summary>
/// Gestiona la salud del jugador, el daño recibido, la invencibilidad temporal, la muerte y el respawn.
/// También actualiza la interfaz de usuario de la barra de vida.
/// </summary>
public class PlayerHealth : MonoBehaviour
{
    [Header("Health Settings")]
    /// <summary>
    /// Salud máxima del jugador.
    /// </summary>
    public float maxHealth = 100f;

    /// <summary>
    /// Salud actual del jugador.
    /// </summary>
    public float currentHealth;

    /// <summary>
    /// Referencia al slider de la barra de vida.
    /// </summary>
```

LETHAL RUN



Izan De La Cruz

David Salvador

Marc Rojano

Brian Orozco

```
/// </summary>
public Slider healthSlider;

/// <summary>
/// Tiempo de invencibilidad tras recibir daño.
/// </summary>
public float invincibilityTime = 1f;

[Header("Respawn Settings")]
[SerializeField] private float respawnTime = 3.9f;

// Componentes
private Animator animator;
private ThirdPersonController thirdPersonController;
private CharacterController characterController;
private bool isInvincible = false;
private bool isRespawning = false;

/// <summary>
/// Inicializa la salud y asigna componentes y el slider de vida al jugador
local.
/// </summary>
void Start()
{
    currentHealth = maxHealth;
    thirdPersonController = GetComponent<ThirdPersonController>();
    animator = GetComponent<Animator>();
    characterController = GetComponent<CharacterController>();

    // Asignar dinámicamente el slider al jugador local
    if (thirdPersonController.isLocalPlayer)
    {
        GameObject sliderObj = GameObject.FindGameObjectWithTag("Slider");
        if (sliderObj != null)
        {
            healthSlider = sliderObj.GetComponent<Slider>();
            healthSlider.MaxValue = maxHealth;
            healthSlider.value = maxHealth;
        }
    }
}

/// <summary>
```

```
/// Aplica daño al jugador, inicia invencibilidad temporal y gestiona la muerte
// si la salud llega a cero.
/// </summary>
/// <param name="amount">Cantidad de daño a recibir.</param>
public void TakeDamage(float amount)
{
    if (isInvincible || isRespawning) return;

    currentHealth = Mathf.Clamp(currentHealth - amount, 0f, maxHealth);
    UpdateHealthUI();

    if (currentHealth <= 0) Die();
    else StartCoroutine(InvincibilityRoutine());
}

/// <summary>
/// Actualiza la barra de vida en la interfaz de usuario.
/// </summary>
private void UpdateHealthUI()
{
    if (healthSlider != null)
        healthSlider.value = currentHealth;
}

/// <summary>
/// Corrutina que otorga invencibilidad temporal tras recibir daño.
/// </summary>
private IEnumerator InvincibilityRoutine()
{
    isInvincible = true;
    yield return new WaitForSeconds(invincibilityTime);
    isInvincible = false;
}

/// <summary>
/// Gestiona la muerte del jugador, desactiva el control y lanza el respawn.
/// </summary>
private void Die()
{
    isRespawning = true;
    animator.SetBool("IsDead", true);
    thirdPersonController.enabled = false;
    StartCoroutine(RespawnRoutine());
}
```

```

/// <summary>
/// Corrutina que gestiona el respawn del jugador tras morir.
/// </summary>
private IEnumerator RespawnRoutine()
{
    yield return new WaitForSeconds(respawnTime);

    if (thirdPersonController.hasCheckpoint)
    {
        characterController.enabled = false;
        transform.position = thirdPersonController.lastCheckpointPosition;
        characterController.enabled = true;
    }

    currentHealth = maxHealth;
    UpdateHealthUI();
    animator.SetBool("IsDead", false);
    thirdPersonController.enabled = true;
    isRespawning = false;
}
}
  
```

Controller(RingEmote):

Aquest script controla l'activació i desactivació del menú circular (RingMenu) per al jugador local en un joc multijugador utilitzant Mirror. Quan es prem la tecla Tab, es mostra el menú i s'oculta en deixar-la anar, assegurant que només el jugador local pugui interactuar amb aquest element de la interfície. Així, facilita l'accés ràpid a opcions o habilitats mitjançant el RingMenu durant la partida.

```

using UnityEngine;
using Mirror;

/// <summary>
  
```

LETHAL RUN



Izan De La Cruz

David Salvador

Marc Rojano

Brian Orozco

```
/// Controla la activación y desactivación del menú circular (RingMenu) para el
/// jugador local en un entorno multijugador.
/// Solo el jugador local puede mostrar u ocultar el RingMenu usando la tecla
/// Tab.
/// </summary>
public class Controller : NetworkBehaviour
{
    /// <summary>
    /// Inicializa el estado del RingMenu al iniciar el jugador local.
    /// </summary>
    void Start()
    {
        if (!isLocalPlayer) return; // Solo el jugador local ejecuta esto

        if (RingMenu.Instance != null)
        {
            RingMenu.Instance.gameObject.SetActive(false);
        }
    }

    /// <summary>
    /// Controla la visibilidad del RingMenu según la entrada del jugador local.
    /// </summary>
    void Update()
    {
        if (!isLocalPlayer || RingMenu.Instance == null) return; // Solo el jugador
        local lo controla

        if (Input.GetKeyDown(KeyCode.Tab))
        {
            RingMenu.Instance.gameObject.SetActive(true);
        }

        if (Input.GetKeyUp(KeyCode.Tab))
        {
            RingMenu.Instance.gameObject.SetActive(false);
        }
    }
}
```

{}

Ring Menu (Emotes/graffitis):

Aquest script controla el menú circular (RingMenu) que permet al jugador local seleccionar animacions o accions especials, com fer grafits a l'escenari. Gestiona la visualització i selecció de les opcions del menú, així com l'execució de l'acció escollida, incloent la interacció amb el backend per obtenir imatges personalitzades. A més, assegura que només el jugador local pugui interactuar amb aquest menú i que la interfície sigui intuïtiva i visualment clara.

```
using System.Collections;
using StarterAssets;
using UnityEngine;
using UnityEngine.Networking;
using Mirror;

/// <summary>
/// Controla el menú circular (RingMenu) que permite al jugador local seleccionar acciones o grafitis.
/// Gestiona la visualización, selección y ejecución de acciones del menú, incluyendo la lógica de red y la interacción con el backend.
/// </summary>
public class RingMenu : MonoBehaviour
{
    /// <summary>
    /// Lista de animaciones o acciones disponibles en el menú circular.
    /// </summary>
    public RingAnimation[] ringAnimations;

    /// <summary>
    /// Prefab de cada pieza del menú circular.
    /// </summary>
    public RingPiece ringPiecePrefab;

    private Animator playerAnimator;
    private RingPiece[] ringPieces;
```

LETHAL RUN



Izan De La Cruz
David Salvador
Marc Rojano
Brian Orozco

```
private float degreesPerPiece;
private float gapDegrees = 2f;
private bool triedFindingPlayer = false;
private bool menusClosing = false;
private ThirdPersonController _localPlayer;

/// <summary>
/// Instancia singleton del RingMenu.
/// </summary>
public static RingMenu Instance { get; private set; }

/// <summary>
/// Asegura que solo haya una instancia activa de RingMenu.
/// </summary>
private void Awake()
{
    if (Instance != null && Instance != this)
    {
        Destroy(this.gameObject);
        return;
    }
    Instance = this;
}

/// <summary>
/// Inicializa las piezas del menú circular y su disposición.
/// </summary>
private void Start()
{
    degreesPerPiece = 360f / ringAnimations.Length;
    float distanceTolcon =
    Vector3.Distance(ringPiecePrefab.icon.transform.position,
    ringPiecePrefab.background.transform.position);
    ringPieces = new RingPiece[ringAnimations.Length];

    for (int i = 0; i < ringAnimations.Length; i++)
    {
        ringPieces[i] = Instantiate(ringPiecePrefab, transform);
        ringPieces[i].background.fillAmount = (1f / ringAnimations.Length) -
(gapDegrees / 360f);
        ringPieces[i].background.transform.localRotation = Quaternion.Euler(0, 0,
degreesPerPiece / 2f + gapDegrees / 2f + i * degreesPerPiece);
        ringPieces[i].icon.sprite = ringAnimations[i].icon;
```

LETHAL RUN



Izan De La Cruz

David Salvador

Marc Rojano

Brian Orozco

```
    Vector3 directionVector = Quaternion.AngleAxis(i * degreesPerPiece,
Vector3.forward) * Vector3.up;
    Vector3 movementVector = directionVector * distanceTolcon;
    ringPieces[i].icon.transform.localPosition =
ringPieces[i].background.transform.localPosition + movementVector;
}

}

/// <summary>
/// Prepara el menú al activarse, mostrando el cursor y desbloqueándolo.
/// </summary>
private void OnEnable()
{
    menuIsClosing = false;
    Cursor.visible = true;
    Cursor.lockState = CursorLockMode.None;
}

/// <summary>
/// Actualiza el menú cada frame: busca el jugador local, resalta el elemento activo
y responde a la entrada del ratón.
/// </summary>
private void Update()
{
    if (_localPlayer == null)
        TryFindLocalPlayer();

    int activeElement = GetActiveElement();
    HighlightActiveElement(activeElement);
    RespondToMouseInput(activeElement);
}

/// <summary>
/// Intenta encontrar y asignar el jugador local y su animador.
/// </summary>
private void TryFindLocalPlayer()
{
    foreach (var player in FindObjectsOfType<ThirdPersonController>())
    {
        if (player.isLocalPlayer)
        {
            _localPlayer = player;
            playerAnimator = player.GetComponentInChildren<Animator>();
            triedFindingPlayer = true;
        }
    }
}
```

LETHAL RUN



Izan De La Cruz

David Salvador

Marc Rojano

Brian Orozco

```
        return;
    }
}
// Si no lo encuentra, sigue intentando en Update
}

/// <summary>
/// Calcula qué elemento del menú está activo según la posición del ratón.
/// </summary>
/// <returns>Índice del elemento activo.</returns>
private int GetActiveElement()
{
    Vector3 screenCenter = new Vector3(Screen.width / 2, Screen.height / 2);
    Vector3 cursorVector = Input.mousePosition - screenCenter;

    float mouseAngle = Vector3.SignedAngle(Vector3.up, cursorVector,
    Vector3.forward) + degreesPerPiece / 2f;
    float normalizedMouseAngle = NormalizeAngle(mouseAngle);

    return (int)(normalizedMouseAngle / degreesPerPiece);
}

/// <summary>
/// Resalta visualmente el elemento actualmente activo del menú.
/// </summary>
/// <param name="activeElement">Índice del elemento activo.</param>
private void HighlightActiveElement(int activeElement)
{
    for (int i = 0; i < ringPieces.Length; i++)
    {
        ringPieces[i].background.color = (i == activeElement)
            ? new Color(1f, 1f, 1f, 0.75f)
            : new Color(1f, 1f, 1f, 0.5f);
    }
}

/// <summary>
/// Ejecuta la acción asociada al elemento activo cuando el jugador hace clic.
/// </summary>
/// <param name="activeElement">Índice del elemento activo.</param>
private void RespondToMouseInput(int activeElement)
{
    if (menuIsClosing) return;
```

LETHAL RUN



Izan De La Cruz
David Salvador
Marc Rojano
Brian Orozco

```
if (Input.GetMouseButtonDown(0))
{
    menusClosing = true;
    var selected = ringAnimations[activeElement];

    if (selected.actionType == RingActionType.Graffiti)
    {
        Ray ray = Camera.main.ScreenPointToRay(Input.mousePosition);
        RaycastHit hit;
        if (Physics.Raycast(ray, out hit, 100f))
        {
            Vector3 hitPoint = hit.point + hit.normal * 0.01f;
            Quaternion hitRotation = Quaternion.LookRotation(-hit.normal);

            // Lanza la corrutina para obtener la URL y pintar el grafiti
            StartCoroutine(GetGraffitiUrlWithTimeoutAndPaint(selected.graffitiUrl,
hitPoint, hitRotation));
        }
        else
        {
            gameObject.SetActive(false);
        }
    }
    else if (_localPlayer != null)
    {
        // Sincroniza la animación en red
        _localPlayer.CmdPlayAnimationTrigger(selected.triggerName);
        gameObject.SetActive(false);
    }
}

/// <summary>
/// Corrutina que obtiene la URL del grafiti desde el backend (con timeout) y llama
a la función de pintado.
/// </summary>
/// <param name="fallbackUrl">URL de respaldo si falla la petición.</param>
/// <param name="hitPoint">Punto de impacto donde pintar.</param>
/// <param name="hitRotation">Rotación para el grafiti.</param>
private IEnumerator GetGraffitiUrlWithTimeoutAndPaint(string fallbackUrl, Vector3
hitPoint, Quaternion hitRotation)
{
    int userId = PlayerPrefs.GetInt("user.id", -1);
    string token = PlayerPrefs.GetString("user.token", "");
```

LETHAL RUN



Izan De La Cruz

David Salvador

Marc Rojano

Brian Orozco

```
if (userId == -1 || string.IsNullOrEmpty(token))
{
    yield return PaintWithFallback(fallbackUrl, hitPoint, hitRotation);
    yield break;
}

string apiUrl =
$"https://lethalrun.cat/images-service/pictures/get-active-picture/{userId}";
float timeout = 5f;
string graffitiUrl = fallbackUrl;

UnityWebRequest www = UnityWebRequest.Get(apiUrl);
www.SetRequestHeader("Authorization", "Bearer " + token);

var operation = www.SendWebRequest();

float timer = 0f;
bool usedFallback = false;
while (!operation.isDone)
{
    if (timer > timeout)
    {
        www.Abort();
        usedFallback = true;
        break;
    }
    timer += Time.deltaTime;
    yield return null;
}

if (!usedFallback && www.result == UnityWebRequest.Result.Success)
{
    string json = www.downloadHandler.text;
    try
    {
        GraffitiPictureResponse response =
JsonUtility.FromJson<GraffitiPictureResponse>(json);
        if (string.IsNullOrEmpty(response.path))
        {
            graffitiUrl = fallbackUrl;
        }
        else
        {
```

LETHAL RUN



Izan De La Cruz

David Salvador

Marc Rojano

Brian Orozco

```
graffitiUrl =
"https://lethalrun.cat/images-service/images/users/{userId}/{response.path}";
    }
}
catch
{
    graffitiUrl = fallbackUrl;
}
}

var localPlayer = FindLocalPlayer();
if (localPlayer != null)
{
    localPlayer.TryPaintGraffiti(graffitiUrl, hitPoint, hitRotation);
}

gameObject.SetActive(false);
Cursor.visible = false;
Cursor.lockState = CursorLockMode.Locked;
}

/// <summary>
/// Clase auxiliar para parsear la respuesta del endpoint de grafitis.
/// </summary>
[System.Serializable]
private class GraffitiPictureResponse
{
    public int id;
    public bool is_active;
    public string path;
    public string createdAt;
    public string updatedAt;
    public int user_id;
}

/// <summary>
/// Corrutina auxiliar para pintar un grafiti usando la URL de respaldo.
/// </summary>
private IEnumerator PaintWithFallback(string fallbackUrl, Vector3 hitPoint,
Quaternion hitRotation)
{
    var localPlayer = FindLocalPlayer();
    if (localPlayer != null)
    {
```

```
        localPlayer.TryPaintGraffiti(fallbackUrl, hitPoint, hitRotation);
    }
    gameObject.SetActive(false);
    Cursor.visible = false;
    Cursor.lockState = CursorLockMode.Locked;
    yield break;
}

/// <summary>
/// Busca y retorna el jugador local en la escena.
/// </summary>
/// <returns>Referencia al controlador del jugador local o null si no se
encuentra.</returns>
private ThirdPersonController FindLocalPlayer()
{
    foreach (var player in FindObjectsOfType<ThirdPersonController>())
    {
        if (player.isLocalPlayer)
            return player;
    }
    return null;
}

/// <summary>
/// Normaliza un ángulo para que esté entre 0 y 360 grados.
/// </summary>
private float NormalizeAngle(float a) => (a + 360f) % 360f;
}
```

Trampas:**LavaTrap:**

Aquest script gestiona el dany periòdic que rep un jugador en romandre dins d'una zona de lava al joc. Detecta quan el jugador entra, roman o surt de l'àrea perillosa, aplicant dany a intervals regulars mentre està en contacte amb la lava. També controla el temporitzador per assegurar que el dany s'apliqui correctament i de manera contínua només mentre el jugador està exposat. Així, facilita la mecànica de trampes ambientals a l'escenari.

LETHAL RUN



Izan De La Cruz

David Salvador

Marc Rojano

Brian Orozco

using UnityEngine;

```
/// <summary>
/// Gestiona el daño periódico al jugador cuando permanece dentro de una zona de lava.
/// Aplica daño a intervalos definidos mientras el jugador esté en contacto con la lava.
/// </summary>
public class LavaTrap : MonoBehaviour
{
    /// <summary>
    /// Cantidad de daño que la lava infinge al jugador en cada intervalo.
    /// </summary>
    public float damageAmount = 15f;

    /// <summary>
    /// Intervalo de tiempo (en segundos) entre cada aplicación de daño.
    /// </summary>
    public float damageInterval = 0.5f;

    /// <summary>
    /// Temporizador para controlar cuándo aplicar el siguiente daño.
    /// </summary>
    private float damageTimer;

    /// <summary>
    /// Indica si el jugador está actualmente dentro de la lava.
    /// </summary>
    private bool isPlayerInLava = false;

    /// <summary>
    /// Actualiza el temporizador de daño si el jugador está en la lava.
    /// </summary>
    private void Update()
    {
        if (isPlayerInLava && damageTimer > 0)
        {
            damageTimer -= Time.deltaTime;
        }
    }

    /// <summary>
    /// Detecta cuando el jugador entra en la zona de la lava y reinicia el temporizador para aplicar daño inmediato.
    /// </summary>
```

LETHAL RUN



Izan De La Cruz

David Salvador

Marc Rojano

Brian Orozco

```
/// <param name="other">Collider que entra en el trigger.</param>
private void OnTriggerEnter(Collider other)
{
    if (other.CompareTag("Player"))
    {
        isPlayerInLava = true;
        damageTimer = 0f;
    }
}

/// <summary>
/// Aplica daño periódico al jugador mientras permanece en la lava.
/// </summary>
/// <param name="other">Collider que permanece en el trigger.</param>
private void OnTriggerStay(Collider other)
{
    if (other.CompareTag("Player"))
    {
        PlayerHealth playerHealth = other.GetComponent<PlayerHealth>();
        if (playerHealth != null && damageTimer <= 0f)
        {
            playerHealth.TakeDamage(damageAmount);
            damageTimer = damageInterval;
        }
    }
}

/// <summary>
/// Detecta cuando el jugador sale de la zona de la lava y reinicia el estado y
temporizador.
/// </summary>
/// <param name="other">Collider que sale del trigger.</param>
private void OnTriggerExit(Collider other)
{
    if (other.CompareTag("Player"))
    {
        isPlayerInLava = false;
        damageTimer = 0f;
    }
}
```

TrampaPinchos:

Aquest script gestiona una trampa de punxes activable en xarxa per a un joc multijugador. Controla l'animació de pujar i baixar les punxes, sincronitza el seu estat entre servidor i clients, i aplica dany al jugador que entra en contacte amb elles quan estan activades. Així, permet implementar mecàniques de trampes interactives i sincronitzades a l'escenari de joc.

```
using UnityEngine;
using System.Collections;
using Mirror;

/// <summary>
/// Controla el comportamiento de una trampa de pinchos activable en red.
/// Gestiona la animación de subir y bajar los pinchos, la sincronización en red y el
/// daño al jugador.
/// </summary>
public class TrampaPinchos : NetworkBehaviour, IActivable
{
    /// <summary>
    /// Tiempo que tarda en activarse la trampa (en segundos).
    /// </summary>
    public float tiempoActivacion = 0.5f;

    /// <summary>
    /// Altura máxima que alcanzan los pinchos al activarse.
    /// </summary>
    public float alturaMaxima = 1f;

    /// <summary>
    /// Tiempo que los pinchos permanecen arriba antes de volver a bajar.
    /// </summary>
    public float tiempoArriba = 1f;

    /// <summary>
    /// Cantidad de daño que infligen los pinchos al jugador.
    /// </summary>
    public float damageAmount = 100f;

    /// <summary>
    /// Posición inicial de los pinchos.
    /// </summary>
    private Vector3 posicionInicial;
```

LETHAL RUN



Izan De La Cruz

David Salvador

Marc Rojano

Brian Orozco

```
/// <summary>
/// Indica si la trampa está activada, sincronizado en red.
/// </summary>
[SyncVar(hook = nameof(OnActivadaChanged))]
private bool activada = false;

/// <summary>
/// Guarda la posición inicial al iniciar el objeto.
/// </summary>
void Start()
{
    posicionInicial = transform.position;
}

/// <summary>
/// Activa la trampa en el servidor si no está ya activada.
/// </summary>
[Server]
public void Activar()
{
    if (!activada)
    {
        activada = true;
    }
}

/// <summary>
/// Hook que se ejecuta cuando cambia el estado de activación de la trampa.
/// Inicia la animación de subir los pinchos.
/// </summary>
void OnActivadaChanged(bool oldValue, bool newValue)
{
    if (newValue)
    {
        StartCoroutine(SubirPinchos());
    }
}

/// <summary>
/// Llama a la animación de subir pinchos en todos los clientes.
/// </summary>
[ClientRpc]
void RpcSubirPinchos()
```

LETHAL RUN



Izan De La Cruz
David Salvador
Marc Rojano
Brian Orozco

```
{  
    StartCoroutine(SubirPinchos());  
}  
  
/// <summary>  
/// Corrutina que gestiona el movimiento de los pinchos hacia arriba y su  
permanencia.  
/// </summary>  
IEnumerator SubirPinchos()  
{  
    yield return new WaitForSeconds(tiempoActivacion);  
    transform.position = posicionInicial + Vector3.up * alturaMaxima;  
    yield return new WaitForSeconds(tiempoArriba);  
    StartCoroutine(BajarPinchos());  
}  
  
/// <summary>  
/// Corrutina que baja los pinchos y reinicia el estado de la trampa.  
/// </summary>  
IEnumerator BajarPinchos()  
{  
    transform.position = posicionInicial;  
    if (isServer)  
    {  
        activada = false;  
    }  
    yield return null;  
}  
  
/// <summary>  
/// Aplica daño al jugador que entra en contacto con los pinchos cuando están  
activados.  
/// </summary>  
/// <param name="other">Collider del objeto que entra en la trampa.</param>  
private void OnTriggerEnter(Collider other)  
{  
    if (activada && other.CompareTag("Player"))  
    {  
        PlayerHealth playerHealth = other.GetComponent<PlayerHealth>();  
        if (playerHealth != null)  
        {  
            playerHealth.TakeDamage(damageAmount);  
        }  
    }  
}
```

{
}

BolaMovimiento:

Aquest script controla una bola trampa activable en un entorn multijugador amb Mirror. Gestiona la seva activació, el moviment físic i la sincronització d'estat entre servidor i clients, aplicant dany i retrocés (knockback) als jugadors que hi col·lideixen. A més, s'encarrega de desactivar i reiniciar automàticament la bola després d'un temps o en arribar a un punt de finalització, incloent-hi la visualització d'efectes especials.

```
using UnityEngine;
using System.Collections;
using Mirror;

/// <summary>
/// Controla el comportamiento de una bola trampa activable en red.
/// Gestiona su activación, movimiento, daño, knockback y sincronización entre
/// servidor y clientes.
/// </summary>
public class ControlBola : NetworkBehaviour, IActivable
{
    [Header("Configuración Bola")]
    /// <summary>
    /// Duración en segundos que la bola permanece activa.
    /// </summary>
    public float duracionActivacion = 30f;

    /// <summary>
    /// Fuerza de impulso inicial hacia abajo al activar la bola.
    /// </summary>
    public float fuerzalImpulso = 20f;

    /// <summary>
    /// Transform que indica el punto inicial de la bola.
    /// </summary>
    public Transform puntolInicial;
```

LETHAL RUN



Izan De La Cruz

David Salvador

Marc Rojano

Brian Orozco

```
/// <summary>
/// Cantidad de daño que infinge la bola al jugador.
/// </summary>
public float damageAmount = 20f;

/// <summary>
/// Fuerza de retroceso aplicada al jugador al ser golpeado.
/// </summary>
public float knockbackForce = 10f;

[Header("Efectos")]
/// <summary>
/// Prefab del efecto visual al desactivar la bola.
/// </summary>
public GameObject efectoDesactivacion;

/// <summary>
/// Indica si la bola está activa, sincronizado en red.
/// </summary>
[SyncVar(hook = nameof(OnBolaActivaChanged))]
private bool bolaActiva = false;

/// <summary>
/// Posición sincronizada de la bola.
/// </summary>
[SyncVar]
private Vector3 syncedPosition;

/// <summary>
/// Velocidad sincronizada de la bola.
/// </summary>
[SyncVar]
private Vector3 syncedVelocity;

private Rigidbody rb;
private Vector3 posicionOriginal;
private bool estaDesactivando = false;

/// <summary>
/// Inicializa componentes y guarda la posición original.
/// </summary>
void Awake()
{
    rb = GetComponent<Rigidbody>();
```

LETHAL RUN



Izan De La Cruz

David Salvador

Marc Rojano

Brian Orozco

```
    posicionOriginal = transform.position;
}

/// <summary>
/// Configura los parámetros físicos de la bola al iniciar en el servidor.
/// </summary>
public override void OnStartServer()
{
    ConfigurarFisica();
}

/// <summary>
/// Configura las propiedades físicas del Rigidbody según el estado de la bola.
/// </summary>
void ConfigurarFisica()
{
    rb.isKinematic = !bolaActiva;
    rb.useGravity = bolaActiva;
    rb.mass = 1f;
    rb.linearDamping = 0.1f;
    rb.angularDamping = 0.1f;
    Physics.gravity = new Vector3(0, -9.81f, 0);
}

/// <summary>
/// Sincroniza la posición y velocidad de la bola entre servidor y clientes.
/// </summary>
void FixedUpdate()
{
    if (isServer)
    {
        if (bolaActiva)
        {
            rb.AddForce(Vector3.down * fuerzalImpulso, ForceMode.Force);
        }
        syncedPosition = transform.position;
        syncedVelocity = rb.linearVelocity;
    }
    else
    {
        transform.position = Vector3.Lerp(transform.position, syncedPosition,
Time.fixedDeltaTime * 10f);
        rb.linearVelocity = syncedVelocity;
    }
}
```

LETHAL RUN



Izan De La Cruz

David Salvador

Marc Rojano

Brian Orozco

```
}

/// <summary>
/// Hook que responde al cambio de estado de activación de la bola.
/// </summary>
void OnBolaActivaChanged(bool oldValue, bool newValue)
{
    if (newValue)
    {
        if (isServer)
        {
            rb.isKinematic = false;
            rb.useGravity = true;
            rb.linearVelocity = Vector3.zero;
            rb.angularVelocity = Vector3.zero;
            rb.AddForce(Vector3.down * fuerzalimpulso, ForceMode.Impulse);
        }
    }
    else
    {
        if (isServer)
        {
            ResetearBola();
        }
    }
}

/// <summary>
/// Reinicia la bola a su posición y estado original en el servidor.
/// </summary>
[Server]
private void ResetearBola()
{
    rb.isKinematic = true;
    rb.useGravity = false;
    rb.linearVelocity = Vector3.zero;
    rb.angularVelocity = Vector3.zero;
    transform.position = posicionOriginal;
    transform.rotation = Quaternion.identity;
}

/// <summary>
/// Activa la bola y comienza la cuenta atrás para su desactivación.
/// </summary>
```

LETHAL RUN



Izan De La Cruz

David Salvador

Marc Rojano

Brian Orozco

```
[Server]
public void Activar()
{
    if (!bolaActiva && !estaDesactivando)
    {
        bolaActiva = true;
        StartCoroutine(DesactivarBolaDespuesDeTiempo());
    }
}

/// <summary>
/// Detecta la colisión con el trigger de finalización y desactiva la bola tras un breve
retraso.
/// </summary>
[ServerCallback]
private void OnTriggerEnter(Collider other)
{
    if (bolaActiva && !estaDesactivando && other.CompareTag("Finish"))
    {
        StartCoroutine(DesactivarBolaConRetraso());
    }
}

/// <summary>
/// Corrutina que desactiva la bola tras un breve retraso y muestra el efecto visual.
/// </summary>
private IEnumerator DesactivarBolaConRetraso()
{
    estaDesactivando = true;
    yield return new WaitForSeconds(0.1f);
    DesactivarBola();
    if (efectoDesactivacion != null)
    {
        NetworkServer.Spawn(Instantiate(efectoDesactivacion, transform.position,
Quaternion.identity));
    }
    yield return new WaitForSeconds(0.5f);
    estaDesactivando = false;
}

/// <summary>
/// Detecta la colisión con el jugador, aplica daño y knockback localmente.
/// </summary>
/// <param name="collision">Datos de la colisión.</param>
```

LETHAL RUN



Izan De La Cruz

David Salvador

Marc Rojano

Brian Orozco

```
private void OnCollisionEnter(Collision collision)
{
    if (bolaActiva && !estaDesactivando)
    {
        if (collision.collider.CompareTag("Player"))
        {
            PlayerHealth playerHealth =
collision.collider.GetComponent<PlayerHealth>();
            if (playerHealth != null)
            {
                playerHealth.TakeDamage(damageAmount);
                ApplyKnockbackLocal(collision.gameObject, collision.contacts[0].point);
            }
        }
    }
}

/// <summary>
/// Aplica un efecto de retroceso (knockback) al jugador golpeado por la bola.
/// </summary>
/// <param name="player">Objeto jugador.</param>
/// <param name="hitPoint">Punto de impacto.</param>
private void ApplyKnockbackLocal(GameObject player, Vector3 hitPoint)
{
    Vector3 knockbackDirection = (player.transform.position - hitPoint).normalized;
    knockbackDirection.y = 0;

    Rigidbody playerRb = player.GetComponent<Rigidbody>();
    if (playerRb != null)
    {
        Vector3 knockbackForceVector = knockbackDirection * knockbackForce;
        playerRb.AddForce(knockbackForceVector, ForceMode.Impulse);
    }
    else
    {
        Vector3 knockbackPosition = player.transform.position + knockbackDirection
* knockbackForce * Time.deltaTime;
        knockbackPosition.y = player.transform.position.y;
        player.transform.position = knockbackPosition;
    }
}

/// <summary>
/// Desactiva la bola y la reinicia en el servidor.

```

```
/// </summary>
[Server]
private void DesactivarBola()
{
    bolaActiva = false;
    ResetearBola();
}

/// <summary>
/// Corrutina que desactiva la bola tras la duración de activación.
/// </summary>
private IEnumerator DesactivarBolaDespuesDeTiempo()
{
    yield return new WaitForSeconds(duracionActivacion);
    if (bolaActiva)
    {
        DesactivarBola();
    }
}
```

Multijugador:

NetworkManager:

Aquest script implementa un NetworkManager personalitzat per a un joc multijugador amb Mirror, gestionant el registre i l'autenticació dels jugadors, l'assignació de punts d'aparició (spawns) i la selecció de skins personalitzats mitjançant una API externa. Sincronitza el nom de l'escena seleccionada i actualitza les dades del lobby a Steamworks. A més, controla el procés de connexió, la creació de jugadors i la lògica d'assignació de rols especials (com trampes) a l'inici de la partida, assegurant una experiència de joc personalitzada i segura per a cada usuari.

```
using System.Collections;
using System.Collections.Generic;
using System.Linq;
using UnityEngine;
using Mirror;
using UnityEngine.Networking;
using System.Text;
using UnityEngine.SceneManagement;
```

using Steamworks;

```
/// <summary>
/// Mensaje de autenticación que envía el cliente al servidor con email y token.
/// </summary>
public struct PlayerAuthMessage : NetworkMessage
{
    public string email;
    public string token;
}

/// <summary>
/// NetworkManager personalizado que gestiona el registro, autenticación, spawns y
skins de los jugadores,
/// así como la sincronización de la escena y datos del lobby en Steamworks.
/// </summary>
public class CustomSpawnNetworkManager : NetworkManager
{
    /// <summary>
    /// Nombre de la escena de lobby.
    /// </summary>
    public string lobbySceneName = "Lobby";

    [Header("Configuración")]
    /// <summary>
    /// Lista de prefabs de jugador disponibles (uno por skin/rol).
    /// </summary>
    public List<GameObject> playerPrefabs;
    /// <summary>
    /// URL de la API para obtener el skin activo del jugador.
    /// </summary>
    public string nodeApiUrl =
"https://lethalrun.cat/mongo-service/skins/active-slot-number";

    [Header("Spawn Points")]
    private List<Transform> normalSpawns = new List<Transform>();
    private Transform trapSpawn;
    private Dictionary<NetworkConnectionToClient, Transform> assignedSpawns =
new Dictionary<NetworkConnectionToClient, Transform>();
    private bool spawnsAssigned = false;
    private List<NetworkConnectionToClient> orderedConnections = new
List<NetworkConnectionToClient>();

    /// <summary>
```

LETHAL RUN



Izan De La Cruz

David Salvador

Marc Rojano

Brian Orozco

```
/// Diccionario para guardar los datos de autenticación por conexión.  
/// </summary>  
private Dictionary<NetworkConnectionToClient, PlayerAuthMessage>  
playerAuthData = new();  
  
/// <summary>  
/// Respuesta de la API de skins.  
/// </summary>  
[System.Serializable]  
public class SkinResponse  
{  
    public int activeSlotNumber;  
}  
  
/// <summary>  
/// Propiedad estática para obtener el nombre de la escena de juego seleccionada.  
/// </summary>  
public static string GameSceneName  
{  
    get  
    {  
        if (SelectedMap.Instance != null &&  
!string.IsNullOrEmpty(SelectedMap.Instance.SelectedMapSceneName))  
            return SelectedMap.Instance.SelectedMapSceneName;  
        return "Mapa1";  
    }  
}  
  
/// <summary>  
/// Al conectar el cliente, envía el email y el token al servidor.  
/// </summary>  
public override void OnClientConnect()  
{  
    base.OnClientConnect();  
    string email = PlayerPrefs.GetString("user.email", "");  
    string token = PlayerPrefs.GetString("user.token", "");  
    var msg = new PlayerAuthMessage { email = email, token = token };  
    NetworkClient.Send(msg);  
}  
  
/// <summary>  
/// Al iniciar el servidor, registra el handler para recibir los datos de autenticación.  
/// </summary>  
public override void OnStartServer()
```

LETHAL RUN



Izan De La Cruz
David Salvador
Marc Rojano
Brian Orozco

```
{  
    base.OnStartServer();  
    NetworkServer.RegisterHandler<PlayerAuthMessage>(OnReceivePlayerAuth);  
}  
  
/// <summary>  
/// Handler que guarda los datos de autenticación recibidos de cada cliente.  
/// </summary>  
private void OnReceivePlayerAuth(NetworkConnectionToClient conn,  
PlayerAuthMessage msg)  
{  
    playerAuthData[conn] = msg;  
}  
  
/// <summary>  
/// Al cambiar de escena en el servidor, actualiza los datos del lobby y asigna  
spawns si corresponde.  
/// </summary>  
public override void OnServerSceneChanged(string sceneName)  
{  
    // Extrae solo el nombre de la escena  
    string sceneShortName = sceneName;  
    if (sceneShortName.Contains("/"))  
        sceneShortName =  
System.IO.Path.GetFileNameWithoutExtension(sceneShortName);  
  
    // Actualiza el dato de la escena en el lobby de Steam si corresponde  
    if (SteamManager.Initialized && LobbyData.currentLobbyID.IsValid())  
    {  
        SteamMatchmaking.SetLobbyData(LobbyData.currentLobbyID, "scene",  
sceneShortName);  
    }  
  
    base.OnServerSceneChanged(sceneName);  
  
    if (sceneShortName == GameSceneName)  
    {  
        CacheSpawns();  
        AssignSpawns();  
    }  
    else  
    {  
        assignedSpawns.Clear();  
        spawnsAssigned = false;  
    }  
}
```

LETHAL RUN



Izan De La Cruz

David Salvador

Marc Rojano

Brian Orozco

```
        orderedConnections.Clear();
    }
}

/// <summary>
/// Solo permite la conexión de jugadores si el servidor está en la escena de lobby.
/// </summary>
public override void OnServerConnect(NetworkConnectionToClient conn)
{
    if (SceneManager.GetActiveScene().name != lobbySceneName)
    {
        conn.Disconnect();
        return;
    }
    base.OnServerConnect(conn);
}

/// <summary>
/// Cachea los puntos de aparición (spawns) normales y de trampa.
/// </summary>
void CacheSpawns()
{
    normalSpawns.Clear();
    trapSpawn = null;
    foreach (Transform t in startPositions)
    {
        if (t.CompareTag("TrapSpawn"))
        {
            trapSpawn = t;
        }
        else
        {
            normalSpawns.Add(t);
        }
    }
}

/// <summary>
/// Asigna los puntos de aparición a cada conexión, eligiendo aleatoriamente un
jugador como "trampa".
/// </summary>
void AssignSpawns()
{
    assignedSpawns.Clear();
```

LETHAL RUN



Izan De La Cruz

David Salvador

Marc Rojano

Brian Orozco

```
spawnsAssigned = false;
orderedConnections = NetworkServer.connections.Values.ToList();

int playerCount = orderedConnections.Count;
if (playerCount == 0)
{
    return;
}

int trapIndex = Random.Range(0, playerCount);
var normalSpawnsLeft = new List<Transform>(normalSpawns);

for (int i = 0; i < playerCount; i++)
{
    var conn = orderedConnections[i];
    if (i == trapIndex)
    {
        assignedSpawns[conn] = trapSpawn;
    }
    else
    {
        if (normalSpawnsLeft.Count == 0)
        {
            assignedSpawns[conn] = null;
        }
        else
        {
            assignedSpawns[conn] = normalSpawnsLeft[0];
            normalSpawnsLeft.RemoveAt(0);
        }
    }
}
spawnsAssigned = true;
}

/// <summary>
/// Añade un jugador al servidor, eligiendo el prefab adecuado según la respuesta
de la API y el spawn asignado.
/// </summary>
public override void OnServerAddPlayer(NetworkConnectionToClient conn)
{
    StartCoroutine(AddPlayerWithSkin(conn));
}
```

LETHAL RUN



Izan De La Cruz

David Salvador

Marc Rojano

Brian Orozco

```
/// <summary>
/// Corrutina que espera los datos de autenticación, consulta la API de skins y crea
el jugador en la posición correcta.
/// </summary>
private IEnumerator AddPlayerWithSkin(NetworkConnectionToClient conn)
{
    int prefabIndex = 0;
    bool isGameScene =
UnityEngine.SceneManagement.SceneManager.GetActiveScene().name ==
GameSceneName;

// Espera a que el cliente haya mandado sus datos (máximo 5 segundos)
float timeout = 5f;
float elapsed = 0f;
while (!playerAuthData.ContainsKey(conn) && elapsed < timeout)
{
    yield return null;
    elapsed += Time.deltaTime;
}

string email = "";
string token = "";

if (playerAuthData.TryGetValue(conn, out var authMsg))
{
    email = authMsg.email;
    token = authMsg.token;
}

if (isGameScene && spawnsAssigned && assignedSpawns.ContainsKey(conn))
{
    if (!string.IsNullOrEmpty(email) && !string.IsNullOrEmpty(token))
    {
        string jsonBody = $"\"{{\"email\":\"{email}\",\"token\":\"{token}\",\"prefabIndex\":{prefabIndex}}}";
        byte[] bodyRaw = Encoding.UTF8.GetBytes(jsonBody);
        using (UnityWebRequest request = new UnityWebRequest(nodeApiUrl,
"GET"))
        {
            request.uploadHandler = new UploadHandlerRaw(bodyRaw);
            request.downloadHandler = new DownloadHandlerBuffer();
            request.SetRequestHeader("Content-Type", "application/json");
            request.SetRequestHeader("Authorization", $"Bearer {token}");
            yield return request.SendWebRequest();
        }
    }
}
```

LETHAL RUN



Izan De La Cruz

David Salvador

Marc Rojano

Brian Orozco

```
if (request.result == UnityWebRequest.Result.Success)
{
    try
    {
        SkinResponse response =
JsonUtility.FromJson<SkinResponse>(request.downloadHandler.text);
        prefabIndex = Mathf.Clamp(response.activeSlotNumber, 0,
playerPrefabs.Count - 1);
    }
    catch
    {
        // Si falla el parseo, se usa el prefab por defecto (0)
    }
}
// Si falla la petición, se usa el prefab por defecto (0)
}

Transform spawn = assignedSpawns[conn];
GameObject prefab = playerPrefabs[Mathf.Clamp(prefabIndex, 0,
playerPrefabs.Count - 1)];
GameObject player = Instantiate(prefab, spawn != null ? spawn.position :
Vector3.zero, spawn != null ? spawn.rotation : Quaternion.identity);
player.name = $"{prefab.name} [connId={conn.connectionId}]";
NetworkServer.AddPlayerForConnection(conn, player);
}
else
{
    GameObject prefab = playerPrefabs != null && playerPrefabs.Count > 0 ?
playerPrefabs[0] : playerPrefab;
    Transform spawn = GetStartPosition();
    GameObject player = Instantiate(prefab, spawn != null ? spawn.position :
Vector3.zero, spawn != null ? spawn.rotation : Quaternion.identity);
    player.name = $"{prefab.name} [connId={conn.connectionId}]";
    NetworkServer.AddPlayerForConnection(conn, player);
}
}
```