

# **Informe de recopilación de Scripts (Proyecto Arkmed)**

Santiago Castañeda Pérez

Universidad Tecnológica de Pereira  
Computación Gráfica  
Profesor: Andrés Felipe Ramírez

22 de mayo de 2025

# Índice

1. AudioManager.cs	4
2. MainMenuManager.cs	4
3. IntroManager.cs	5
4. FinalManager.cs	8
5. Credits.cs	10
6. ZombieAI.cs	11
7. TextoParpadeante.cs	16
8. PoliceSiren.cs	16
9. PauseManager.cs	17
10.DeathMenu.cs	18
11.PlayerMovement.cs	20
12.PlayerInventory.cs	32
13.PlayerDoorRaycast.cs	32
14.DoorScript.cs	34
15.KeyPickup.cs	35
16.ItemPickup.cs	36
17.PlayerSystemRaycast.cs	36
18.SystemInteractionMenu.cs	37
19.PlayerDocumentRaycast.cs	39
20.PlayerDocumentInteraction.cs	40
21.InteractionZone.cs	41
22.DocumentReader.cs	42
23.BoosFinalAI.cs	42
24.BulletScript.cs	46
25.FloatingText.cs	47
26.CameraFollow.cs	48

<b>27.GameManager.cs</b>	<b>49</b>
<b>28.PlayerData.cs</b>	<b>49</b>
<b>29.DialogoManager.cs</b>	<b>50</b>
<b>30.MissingScripts.cs</b>	<b>51</b>

## 1. AudioManager.cs

```
1 using UnityEngine;
2
3 public class AudioManager : MonoBehaviour
4 {
5     public static AudioManager instancia;
6
7     public AudioSource Musica_Fondo;
8     public AudioClip Sonido_Click;
9
10    private void Awake()
11    {
12        if (instancia == null)
13        {
14            instancia = this;
15            DontDestroyOnLoad(gameObject);
16        }
17        else
18        {
19            Destroy(gameObject);
20        }
21    }
22
23    public void ReproducirClick()
24    {
25        AudioSource.PlayClipAtPoint(Sonido_Click, Camera.main.transform.
position);
26    }
27
28    public void DetenerMusica()
29    {
30        if (Musica_Fondo != null)
31            Musica_Fondo.Stop();
32    }
33 }
```

Listing 1: AudioManager.cs

## 2. MainMenuManager.cs

```
1 using UnityEngine;
2 using UnityEngine.SceneManagement;
3
4 public class MainMenuManager : MonoBehaviour
5 {
6     public GameObject PanelControles; // Panel donde est n los
controles
7     public GameObject Jugar;          // Bot n "Jugar"
8     public GameObject Salir;           // Bot n "Salir"
9     public GameObject Controles;       // Bot n "Controles"
10    public GameObject Cerrar;           // Bot n "Cerrar" en el Panel de
Controles
11
12    // Funci n para iniciar el juego
13    public void JugarJuego()
14    {
```

```

15         if (AudioManager.instancia != null)
16         {
17             AudioManager.instancia.DetenerMusica();
18         }
19
20         PlayerPrefs.DeleteAll();
21
22         SceneManager.LoadScene("IntroScene");
23     }
24
25     // Funci n para mostrar los controles
26     public void MostrarControles()
27     {
28         PanelControles.SetActive(true); // Activa el panel de controles
29         Jugar.SetActive(false);         // Desactiva el bot n Jugar
30         Salir.SetActive(false);          // Desactiva el bot n Salir
31         Controles.SetActive(false);      // Desactiva el bot n
32         Controles
33         Cerrar.SetActive(true);           // Activa el bot n Cerrar
34     }
35
36     // Funci n para ocultar los controles y regresar al men principal
37     public void OcultarControles()
38     {
39         PanelControles.SetActive(false); // Desactiva el panel de
40         controles
41         Jugar.SetActive(true);            // Activa el bot n Jugar
42         Salir.SetActive(true);            // Activa el bot n Salir
43         Controles.SetActive(true);        // Activa el bot n Controles
44         Cerrar.SetActive(false);          // Desactiva el bot n Cerrar
45     }
46
47     public void SalirDelJuego()
48     {
49         Debug.Log("Saliendo del juego...");
50         Application.Quit();
51     }
52
53     // M todo Update para desactivar la tecla Escape
54     private void Update()
55     {
56         // No hacemos nada si se presiona Escape
57         if (Input.GetKeyDown(KeyCode.Escape))
58         {
59             // Escape est desactivado
60         }
61     }
62 }

```

Listing 2: MainMenuManager.cs

### 3. IntroManager.cs

```

1     using System.Collections;
2     using UnityEngine;
3     using UnityEngine.SceneManagement;
4     using TMPPro;
5     using UnityEngine.UI;

```

```

6
7 public class IntroManager : MonoBehaviour
8 {
9     [Header("UI")]
10    public TextMeshProUGUI textoUI;
11    public Image panelFondo; // para fade
12
13    [Header("Texto")]
14    [TextArea(3, 10)]
15    public string[] parrafos;
16    public float velocidadTipeo = 0.03f;
17
18    [Header("Audio")]
19    public AudioSource audioSource;
20    public AudioClip sonidoLetra;
21
22    [Header("Fade")]
23    public float duracionFade = 0.5f;
24
25    private int indiceParrafo = 0;
26    private bool escribiendo = false;
27    private bool puedeContinuar = false;
28
29    void Start()
30    {
31        textoUI.text = "";
32
33        // Inicializar datos del jugador
34        if (PlayerPrefs.HasKey("Vida"))
35        {
36            Debug.Log("Cargando datos del jugador desde PlayerPrefs en
IntroManager.");
37            PlayerPrefs.GetInt("Vida"); // Vida del jugador
38            PlayerPrefs.GetInt("Municion"); // Munici n total
39            PlayerPrefs.GetInt("Puntuacion"); // Puntuaci n
40            PlayerPrefs.GetInt("Botiquines"); // Botiquines
41        }
42        else
43        {
44            Debug.Log("No se encontraron datos guardados. Inicializando
valores predeterminados en IntroManager.");
45            PlayerPrefs.SetInt("Vida", 100); // Vida m xima
46            PlayerPrefs.SetInt("Municion", 30); // Munici n m xima
47            PlayerPrefs.SetInt("Puntuacion", 0); // Puntuaci n inicial
48            PlayerPrefs.SetInt("Botiquines", 0); // Sin botiquines
49            PlayerPrefs.Save();
50        }
51
52        StartCoroutine(MostrarParrafo()); // Mostrar el texto de
introducci n
53    }
54
55
56
57    void Update()
58    {
59        if (!escribiendo && puedeContinuar && (Input.anyKeyDown || Input
.GetMouseButtonDown(0)))

```

```

60     {
61         if (indiceParrafo < parrafos.Length)
62         {
63             StartCoroutine(MostrarParrafo());
64         }
65         else
66         {
67             SceneManager.LoadScene("GameplayScene 1");
68         }
69     }
70 }
71
72 IEnumerator MostrarParrafo()
73 {
74     escribiendo = true;
75     puedeContinuar = false;
76     textoUI.text = "";
77
78     // Inicia sonido largo del p rrafo
79     if (audioSource != null && sonidoLetra != null)
80     {
81         audioSource.Stop();
82         audioSource.clip = sonidoLetra;
83         audioSource.Play();
84     }
85
86     yield return StartCoroutine(FadeIn());
87
88     string parrafo = parrafos[indiceParrafo];
89     foreach (char letra in parrafo)
90     {
91         textoUI.text += letra;
92         yield return new WaitForSeconds(velocidadTipeo);
93     }
94
95     // Detiene el sonido al terminar de escribir
96     if (audioSource != null)
97     {
98         audioSource.Stop();
99     }
100
101     yield return new WaitForSeconds(0.5f);
102     indiceParrafo++;
103     puedeContinuar = true;
104     escribiendo = false;
105 }
106
107
108
109 IEnumerator FadeIn()
110 {
111     float tiempo = 0;
112     Color color = panelFondo.color;
113
114     while (tiempo < duracionFade)
115     {
116         tiempo += Time.deltaTime;
117         float alpha = Mathf.Lerp(0f, 1f, tiempo / duracionFade);

```

```

118         panelFondo.color = new Color(color.r, color.g, color.b,
alpha);
119         yield return null;
120     }
121
122     panelFondo.color = new Color(color.r, color.g, color.b, 1f);
123 }
124 }

```

Listing 3: IntroManager.cs

## 4. FinalManager.cs

```

1 using System.Collections;
2 using UnityEngine;
3 using UnityEngine.SceneManagement;
4 using TMPro;
5 using UnityEngine.UI;
6
7 public class FinalManager : MonoBehaviour
8 {
9     [Header("UI")]
10    public TextMeshProUGUI textoUI;
11    public Image panelFondo; // para fade
12
13    [Header("Texto")]
14    [TextArea(3, 10)]
15    public string[] parrafos;
16    public float velocidadTipeo = 0.03f;
17
18    [Header("Audio")]
19    public AudioSource audioSource;
20    public AudioClip sonidoLetra;
21
22    [Header("Fade")]
23    public float duracionFade = 0.5f;
24
25    private int indiceParrafo = 0;
26    private bool escribiendo = false;
27    private bool puedeContinuar = false;
28
29    void Start()
30    {
31        textoUI.text = "";
32        StartCoroutine(MostrarParrafo());
33    }
34
35    void Update()
36    {
37        if (!escribiendo && puedeContinuar && (Input.anyKeyDown || Input
.GetMouseButtonDown(0)))
38        {
39            if (indiceParrafo < parrafos.Length)
40            {
41                StartCoroutine(MostrarParrafo());
42            }
43            else
44            {

```



```

45         SceneManager.LoadScene("Credits");
46     }
47 }
48 }
49
50 IEnumerator MostrarParrafo()
51 {
52     escribiendo = true;
53     puedeContinuar = false;
54     textoUI.text = "";
55
56     // Sonido de tipeo
57     if (audioSource != null && sonidoLetra != null)
58     {
59         audioSource.Stop();
60         audioSource.clip = sonidoLetra;
61         audioSource.Play();
62     }
63
64     yield return StartCoroutine(FadeIn());
65
66     string parrafo = parrafos[indiceParrafo];
67     foreach (char letra in parrafo)
68     {
69         textoUI.text += letra;
70         yield return new WaitForSeconds(velocidadTipeo);
71     }
72
73     if (audioSource != null)
74     {
75         audioSource.Stop();
76     }
77
78     yield return new WaitForSeconds(0.5f);
79     indiceParrafo++;
80     puedeContinuar = true;
81     escribiendo = false;
82 }
83
84 IEnumerator FadeIn()
85 {
86     float tiempo = 0;
87     Color color = panelFondo.color;
88
89     while (tiempo < duracionFade)
90     {
91         tiempo += Time.deltaTime;
92         float alpha = Mathf.Lerp(0f, 1f, tiempo / duracionFade);
93         panelFondo.color = new Color(color.r, color.g, color.b,
alpha);
94         yield return null;
95     }
96
97     panelFondo.color = new Color(color.r, color.g, color.b, 1f);
98 }
99 }

```

Listing 4: FinalManager.cs

## 5. Credits.cs

```
1 using System.Collections;
2 using UnityEngine;
3 using UnityEngine.SceneManagement;
4 using TMPro;
5 using UnityEngine.UI;
6
7 public class Credits : MonoBehaviour
8 {
9     [Header("UI")]
10    public TextMeshProUGUI textoUI;
11    public Image panelFondo; // para fade
12
13    [Header("Texto")]
14    [TextArea(3, 10)]
15    public string[] parrafos;
16    public float velocidadTipeo = 0.03f;
17
18    [Header("Audio")]
19    public AudioSource audioSource;
20    public AudioClip sonidoLetra;
21
22    [Header("Fade")]
23    public float duracionFade = 0.5f;
24
25    private int indiceParrafo = 0;
26    private bool escribiendo = false;
27    private bool puedeContinuar = false;
28
29    void Start()
30    {
31        textoUI.text = "";
32        StartCoroutine(MostrarParrafo());
33    }
34
35    void Update()
36    {
37        if (!escribiendo && puedeContinuar && (Input.anyKeyDown || Input
38        .GetMouseButtonDown(0)))
39        {
40            if (indiceParrafo < parrafos.Length)
41            {
42                StartCoroutine(MostrarParrafo());
43            }
44            else
45            {
46                SceneManager.LoadScene("MainMenu");
47            }
48        }
49
50        IEnumerator MostrarParrafo()
51        {
52            escribiendo = true;
53            puedeContinuar = false;
54            textoUI.text = "";
55
```

```

56 // Sonido de tipeo
57 if (audioSource != null && sonidoLetra != null)
58 {
59     audioSource.Stop();
60     audioSource.clip = sonidoLetra;
61     audioSource.Play();
62 }
63
64 yield return StartCoroutine(FadeIn());
65
66 string parrafo = parrafos[indiceParrafo];
67 foreach (char letra in parrafo)
68 {
69     textoUI.text += letra;
70     yield return new WaitForSeconds(velocidadTipeo);
71 }
72
73 if (audioSource != null)
74 {
75     audioSource.Stop();
76 }
77
78 yield return new WaitForSeconds(0.5f);
79 indiceParrafo++;
80 puedeContinuar = true;
81 escribiendo = false;
82 }
83
84 IEnumerator FadeIn()
85 {
86     float tiempo = 0;
87     Color color = panelFondo.color;
88
89     while (tiempo < duracionFade)
90     {
91         tiempo += Time.deltaTime;
92         float alpha = Mathf.Lerp(0f, 1f, tiempo / duracionFade);
93         panelFondo.color = new Color(color.r, color.g, color.b,
alpha);
94         yield return null;
95     }
96
97     panelFondo.color = new Color(color.r, color.g, color.b, 1f);
98 }
99 }

```

Listing 5: Credits.cs

## 6. ZombieAI.cs

```

1 using UnityEngine;
2 using UnityEngine.UI;
3
4 public class ZombieAI : MonoBehaviour
5 {
6     public float speed = 2f;
7     public float attackRange = 1f;
8     public float attackCooldown = 1f;

```

```

9      public float health = 100f;
10     public float stunDuration = 1f;
11
12     public AudioClip growlSound;
13     public AudioClip hitSound;
14     public AudioClip deathSound;
15
16     public GameObject healthBarPrefab;
17     private Image healthBarFill;
18     private GameObject healthBarInstance;
19     private Vector3 healthBarOffset = new Vector3(0, 1.5f, 0); // Ajusta
    la altura
20
21     private Transform player;
22     private Animator animator;
23     private Rigidbody2D rb;
24     private AudioSource audioSource;
25
26     private float lastAttackTime;
27     private bool isStunned = false;
28     private bool isDead = false;
29
30     [SerializeField] private GameObject medkitPrefab;
31     [SerializeField] private GameObject ammoPrefab;
32     [SerializeField] private GameObject keyPrefab; // Prefab de la llave
33
34     void Start()
35     {
36         if (healthBarPrefab != null)
37         {
38             // Instanciar la barra de vida y obtener el componente de
    relleno
39             healthBarInstance = Instantiate(healthBarPrefab, transform.
    position + healthBarOffset, Quaternion.identity);
40             healthBarFill = healthBarInstance.GetComponentInChildren<
    Image>();
41         }
42
43         player = GameObject.FindGameObjectWithTag("Player")?.transform;
44         animator = GetComponent<Animator>();
45         rb = GetComponent<Rigidbody2D>();
46         audioSource = GetComponent<AudioSource>();
47
48         PlayGrowl();
49     }
50
51     void Update()
52     {
53         if (player == null || isDead) return;
54
55         if (isStunned)
56         {
57             animator.SetBool("isRunning", false);
58             return;
59         }
60
61         float distance = Vector2.Distance(transform.position, player.
    position);

```

```

62     bool isAttacking = distance <= attackRange;
63
64     animator.SetBool("isAttacking", isAttacking);
65     animator.SetBool("isRunning", !isAttacking);
66
67     if (!isAttacking)
68     {
69         Vector2 direction = (player.position - transform.position).
normalized;
70         transform.position += (Vector3)direction * speed * Time.
deltaTime;
71
72         if (direction.x > 0)
73             transform.localScale = new Vector3(1, 1, 1);
74         else if (direction.x < 0)
75             transform.localScale = new Vector3(-1, 1, 1);
76     }
77     else
78     {
79         if (Time.time - lastAttackTime >= attackCooldown)
80         {
81             Attack();
82             lastAttackTime = Time.time;
83         }
84     }
85
86     // Seguir al zombie
87     if (healthBarInstance != null)
88     {
89         healthBarInstance.transform.position = transform.position +
healthBarOffset;
90     }
91 }
92
93 public void TakeDamage(float damage)
94 {
95     if (isDead) return;
96
97     health -= damage;
98     PlaySound(hitSound);
99
100     if (health <= 0)
101     {
102         Die();
103     }
104     else
105     {
106         if (!isStunned)
107         {
108             isStunned = true;
109             Invoke("ResetStun", stunDuration);
110         }
111     }
112
113     if (healthBarFill != null)
114     {
115         float percentage = health / 100f;
116         healthBarFill.fillAmount = percentage;

```

```

117         if (percentage > 0.6f)
118             healthBarFill.color = Color.green;
119         else if (percentage > 0.3f)
120             healthBarFill.color = Color.yellow;
121         else
122             healthBarFill.color = Color.red;
123     }
124 }
125
126
127 void DropLoot()
128 {
129     Vector3 dropPosition = transform.position + new Vector3(Random.
130 Range(-0.3f, 0.3f), -0.3f, 0);
131
132     if (Random.value < 0.5f)
133         Instantiate(medkitPrefab, dropPosition, Quaternion.identity)
134 ;
135     else
136         Instantiate(ammoPrefab, dropPosition, Quaternion.identity);
137 }
138
139 void DropKey()
140 {
141     if (keyPrefab != null)
142     {
143         if (GameObject.FindGameObjectWithTag("Key") == null)
144         {
145             Vector3 dropPosition = transform.position + new Vector3
146 (0f, -0.3f, 0);
147             Instantiate(keyPrefab, dropPosition, Quaternion.identity
148 );
149             Debug.Log("Llave soltada por el zombie.");
150         }
151         else
152         {
153             Debug.Log("Ya hay una llave en la escena, no se soltar
154 otra.");
155         }
156     }
157     else
158     {
159         Debug.LogWarning("El prefab de la llave no est asignado en
160 el inspector.");
161     }
162 }
163
164 void Die()
165 {
166     isDead = true;
167     animator.SetTrigger("DieTrigger");
168     PlaySound(deathSound);
169
170     // Suelta bot n y llave
171     DropLoot();
172     DropKey();

```

```

169
170     // A adir puntuaci n al jugador
171     if (player != null)
172     {
173         PlayerMovement playerScript = player.GetComponent<
PlayerMovement>();
174         if (playerScript != null)
175         {
176             int points = Random.value < 0.2f ? 10 : 5;
177             string popup = points == 10 ? "    Crtico    +10!" : $"+{
points}";
178             playerScript.AddScore(points, transform.position, popup)
;
179         }
180     }
181
182     Destroy(gameObject, 0.5f);
183     Destroy(healthBarInstance);
184 }
185
186 void ResetStun()
187 {
188     isStunned = false;
189     animator.SetBool("isStunned", false);
190     rb.bodyType = RigidbodyType2D.Dynamic;
191 }
192
193 void Attack()
194 {
195     if (player != null)
196     {
197         PlayerMovement playerScript = player.GetComponent<
PlayerMovement>();
198         if (playerScript != null)
199         {
200             playerScript.TakeDamage(10);
201         }
202     }
203 }
204
205 void PlaySound(AudioClip clip)
206 {
207     if (audioSource != null && clip != null)
208     {
209         audioSource.PlayOneShot(clip);
210     }
211 }
212
213 void PlayGrowl()
214 {
215     if (audioSource != null && growlSound != null)
216     {
217         audioSource.loop = true;
218         audioSource.clip = growlSound;
219         audioSource.Play();
220     }
221 }

```

Listing 6: ZombieAI.cs

## 7. TextoParpadeante.cs

```

1 using UnityEngine;
2 using TMPro;
3
4 public class TextoParpadeante : MonoBehaviour
5 {
6     public float velocidadParpadeo = 1f;
7     private TextMeshProUGUI textoUI;
8     private Color colorOriginal;
9
10    void Start()
11    {
12        textoUI = GetComponent<TextMeshProUGUI>();
13        colorOriginal = textoUI.color;
14    }
15
16    void Update()
17    {
18        float alfa = Mathf.Abs(Mathf.Sin(Time.time * velocidadParpadeo))
19        ;
20        textoUI.color = new Color(colorOriginal.r, colorOriginal.g,
21        colorOriginal.b, alfa);
22    }
23 }

```

Listing 7: TextoParpadeante.cs

## 8. PoliceSiren.cs

```

1 using UnityEngine;
2
3 public class PoliceSiren : MonoBehaviour
4 {
5     public GameObject redLight;
6     public GameObject blueLight;
7     public float blinkInterval = 0.3f;
8
9     private float timer;
10    private bool isRedActive = true;
11
12    void Start()
13    {
14        redLight.SetActive(true);
15        blueLight.SetActive(false);
16    }
17
18    void Update()
19    {
20        timer += Time.deltaTime;
21        if (timer >= blinkInterval)
22        {
23            timer = 0f;

```



```

24         isRedActive = !isRedActive;
25         redLight.SetActive(isRedActive);
26         blueLight.SetActive(!isRedActive);
27     }
28 }
29 }

```

Listing 8: PoliceSiren.cs

## 9. PauseManager.cs

```

1  using UnityEngine;
2  using UnityEngine.SceneManagement;
3  using UnityEngine.UI;
4
5  public class PausaManager : MonoBehaviour
6  {
7      public GameObject panelPausa;
8      public Slider healthBar;
9
10     private bool juegoPausado = false;
11
12     void Update()
13     {
14         if (Input.GetKeyDown(KeyCode.Escape))
15         {
16             if (juegoPausado)
17                 Reanudar();
18             else
19                 Pausar();
20         }
21     }
22
23     public void Pausar()
24     {
25         panelPausa.SetActive(true);
26         Time.timeScale = 0f;
27         juegoPausado = true;
28
29         // Ocultar todo lo que contenga Player
30         PlayerMovement player = Object.FindFirstObjectByType<
PlayerMovement>();
31         if (player != null)
32         {
33             if (player.medkitText != null)
34                 player.PanelBotiquines.gameObject.SetActive(false);
35
36             if (player.ammoText != null)
37                 player.PanelMunicion.gameObject.SetActive(false);
38
39             if (player.healthBar != null)
40                 player.healthBar.gameObject.SetActive(false); // Ocultar
healthBar
41
42             if (player.keyPanel != null)
43                 player.keyPanel.gameObject.SetActive(false); // Ocultar
keyPanel
44         }

```

```

45     }
46
47     public void Reanudar()
48     {
49         panelPausa.SetActive(false);
50         Time.timeScale = 1f;
51         juegoPausado = false;
52
53         // Mostrar todo lo que contenga Player
54         PlayerMovement player = Object.FindFirstObjectByType<
PlayerMovement>();
55         if (player != null)
56         {
57             if (player.medkitText != null)
58                 player.PanelBotiquines.gameObject.SetActive(true);
59
60             if (player.ammoText != null)
61                 player.PanelMunicion.gameObject.SetActive(true);
62
63             if (player.healthBar != null)
64                 player.healthBar.gameObject.SetActive(true); // Mostrar
healthBar
65
66             player.UpdateKeyPanel();
67         }
68     }
69
70     public void Salir()
71     {
72         panelPausa.SetActive(false);
73         Time.timeScale = 1f;
74         juegoPausado = false;
75
76         // Destruir el jugador y todos sus hijos
77         PlayerMovement player = Object.FindFirstObjectByType<
PlayerMovement>();
78         if (player != null)
79         {
80             Destroy(player.gameObject); // Destruye el jugador y todos
sus hijos
81         }
82
83         SceneManager.LoadScene("MainMenu"); // Cambia a la escena del
men principal
84     }
85
86 }

```

Listing 9: PauseManager.cs

## 10. DeathMenu.cs

```

1 using UnityEngine;
2 using UnityEngine.SceneManagement;
3
4 public class DeathMenu : MonoBehaviour
5 {
6     private static DeathMenu instance;

```

```

7
8 private void Awake()
9 {
10     if (instance != null && instance != this)
11     {
12         Destroy(gameObject);
13         return;
14     }
15
16     instance = this;
17     DontDestroyOnLoad(gameObject);
18 }
19
20 private void OnEnable()
21 {
22     SceneManager.sceneLoaded += OnSceneLoaded;
23 }
24
25 private void OnDisable()
26 {
27     SceneManager.sceneLoaded -= OnSceneLoaded;
28 }
29
30 private void OnSceneLoaded(Scene scene, LoadSceneMode mode)
31 {
32     // Si persiste en una escena nueva, se destruye autom ticamente
33     if (scene.name != "GameplayScene 1" && scene.name != "
GameplayScene 2" && scene.name != "GameplayScene 3")
34     {
35         Destroy(gameObject);
36     }
37 }
38
39 public void RestartLevel()
40 {
41     Time.timeScale = 1f;
42
43     // Reiniciar el estado del jugador
44     PlayerMovement player = FindFirstObjectByType<PlayerMovement>();
45     if (player != null)
46     {
47         Destroy(player.gameObject); // Destruye el jugador y todos
sus hijos
48     }
49
50     SceneManager.LoadScene("GameplayScene 1"); // Carga la escena
deseada
51 }
52
53 public void GoToMainMenu()
54 {
55     Time.timeScale = 1f;
56
57     // Buscar y destruir el objeto PlayerMovement
58     PlayerMovement player = FindFirstObjectByType<PlayerMovement>();
59     if (player != null)
60     {
61         Destroy(player.gameObject); // Destruye el jugador y todos

```

```

62     sus hijos
63     }
64     SceneManager.LoadScene("MainMenu");
65 }
66 }

```

Listing 10: DeathMenu.cs

## 11. PlayerMovement.cs

```

1  using UnityEngine;
2  using TMPro;
3  using System.Collections;
4  using UnityEngine.SceneManagement;
5
6  public class PlayerMovement : MonoBehaviour
7  {
8      public float speed = 5f;
9      public float jumpForce = 7f;
10     public Animator animator;
11     public GameObject bulletPrefab;
12     public Transform firePoint;
13     public float bulletSpeed = 10f;
14     public TextMeshProUGUI ammoText;
15     public AudioClip casingSound; // Nuevo sonido de casquillo
16     public AudioClip reloadSound;
17     public int maxAmmo = 30; // Capacidad m xima de balas en total
18     // (valor fijo para referencia)
19     public int clipSize = 5; // Balas por recarga
20     private int currentAmmo; // Balas en el cargador (izquierda del
21     //)
22     private int totalAmmo; // Balas en reserva (derecha del /)
23     public GameObject PanelBotiquines; // UI del jugador (opcional,
24     puedes eliminarlo si no lo necesitas)
25     public GameObject PanelMunicion; // UI del jugador (opcional,
26     puedes eliminarlo si no lo necesitas)
27     private static PlayerMovement instance;
28     public GameObject keyPanel; // UI opcional para mostrar la vida (
29     puedes eliminarlo si no lo necesitas)
30     // Removed unused field 'escenaCargando' as it was not being used in
31     the code.
32
33     public int medkitCount = 0;
34     public TextMeshProUGUI medkitText; // Muestra cu ntos botiquines
35     tienes
36
37     public GameObject pickupPrompt; // UI de Presiona E para
38     recoger
39     [HideInInspector] public GameObject itemNearby;
40     public AudioClip pickupSound;
41
42     public TextMeshProUGUI scoreText;
43     public GameObject scorePopupPrefab; // Prefab con un texto que flota
44
45     private int score = 0;
46
47     // Sonidos

```

```

40 public AudioClip shootSound;
41 public AudioClip jumpSound;
42 public AudioClip stepSound;
43
44 private Rigidbody2D rb;
45 private SpriteRenderer spriteRenderer;
46 private AudioSource audioSource;
47 private AudioSource stepAudioSource;
48
49 // Suelo
50 private bool isGrounded;
51 public Transform groundCheck;
52 public float groundCheckRadius = 0.2f;
53 public LayerMask groundLayer;
54
55 // Control de disparos
56 private float lastShootTime = 0f; // Tiempo del ltimo disparo
57 private int shotsSinceLastReload = 0;
58 public float shootCooldown = 0.2f; // Tiempo de recarga entre
disparos
59 private bool isReloading = false;
60
61 public int maxHealth = 100;
62 private int currentHealth;
63
64 public AudioClip hurtSound;
65 public UnityEngine.UI.Slider healthBar;
66 public AudioClip deathSound;
67 public GameObject deathEffect;
68 public TextMeshProUGUI healthText; // UI opcional para mostrar la
vida
69
70 public GameObject deathMenuCanvas;
71
72 public string sceneToLoad; // Nombre de la escena a cargar
73
74 void Start()
75 {
76     if (PlayerPrefs.HasKey("Vida"))
77     {
78         Debug.Log("Cargando datos del jugador desde PlayerPrefs.");
79         currentHealth = PlayerPrefs.GetInt("Vida");
80         totalAmmo = PlayerPrefs.GetInt("Municion");
81         score = PlayerPrefs.GetInt("Puntuacion");
82         medkitCount = PlayerPrefs.GetInt("Botiquines");
83         currentAmmo = clipSize;
84
85         UpdateKeyPanel(); // Inicializa el estado del panel
86
87         UpdateUI();
88         UpdateHealthUI();
89     }
90     else
91     {
92         Debug.Log("No se encontraron datos guardados. Inicializando
valores predeterminados.");
93         currentHealth = maxHealth;
94         currentAmmo = clipSize;

```

```

95         totalAmmo = maxAmmo - clipSize;
96         medkitCount = 0;
97         score = 0;
98
99         UpdateKeyPanel(); // Inicializa el estado del panel
100
101         UpdateUI();
102         UpdateHealthUI();
103     }
104
105     rb = GetComponent<Rigidbody2D>();
106     spriteRenderer = GetComponent<SpriteRenderer>();
107     animator = GetComponent<Animator>();
108
109     audioSource = GetComponent<AudioSource>();
110
111     stepAudioSource = gameObject.AddComponent<AudioSource>();
112     stepAudioSource.clip = stepSound;
113     stepAudioSource.loop = true;
114     stepAudioSource.playOnAwake = false;
115     stepAudioSource.volume = 0.5f;
116
117     if (healthBar != null)
118     {
119         healthBar.maxValue = maxHealth;
120         healthBar.value = currentHealth;
121     }
122
123     if (healthText != null)
124         healthText.text = $"Vida: {currentHealth}";
125
126     UpdateKeyPanel(); // Inicializa el estado del panel
127
128     UpdateUI();
129
130     LoadPlayerData();
131 }
132
133
134 void Update()
135 {
136     if (Time.timeScale == 0f) return;
137
138     // Verificar si est tocando suelo
139     isGrounded = Physics2D.OverlapCircle(groundCheck.position,
groundCheckRadius, groundLayer);
140
141     float move = 0f;
142     if (Input.GetKey(KeyCode.D))
143         move = 1f;
144
145     else if (Input.GetKey(KeyCode.A))
146         move = -1f;
147
148     rb.linearVelocity = new Vector2(move * speed, rb.linearVelocity.
y);
149
150     // Voltear sprite

```

```

151     if (move > 0)
152         spriteRenderer.flipX = false;
153
154     else if (move < 0)
155         spriteRenderer.flipX = true;
156
157     // Animaci n
158     animator.SetFloat("Speed", Mathf.Abs(move));
159
160     // Sonido de pasos: solo si moviendo y tocando el suelo
161     if (Mathf.Abs(move) > 0.1f && isGrounded)
162     {
163         if (!stepAudioSource.isPlaying)
164             stepAudioSource.Play();
165     }
166
167     else
168     {
169         if (stepAudioSource.isPlaying)
170             stepAudioSource.Stop();
171     }
172
173     // Saltar
174     if (isGrounded && Input.GetKeyDown(KeyCode.Space))
175     {
176         rb.linearVelocity = new Vector2(rb.linearVelocity.x,
jumpForce);
177         PlaySound(jumpSound);
178     }
179
180     if (!isReloading && Input.GetKeyDown(KeyCode.F) && Mathf.Abs(rb.
linearVelocity.x) < 0.1f && currentAmmo > 0)
181     {
182         if (Time.time - lastShootTime >= shootCooldown)
183         {
184             Shoot();
185             animator.SetTrigger("Shoot");
186             PlaySound(shootSound);
187             currentAmmo--;
188             lastShootTime = Time.time;
189
190             shotsSinceLastReload++;
191
192             if (shotsSinceLastReload >= clipSize)
193             {
194                 StartCoroutine(ReloadRoutine());
195             }
196         }
197     }
198
199     if (Input.GetKeyDown(KeyCode.G))
200     {
201         SavePlayerData();
202         Debug.Log("Datos guardados");
203     }
204
205     if (ammoText != null)
206         ammoText.text = $"{{currentAmmo}}/{{totalAmmo}}";

```

```

207
208     float minX = -10f;
209     float maxX = 10f;
210
211     Vector3 clampedPosition = transform.position;
212     clampedPosition.x = Mathf.Clamp(clampedPosition.x, minX, maxX);
213     transform.position = clampedPosition;
214
215     if (Input.GetKeyDown(KeyCode.H))
216     {
217         TakeDamage(10); // Quita 10 de vida al presionar H (solo
para pruebas)
218     }
219
220     // Recargar manualmente con R
221     if (Input.GetKeyDown(KeyCode.R) && !isReloading && currentAmmo <
clipSize && totalAmmo > 0)
222     {
223         StartCoroutine(ReloadRoutine());
224     }
225
226     if (Input.GetKeyDown(KeyCode.Tab))
227     {
228         if (medkitCount > 0 && currentHealth < maxHealth)
229         {
230             currentHealth = Mathf.Min(currentHealth + 30, maxHealth)
; // Cura 30 puntos
231             medkitCount--;
232             UpdateHealthUI();
233             UpdateUI();
234         }
235     }
236
237     // Verificar si el jugador presiona la tecla 'E' para recoger el
tem
238     if (Input.GetKeyDown(KeyCode.E) && itemNearby != null)
239     {
240         PickupItem(itemNearby); // Llama al m todo para destruir
el tem
241     }
242 }
243
244 void PickupItem(GameObject item)
245 {
246     string tag = item.tag;
247
248     if (tag == "Medkit")
249     {
250         medkitCount++;
251         Destroy(item); // Destruir el tem completo
252     }
253     else if (tag == "Ammo")
254     {
255         totalAmmo += 2;
256         Destroy(item);
257     }
258
259     else if (tag == "Key")

```



```

260     {
261         PlayerInventory inventory = GetComponent<PlayerInventory>();
262         if (inventory != null)
263         {
264             inventory.llaves++;
265             Destroy(item);
266             UpdateKeyPanel(); // Actualiza el estado del panel
267         }
268     }
269     // Reproducir el sonido de recolecci n
270     AudioSource.PlayClipAtPoint(pickupSound, transform.position, 3.0
f);
271
272     UpdateUI();
273 }
274
275 public void UpdateKeyPanel()
276 {
277     PlayerInventory inventory = GetComponent<PlayerInventory>();
278     if (inventory != null && keyPanel != null)
279     {
280         keyPanel.SetActive(inventory.llaves > 0); // Muestra el
panel si el jugador tiene al menos una llave
281     }
282 }
283
284 private void OnTriggerEnter2D(Collider2D collision)
285 {
286     Debug.Log($"Colisi n detectada con: {collision.gameObject.name}
");
287
288     if (collision.CompareTag("Key"))
289     {
290         itemNearby = collision.gameObject; // Asigna la llave como
el tem cercano
291         pickupPrompt.SetActive(true); // Muestra el mensaje de "
Presiona E para recoger"
292         Debug.Log("Llave detectada. Mostrando mensaje de recoger.");
293     }
294     else if (collision.CompareTag("EnergyBall"))
295     {
296         TakeDamage(5); // Ajusta el da o que recibe el jugador
297         Destroy(collision.gameObject); // Destruye la bola de
energ a
298         Debug.Log("El jugador ha recibido da o de la bola de
energ a.");
299     }
300 }
301
302 private void OnTriggerExit2D(Collider2D collision)
303 {
304     Debug.Log($"Saliendo de colisi n con: {collision.gameObject.
name}");
305     if (itemNearby == collision.gameObject)
306     {
307         itemNearby = null;
308         if (pickupPrompt != null)
309             pickupPrompt.SetActive(false);

```

```

310         Debug.Log(" tem fuera de alcance. Ocultando mensaje.");
311     }
312 }
313 void Awake()
314 {
315     if (UnityEngine.SceneManagement.SceneManager.GetActiveScene().
name == "GameplayScene 1")
316     {
317         PlayerPrefs.DeleteAll();
318         Debug.Log("PlayerPrefs reiniciados al iniciar GameplayScene
1.");
319     }
320
321     if (Object.FindObjectsByType<PlayerMovement>(FindObjectsSortMode
.None).Length > 1)
322     {
323         Destroy(gameObject); // evita duplicados
324     }
325     else
326     {
327         DontDestroyOnLoad(gameObject); // persiste el jugador entre
escenas
328         SceneManager.sceneLoaded += OnSceneLoaded;
329     }
330 }
331
332
333 void OnDisable()
334 {
335     Debug.Log("Desuscribiendo OnSceneLoaded del evento SceneManager.
sceneLoaded.");
336     SceneManager.sceneLoaded -= OnSceneLoaded;
337 }
338
339 public void OnSceneLoaded(Scene scene, LoadSceneMode mode)
340 {
341     if (this == null)
342     {
343         Debug.LogWarning("El objeto PlayerMovement ha sido destruido
. Ignorando OnSceneLoaded.");
344         return;
345     }
346
347     GameObject spawnPoint = GameObject.Find("SpawnPoint");
348     if (spawnPoint != null)
349     {
350         transform.position = spawnPoint.transform.position;
351     }
352     else
353     {
354         Debug.LogWarning("SpawnPoint not found in the new scene.
Ensure a GameObject named 'SpawnPoint' exists.");
355     }
356
357     foreach (SpriteRenderer sr in GetComponentsInChildren<
SpriteRenderer>())
358     {
359         sr.sortingLayerName = "Player";

```

```

360         sr.sortingOrder = 5;
361     }
362
363     UpdateHealthUI();
364
365     if (deathMenuCanvas != null)
366         deathMenuCanvas.SetActive(false);
367
368     if (scoreText != null)
369     {
370         scoreText.gameObject.SetActive(true);
371         scoreText.text = $"Puntos: {score}";
372     }
373     if (rb != null)
374     {
375         rb.linearVelocity = Vector2.zero;
376     }
377     isReloading = false; // Reinicia el estado de recarga
378     this.enabled = true;
379 }
380
381 public void UpdateUI()
382 {
383     PlayerInventory inventory = GetComponent<PlayerInventory>();
384     if (medkitText != null)
385         medkitText.text = $"{{medkitCount}}";
386
387     if (ammoText != null)
388         ammoText.text = $"{{currentAmmo}}/{{totalAmmo}}";
389
390     if (scoreText != null)
391         scoreText.text = $"Puntos: {score}";
392
393     if (keyPanel != null)
394         keyPanel.SetActive(inventory.llaves > 0); // Muestra el
panel si el jugador tiene al menos una llave
395 }
396
397 public void AddScore(int amount, Vector3 worldPosition, string
popupText = "")
398 {
399     score += amount;
400
401     if (scoreText != null)
402         scoreText.text = $"Puntos: {score}";
403
404     if (scorePopupPrefab != null)
405     {
406         GameObject popup = Instantiate(scorePopupPrefab,
worldPosition, Quaternion.identity);
407         TextMeshPro popupTMP = popup.GetComponentInChildren<
TextMeshPro>();
408         if (popupTMP != null)
409         {
410             popupTMP.text = string.IsNullOrEmpty(popupText) ? $"+{
amount}" : popupText;
411         }
412     }

```

```

413         Destroy(popup, 1.5f);
414     }
415 }
416
417 public void UpdateHealthUI()
418 {
419     healthBar.value = currentHealth;
420     if (healthText != null)
421         healthText.text = $"Vida: {currentHealth}";
422 }
423
424 void Shoot()
425 {
426     float direction = spriteRenderer.flipX ? -1f : 1f;
427     Vector3 spawnOffset = new Vector3(0.5f * direction, 0f, 0f);
428
429     Vector3 spawnPosition = firePoint.position + spawnOffset;
430     GameObject bullet = Instantiate(bulletPrefab, spawnPosition,
Quaternion.identity);
431
432     Rigidbody2D bulletRb = bullet.GetComponent<Rigidbody2D>();
433     bulletRb.linearVelocity = new Vector2(direction * bulletSpeed, 0
f);
434
435     // Ignorar colisi n con el jugador
436     Collider2D bulletCollider = bullet.GetComponent<Collider2D>();
437     Collider2D playerCollider = GetComponent<Collider2D>();
438     Physics2D.IgnoreCollision(bulletCollider, playerCollider);
439
440     // Disparar el sonido de casquillo despu s de 0.2 segundos
441     Invoke(nameof(PlayCasingSound), Random.Range(0.3f, 0.6f));
442 }
443
444 void PlayCasingSound()
445 {
446     PlaySound(casingSound);
447 }
448
449 void PlaySound(AudioClip clip)
450 {
451     if (clip != null)
452         AudioSource.PlayClipAtPoint(clip, transform.position);
453 }
454
455 void OnDrawGizmosSelected()
456 {
457     // Para ver el c rculo en el editor
458     if (groundCheck != null)
459     {
460         Gizmos.color = Color.red;
461         Gizmos.DrawWireSphere(groundCheck.position,
groundCheckRadius);
462     }
463 }
464
465 private IEnumerator ReloadRoutine()
466 {
467     isReloading = true;

```

```

468     animator.SetTrigger("Reload");
469     PlaySound(reloadSound);
470
471     yield return new WaitForSeconds(1.5f);
472
473     int needed = clipSize - currentAmmo;
474
475     // Recarga solo la cantidad necesaria y disponible
476     int ammoToReload = Mathf.Min(needed, totalAmmo);
477     currentAmmo += ammoToReload;
478     totalAmmo -= ammoToReload;
479
480
481     isReloading = false;
482     shotsSinceLastReload = 0;
483 }
484
485 public void TakeDamage(int damage)
486 {
487     currentHealth -= damage;
488     currentHealth = Mathf.Clamp(currentHealth, 0, maxHealth);
489     healthBar.value = currentHealth;
490
491     if (hurtSound != null)
492         PlaySound(hurtSound); // reproducir sonido de da o
493
494     if (currentHealth <= 0)
495     {
496         Die();
497     }
498 }
499
500 public void SavePlayerData()
501 {
502     PlayerPrefs.SetInt("Municion", totalAmmo);
503     PlayerPrefs.SetInt("MunicionActual", currentAmmo);
504     PlayerPrefs.SetInt("Puntuacion", score);
505     PlayerPrefs.SetInt("Botiquines", medkitCount);
506     PlayerPrefs.SetInt("Vida", currentHealth);
507     PlayerPrefs.Save();
508 }
509
510 void LoadPlayerData()
511 {
512     if (PlayerPrefs.HasKey("Vida"))
513     {
514         Debug.Log("Recargando datos desde PlayerPrefs en
515 LoadPlayerData.");
516         currentHealth = PlayerPrefs.GetInt("Vida");
517         totalAmmo = PlayerPrefs.GetInt("Municion");
518         currentAmmo = PlayerPrefs.GetInt("MunicionActual");
519         score = PlayerPrefs.GetInt("Puntuacion");
520         medkitCount = PlayerPrefs.GetInt("Botiquines");
521     }
522     else
523     {
524         Debug.Log("No hay datos en PlayerPrefs. Usando valores
525 predeterminados en LoadPlayerData.");
526     }
527 }

```

```

524         currentAmmo = clipSize;
525         totalAmmo = maxAmmo - clipSize;
526         currentHealth = maxHealth;
527         score = 0;
528         medkitCount = 0;
529     }
530
531     UpdateUI();
532     UpdateHealthUI();
533 }
534
535 public void ChangeScene(string nextScene)
536 {
537     SavePlayerData();
538     SceneManager.LoadScene(nextScene);
539 }
540
541 void CargarEscena()
542 {
543     PlayerMovement player = Object.FindFirstObjectByType<
PlayerMovement>();
544     if (player != null)
545     {
546         player.SavePlayerData();
547         SceneManager.sceneLoaded -= player.OnSceneLoaded; //
Desuscribirse del evento
548         Destroy(player.gameObject); // Destruir el objeto Player
549     }
550
551     if (!string.IsNullOrEmpty(sceneToLoad) && Application.
CanStreamedLevelBeLoaded(sceneToLoad))
552     {
553         Debug.Log($"Cargando escena: {sceneToLoad}");
554         SceneManager.LoadScene(sceneToLoad);
555     }
556     else
557     {
558         Debug.LogError($"La escena '{sceneToLoad}' no existe o no
est incluida en los Build Settings.");
559     }
560 }
561
562 void Die()
563 {
564     Debug.Log(" El jugador ha muerto!");
565
566     if (deathSound != null)
567         PlaySound(deathSound);
568
569     if (deathEffect != null)
570         Instantiate(deathEffect, transform.position, Quaternion.
identity);
571
572     // Activar animaci n y desactivar movimiento
573     animator.SetTrigger("Die");
574     rb.linearVelocity = Vector2.zero;
575     this.enabled = false;
576

```

```

577     StartCoroutine(ShowDeathMenuAfterDelay(0.5f));
578 }
579
580 public void ResetPlayerState()
581 {
582     currentHealth = maxHealth;
583     currentAmmo = clipSize;
584     totalAmmo = maxAmmo - clipSize;
585     medkitCount = 0;
586     score = 0;
587
588     // Reiniciar el contador de llaves
589     PlayerInventory inventory = GetComponent<PlayerInventory>();
590     if (inventory != null)
591     {
592         inventory.llaves = 0; // Reinicia las llaves a 0
593         Debug.Log("El contador de llaves se ha reiniciado.");
594     }
595
596     UpdateUI();
597     UpdateHealthUI();
598
599     rb.linearVelocity = Vector2.zero;
600     this.enabled = true;
601
602     // Reiniciar el Animator
603     animator.ResetTrigger("Die");
604     animator.Play("swat_idle");
605
606     // Reiniciar el men de muerte
607     if (deathMenuCanvas != null)
608     {
609         deathMenuCanvas.SetActive(false);
610     }
611 }
612
613 IEnumerator ShowDeathMenuAfterDelay(float delay)
614 {
615     yield return new WaitForSeconds(delay);
616
617     if (deathMenuCanvas != null)
618     {
619         deathMenuCanvas.SetActive(true);
620     }
621 }
622
623 public int GetCurrentHealth() => currentHealth;
624 public void SetCurrentHealth(int value) => currentHealth = value;
625
626 public int GetCurrentAmmo() => currentAmmo;
627 public void SetCurrentAmmo(int value) => currentAmmo = value;
628
629 public int GetTotalAmmo() => totalAmmo;
630 public void SetTotalAmmo(int value) => totalAmmo = value;
631
632 public int GetMedkitCount() => medkitCount;
633 public void SetMedkitCount(int value) => medkitCount = value;
634

```

```

635     public int GetScore() => score;
636     public void SetScore(int value) => score = value;
637
638 }

```

Listing 11: PlayerMovement.cs

## 12. PlayerInventory.cs

```

1 using UnityEngine;
2
3 public class PlayerInventory : MonoBehaviour
4 {
5     public int llaves = 0;
6
7     public void TomarLlave()
8     {
9         llaves++;
10        Debug.Log("Llave obtenida. Total: " + llaves);
11    }
12
13    public bool UsarLlave()
14    {
15        if (llaves > 0)
16        {
17            llaves--;
18            return true;
19        }
20        return false;
21    }
22 }

```

Listing 12: PlayerInventory.cs

## 13. PlayerDoorRaycast.cs

```

1     using UnityEngine;
2 using TMPro;
3 using UnityEngine.SceneManagement;
4
5 public class PlayerDoorRaycast : MonoBehaviour
6 {
7     public float rayDistance = 2f;
8     public LayerMask doorLayer;
9     public GameObject mensajeEntrarUI;
10    public GameObject mensajeSinLlaveUI;
11    public string sceneToLoad;
12    public AudioSource audioSource;
13    public AudioClip doorOpenSound;
14
15    private bool puedeEntrar = false;
16    private bool escenaCargando = false;
17
18    void Update()
19    {
20        MostrarMensajeEntrar();
21    }

```



```

22         if (Input.GetKeyDown(KeyCode.E) && puedeEntrar && !
escenaCargando)
23     {
24         PlayerInventory inventory = GetComponent<PlayerInventory>();
25         if (inventory != null && inventory.llaves > 0)
26     {
27             escenaCargando = true;
28             inventory.UsarLlave();
29             audioSource.PlayOneShot(doorOpenSound);
30             Invoke("CargarEscena", doorOpenSound.length);
31             if (mensajeSinLlaveUI != null)
32                 mensajeSinLlaveUI.SetActive(false); // Oculta el
mensaje si ten a llave
33         }
34         else
35     {
36             Debug.Log("No tienes llaves para abrir esta puerta.");
37             if (mensajeSinLlaveUI != null)
38         {
39                 mensajeSinLlaveUI.SetActive(true);
40                 CancelInvoke("OcultarMensajeSinLlave");
41                 Invoke("OcultarMensajeSinLlave", 2f); // Oculta el
mensaje despu s de 2 segundos
42             }
43         }
44     }
45 }
46
47 void OcultarMensajeSinLlave()
48 {
49     if (mensajeSinLlaveUI != null)
50         mensajeSinLlaveUI.SetActive(false);
51 }
52
53 void CargarEscena()
54 {
55     PlayerMovement player = Object.FindFirstObjectByType<
PlayerMovement>();
56     if (player != null)
57     {
58         player.SavePlayerData();
59         SceneManager.sceneLoaded -= player.OnSceneLoaded;
60         Destroy(player.gameObject);
61     }
62
63     if (!string.IsNullOrEmpty(sceneToLoad) && Application.
CanStreamedLevelBeLoaded(sceneToLoad))
64     {
65         Debug.Log($"Cargando escena: {sceneToLoad}");
66         SceneManager.LoadScene(sceneToLoad);
67     }
68     else
69     {
70         Debug.LogError($"La escena '{sceneToLoad}' no existe o no
est incluida en los Build Settings.");
71     }
72 }
73

```

```

74 void MostrarMensajeEntrar()
75 {
76     Vector2 direccion = transform.right * (GetComponent<
SpriteRenderer>().flipX ? -1 : 1);
77     Debug.DrawRay((Vector2)transform.position + Vector2.up * 0.5f,
direccion * rayDistance, Color.red);
78     RaycastHit2D hit = Physics2D.Raycast((Vector2)transform.position
+ Vector2.up * 0.5f, direccion, rayDistance, doorLayer);
79
80     if (hit.collider != null && hit.collider.CompareTag("Puerta"))
81     {
82         mensajeEntrarUI.SetActive(true);
83         puedeEntrar = true;
84     }
85     else
86     {
87         mensajeEntrarUI.SetActive(false);
88         puedeEntrar = false;
89         if (mensajeSinLlaveUI != null)
90             mensajeSinLlaveUI.SetActive(false);
91     }
92 }
93
94 }

```

Listing 13: PlayerDoorRaycast.cs

## 14. DoorScript.cs

```

1 using UnityEngine;
2 using UnityEngine.SceneManagement;
3
4 public class Door : MonoBehaviour
5 {
6     public string sceneToLoad; // Nombre de la escena a cargar
7
8     private void OnValidate()
9     {
10         if (string.IsNullOrEmpty(sceneToLoad))
11         {
12             Debug.LogWarning($"The 'sceneToLoad' field is empty on {
gameObject.name}. Assign a scene in the Inspector.");
13         }
14     }
15
16     private void OnTriggerEnter(Collider other)
17     {
18         if (other.CompareTag("Player"))
19         {
20             PlayerInventory inv = other.GetComponent<PlayerInventory>();
21             if (inv != null && inv.UsarLlave())
22             {
23                 Debug.Log("Puerta abierta. Cambiando de escena...");
24                 LoadScene();
25             }
26             else
27             {

```

```

28         Debug.Log("Necesitas una llave para abrir esta puerta.")
29     ;
30     }
31 }
32
33 public void LoadScene()
34 {
35     if (!string.IsNullOrEmpty(sceneToLoad))
36     {
37         if (Application.CanStreamedLevelBeLoaded(sceneToLoad))
38         {
39             Debug.Log("Cargando escena: " + sceneToLoad);
40             SceneManager.LoadScene(sceneToLoad);
41         }
42         else
43         {
44             Debug.LogError($"La escena '{sceneToLoad}' no existe o
no est incluida en Build Settings.");
45         }
46     }
47     else
48     {
49         Debug.LogError("El campo 'sceneToLoad' en el script Door
est vac o. Asigna una escena en el Inspector.");
50     }
51 }
52 }

```

Listing 14: DoorScript.cs

## 15. KeyPickup.cs

```

1 using UnityEngine;
2
3 public class KeyPickup : MonoBehaviour
4 {
5     private void OnTriggerEnter(Collider other)
6     {
7         if (other.CompareTag("Player"))
8         {
9             PlayerInventory inventory = other.GetComponent<
PlayerInventory>();
10             if (inventory != null)
11             {
12                 // Limpia el mensaje de recoger si existe
13                 PlayerMovement player = other.GetComponent<
PlayerMovement>();
14                 if (player != null)
15                 {
16                     player.itemNearby = null;
17                     if (player.pickupPrompt != null)
18                         player.pickupPrompt.SetActive(false);
19                 }
20
21                 inventory.TomarLlave();
22                 Destroy(gameObject); // Destruye la llave despu s de
recogerla

```

```

23     }
24 }
25 }
26 }

```

Listing 15: KeyPickup.cs

## 16. ItemPickup.cs

```

1 using UnityEngine;
2
3 public class ItemPickup : MonoBehaviour
4 {
5     private void OnTriggerEnter2D(Collider2D other)
6     {
7         if (other.CompareTag("Player"))
8         {
9             PlayerMovement player = other.GetComponentInParent<
PlayerMovement>();
10             if (player != null)
11             {
12                 player.itemNearby = gameObject;
13                 player.pickupPrompt?.SetActive(true);
14             }
15         }
16     }
17
18     private void OnTriggerExit2D(Collider2D collision)
19     {
20         Debug.Log($"Saliendo de colisi n con: {collision.gameObject.
name}");
21
22         PlayerMovement player = collision.GetComponentInParent<
PlayerMovement>();
23         if (player != null && player.itemNearby == gameObject)
24         {
25             player.itemNearby = null;
26             if (player.pickupPrompt != null)
27                 player.pickupPrompt.SetActive(false);
28             Debug.Log(" tem fuera de alcance. Ocultando mensaje.");
29         }
30     }
31 }

```

Listing 16: ItemPickup.cs

## 17. PlayerSystemRaycast.cs

```

1 using UnityEngine;
2
3 public class PlayerSystemRaycast : MonoBehaviour
4 {
5     public float rayDistance = 2f;
6     public LayerMask systemLayer;
7     public GameObject mensajeInteractuarUI;
8     public SystemInteractionMenu systemMenu;
9

```

```

10 private bool puedeInteractuar = false;
11 private PlayerInventory playerInventory;
12
13 void Start()
14 {
15     playerInventory = GetComponent<PlayerInventory>();
16 }
17
18 void Update()
19 {
20     MostrarMensajeInteractuar();
21
22     if (Input.GetKeyDown(KeyCode.E) && puedeInteractuar)
23     {
24         if (systemMenu != null && playerInventory != null &&
25 playerInventory.llaves > 0)
26         {
27             systemMenu.AbrirMenu();
28         }
29     }
30
31     void MostrarMensajeInteractuar()
32     {
33         Vector2 direccion = transform.right * (GetComponent<
34 SpriteRenderer>().flipX ? -1 : 1);
35         Debug.DrawRay((Vector2)transform.position + Vector2.up * 0.5f,
36 direccion * rayDistance, Color.cyan);
37         RaycastHit2D hit = Physics2D.Raycast((Vector2)transform.position
38 + Vector2.up * 0.5f, direccion, rayDistance, systemLayer);
39
40         // Solo muestra el mensaje si el jugador tiene al menos una
41 llave
42         if (hit.collider != null && hit.collider.CompareTag("System") &&
43 playerInventory != null && playerInventory.llaves > 0)
44         {
45             mensajeInteractuarUI.SetActive(true);
46             puedeInteractuar = true;
47             systemMenu = hit.collider.GetComponent<SystemInteractionMenu
48 >();
49         }
50         else
51         {
52             mensajeInteractuarUI.SetActive(false);
53             puedeInteractuar = false;
54             systemMenu = null;
55         }
56     }
57 }

```

Listing 17: PlayerSystemRaycast.cs

## 18. SystemInteractionMenu.cs

```

1 using UnityEngine;
2 using UnityEngine.UI;
3 using UnityEngine.SceneManagement;
4

```

```

5 public class SystemInteractionMenu : MonoBehaviour
6 {
7     public GameObject menuUI; // Panel del men (Canvas)
8     public Button opcion1Button;
9     public Button opcion2Button;
10    public string escenaOpcion1;
11    public string escenaOpcion2;
12
13    private bool jugadorCerca = false;
14
15    void Start()
16    {
17        menuUI.SetActive(false);
18
19        opcion1Button.onClick.AddListener(() => CargarEscenaFinal(
escenaOpcion1));
20        opcion2Button.onClick.AddListener(() => CargarEscenaFinal(
escenaOpcion2));
21    }
22
23    void OnTriggerEnter2D(Collider2D other)
24    {
25        if (other.CompareTag("Player"))
26        {
27            jugadorCerca = true;
28        }
29    }
30
31    void OnTriggerExit2D(Collider2D other)
32    {
33        if (other.CompareTag("Player"))
34        {
35            jugadorCerca = false;
36            menuUI.SetActive(false);
37            Time.timeScale = 1f;
38        }
39    }
40
41    void CargarEscenaFinal(string nombreEscena)
42    {
43        // Elimina el jugador y su UI antes de cambiar de escena
44        PlayerMovement player = Object.FindFirstObjectByType<
PlayerMovement>();
45        if (player != null)
46        {
47            Destroy(player.gameObject);
48        }
49
50        Time.timeScale = 1f;
51        UnityEngine.SceneManagement.SceneManager.LoadScene(nombreEscena)
;
52    }
53
54    public void AbrirMenu()
55    {
56        menuUI.SetActive(true);
57        Time.timeScale = 0f;
58    }

```

## 19. PlayerDocumentRaycast.cs

```

1 using UnityEngine;
2 using TMPro;
3
4 public class PlayerDocumentRaycast : MonoBehaviour
5 {
6     public float rayDistance = 2f;
7     public LayerMask documentLayer;
8     public GameObject lecturaPanel;
9     public TextMeshProUGUI textoUI;
10    public TextMeshProUGUI mensajeLecturaUI;
11
12
13    private bool leyendo = false;
14    private DocumentReader documentoActual;
15
16    void Update()
17    {
18        if (!leyendo)
19        {
20            MostrarMensajeLectura(); // Nueva funcin para mostrar el
mensaje si hay un documento
21        }
22
23        if (Input.GetKeyDown(KeyCode.Q))
24        {
25            if (leyendo)
26            {
27                CerrarLectura();
28            }
29            else
30            {
31                DetectarDocumento();
32            }
33        }
34    }
35
36
37    void DetectarDocumento()
38    {
39        Vector2 direccion = transform.right * (GetComponent<
SpriteRenderer>().flipX ? -1 : 1);
40        RaycastHit2D hit = Physics2D.Raycast((Vector2)transform.position
+ Vector2.up * 0.5f, direccion, rayDistance, documentLayer);
41
42        if (hit.collider != null && hit.collider.CompareTag("Documento")
)
43        {
44            documentoActual = hit.collider.GetComponent<DocumentReader
>();
45            if (documentoActual != null)
46            {
47                AbrirLectura(documentoActual.GetText());

```

```

48         }
49     }
50 }
51
52 void AbrirLectura(string texto)
53 {
54     lecturaPanel.SetActive(true);
55     textoUI.text = texto;
56     leyendo = true;
57     mensajeLecturaUI.gameObject.SetActive(false);
58 }
59
60 void MostrarMensajeLectura()
61 {
62     Vector2 direccion = transform.right * (GetComponent<
63     SpriteRenderer>().flipX ? -1 : 1);
64     RaycastHit2D hit = Physics2D.Raycast((Vector2)transform.position
65     + Vector2.up * 0.5f, direccion, rayDistance, documentLayer);
66
67     if (hit.collider != null && hit.collider.CompareTag("Documento"))
68     {
69         mensajeLecturaUI.gameObject.SetActive(true);
70     }
71     else
72     {
73         mensajeLecturaUI.gameObject.SetActive(false);
74     }
75 }
76
77 void CerrarLectura()
78 {
79     lecturaPanel.SetActive(false);
80     mensajeLecturaUI.gameObject.SetActive(false); // Ocultar mensaje
81     leyendo = false;
82     documentoActual = null;
83 }
84 }

```

Listing 19: PlayerDocumentRaycast.cs

## 20. PlayerDocumentInteraction.cs

```

1 using UnityEngine;
2 using TMPro;
3
4 public class PlayerDocumentInteraction : MonoBehaviour
5 {
6     public GameObject panelLectura;
7     public TextMeshProUGUI textoUI;
8
9     private DocumentReader documentoActual;
10    private bool leyendo = false;
11
12    void Update()
13    {

```



```

14         if (documentoActual != null && Input.GetKeyDown(KeyCode.E))
15         {
16             Debug.Log("Presionando E para documento: " + documentoActual
17                 .name);
18             if (!leyendo)
19             {
20                 textoUI.text = documentoActual.GetText();
21                 panelLectura.SetActive(true);
22                 leyendo = true;
23                 Time.timeScale = 0;
24             }
25             else
26             {
27                 panelLectura.SetActive(false);
28                 leyendo = false;
29                 Time.timeScale = 1;
30             }
31         }
32
33     public void SetDocumento(DocumentReader doc)
34     {
35         documentoActual = doc;
36     }
37
38     public void ClearDocumento()
39     {
40         documentoActual = null;
41     }
42 }

```

Listing 20: PlayerDocumentInteraction.cs

## 21. InteractionZone.cs

```

1 using UnityEngine;
2
3 public class InteractionZone : MonoBehaviour
4 {
5     private PlayerDocumentInteraction playerInteraction;
6
7     private void Start()
8     {
9         playerInteraction = GetComponentInParent<
10         PlayerDocumentInteraction>();
11     }
12
13     void OnTriggerEnter(Collider other)
14     {
15         if (other.CompareTag("Documento"))
16         {
17             Debug.Log("Documento detectado: " + other.name);
18             playerInteraction.SetDocumento(other.GetComponent<
19             DocumentReader>());
20         }
21     }
22 }

```

```

22
23     void OnTriggerExit(Collider other)
24     {
25         if (other.CompareTag("Documento"))
26         {
27             playerInteraction.ClearDocumento();
28         }
29     }
30 }

```

Listing 21: InteractionZone.cs

## 22. DocumentReader.cs

```

1 using UnityEngine;
2
3 public class DocumentReader : MonoBehaviour
4 {
5     [TextArea]
6     public string textoDocumento;
7
8     public AudioClip sonidoApertura;
9     private AudioSource audioSource;
10
11     private void Awake()
12     {
13         audioSource = gameObject.AddComponent<AudioSource>();
14     }
15
16     public string GetTexto()
17     {
18         if (sonidoApertura != null && audioSource != null)
19         {
20             audioSource.PlayOneShot(sonidoApertura);
21         }
22         return textoDocumento;
23     }
24 }

```

Listing 22: DocumentReader.cs

## 23. BoosFinalAI.cs

```

1 using UnityEngine;
2 using UnityEngine.UI;
3 using System.Collections;
4
5 public class BossFinalAI : MonoBehaviour
6 {
7     public float speed = 0.8f;
8     public float attackCooldown = 2f;
9     public float health = 100f;
10    public float stunDuration = 2f;
11
12    public float energyBallCooldown = 3f; // Nuevo cooldown para la bola
    de energ a
13    private float lastEnergyBallTime = 0f; // ltima vez que dispar

```

```

14
15     public AudioClip growlSound;
16     public AudioClip hitSound;
17     public AudioClip deathSound;
18     public AudioClip specialAttackSound; // Sonido para el ataque
especial
19     public AudioClip normalAttackSound; // Sonido para el ataque normal
20
21     public GameObject energyBallPrefab;
22     public Transform firePoint;
23
24     public Image bossHealthBar; // Referencia a la UI fija
25
26     private Transform player;
27     private Animator animator;
28     private Rigidbody2D rb;
29     private AudioSource audioSource;
30
31     private float lastAttackTime;
32     private bool isStunned = false;
33     private bool isDead = false;
34
35     [SerializeField] private GameObject keyPrefab;
36
37     void Start()
38     {
39         player = GameObject.FindGameObjectWithTag("Player").transform;
40         animator = GetComponent<Animator>();
41         rb = GetComponent<Rigidbody2D>();
42         audioSource = GetComponent<AudioSource>();
43
44         if (bossHealthBar != null)
45             bossHealthBar.fillAmount = 2f;
46
47         PlayGrowl();
48     }
49
50     void OnTriggerEnter2D(Collider2D collision)
51     {
52         if (collision.CompareTag("Bala"))
53         {
54             TakeDamage(50);
55             Destroy(collision.gameObject); // Destruye el proyectil
56             Debug.Log("El jefe ha recibido da o de la bala del jugador.
");
57         }
58     }
59
60     void Update()
61     {
62         if (player == null || isDead) return;
63
64         if (isStunned)
65         {
66             animator.SetBool("isRunning", false);
67             return;
68         }
69

```

```

70     Vector2 direction = (player.position - transform.position).
normalized;
71     float distance = Vector2.Distance(transform.position, player.
position);
72
73     // Disparo de bola de energ a cada X segundos (independiente
del ataque)
74     if (Time.time - lastEnergyBallTime >= energyBallCooldown)
75     {
76         lastEnergyBallTime = Time.time;
77         FireEnergyBall();
78     }
79
80     // Ataque especial o ataque a distancia cada attackCooldown
81     bool isAttacking = Time.time - lastAttackTime < 1f;
82     animator.SetBool("isRunning", !isAttacking);
83
84     if (!isAttacking)
85     {
86         transform.position += (Vector3)direction * speed * Time.
deltaTime;
87
88         if (direction.x > 0)
89             transform.localScale = new Vector3(1, 1, 1);
90         else if (direction.x < 0)
91             transform.localScale = new Vector3(-1, 1, 1);
92     }
93
94     if (Time.time - lastAttackTime >= attackCooldown)
95     {
96         if (distance <= 2f)
97         {
98             animator.SetTrigger("SpecialAttackTrigger");
99             PlaySound(specialAttackSound);
100             PerformSpecialAttack();
101         }
102         else
103         {
104             animator.SetTrigger("AttackTrigger");
105             PlaySound(normalAttackSound);
106         }
107         lastAttackTime = Time.time;
108     }
109 }
110
111
112 IEnumerator DelayedFire(float delay)
113 {
114     yield return new WaitForSeconds(delay);
115     FireEnergyBall();
116 }
117
118 void PerformSpecialAttack()
119 {
120     float specialAttackRange = 2f; // Rango del ataque especial
121     int specialAttackDamage = 5; // Da o del ataque especial
122
123     Collider2D[] hitColliders = Physics2D.OverlapCircleAll(transform

```

```

124     .position, specialAttackRange);
125     foreach (Collider2D collider in hitColliders)
126     {
127         if (collider.CompareTag("Player"))
128         {
129             PlayerMovement player = collider.GetComponent<
130             PlayerMovement>();
131             if (player != null)
132             {
133                 player.TakeDamage(specialAttackDamage);
134                 Debug.Log("El jugador ha recibido da o del ataque
135                 especial del jefe.");
136             }
137         }
138     }
139
140     public void FireEnergyBall()
141     {
142         if (energyBallPrefab != null && firePoint != null)
143         {
144             GameObject ball = Instantiate(energyBallPrefab, firePoint.
145             position, Quaternion.identity);
146             Vector2 dir = (player.position - firePoint.position).
147             normalized;
148             ball.GetComponent<Rigidbody2D>().linearVelocity = dir * 2f;
149         }
150     }
151
152     public void TakeDamage(float damage)
153     {
154         if (isDead) return;
155
156         health -= damage;
157         PlaySound(hitSound);
158
159         if (bossHealthBar != null)
160             bossHealthBar.fillAmount = health / 500f;
161
162         if (health <= 0)
163         {
164             Die();
165         }
166         else if (!isStunned)
167         {
168             isStunned = true;
169             Invoke("ResetStun", stunDuration);
170         }
171     }
172
173     void Die()
174     {
175         isDead = true;
176         animator.SetTrigger("DieTrigger");
177         PlaySound(deathSound);
178
179         DropKey();

```

```

177     Destroy(gameObject, 1f);
178     if (bossHealthBar != null)
179         bossHealthBar.transform.parent.gameObject.SetActive(false);
180 }
181
182 void ResetStun()
183 {
184     isStunned = false;
185 }
186
187 void DropKey()
188 {
189     if (keyPrefab != null && GameObject.FindGameObjectWithTag("Key")
190 == null)
191     {
192         Instantiate(keyPrefab, transform.position, Quaternion.
193 identity);
194         Debug.Log("Llave soltada por el Boss.");
195     }
196 }
197
198 void PlaySound(AudioClip clip)
199 {
200     if (audioSource != null && clip != null)
201     {
202         audioSource.PlayOneShot(clip);
203     }
204 }
205
206 void PlayGrowl()
207 {
208     if (audioSource != null && growlSound != null)
209     {
210         audioSource.loop = true;
211         audioSource.clip = growlSound;
212         audioSource.Play();
213     }
214 }
215 }

```

Listing 23: BoosFinalAI.cs

## 24. BulletScript.cs

```

1 using UnityEngine;
2
3 public class Bullet : MonoBehaviour
4 {
5     public GameObject explosionEffect;
6     public float destroyDelay = 0.1f;
7     public float damage = 50f;
8
9     void OnTriggerEnter2D(Collider2D collision)
10    {
11        // Filtrar: si no es Zombie ni Ground, ignorar completamente
12        if (!collision.CompareTag("Zombie") && !collision.CompareTag("
13        Ground"))
14        {

```

```

14         return;
15     }
16
17     Debug.Log("Colisin REAL con: " + collision.name + " | Tag: "
+ collision.tag);
18
19     // Instanciar la explosin
20     if (explosionEffect != null)
21     {
22         GameObject explosion = Instantiate(explosionEffect,
transform.position, Quaternion.identity);
23         explosion.transform.localScale = new Vector3(4f, 4f, 1f);
24         Destroy(explosion, 0.3f);
25     }
26
27     // D a o al zombie si aplica
28     if (collision.CompareTag("Zombie"))
29     {
30         ZombieAI zombie = collision.GetComponent<ZombieAI>();
31         if (zombie != null)
32         {
33             zombie.TakeDamage(damage);
34         }
35     }
36
37     Destroy(gameObject);
38 }
39 }

```

Listing 24: BulletScript.cs

## 25. FloatingText.cs

```

1 using UnityEngine;
2 using TMPro;
3
4 public class FloatingText : MonoBehaviour
5 {
6     public float floatSpeed = 1f;
7     public float lifetime = 1.2f;
8     public Vector3 floatDirection = Vector3.up;
9     public TextMeshPro text;
10
11     private Color originalColor;
12     private float timer;
13
14     void Start()
15     {
16         if (text == null)
17             text = GetComponent<TextMeshPro>();
18
19         originalColor = text.color;
20     }
21
22     void Update()
23     {
24         transform.position += floatDirection * floatSpeed * Time.
deltaTime;

```

```

25         timer += Time.deltaTime;
26         float alpha = Mathf.Lerp(originalColor.a, 0, timer / lifetime);
27         text.color = new Color(originalColor.r, originalColor.g,
28         originalColor.b, alpha);
29
30         if (timer >= lifetime)
31             Destroy(gameObject);
32     }
33
34     public void SetText(string content, Color color)
35     {
36         if (text == null)
37             text = GetComponent<TextMeshPro>();
38
39         text.text = content;
40         text.color = color;
41         originalColor = color;
42     }
43 }

```

Listing 25: FloatingText.cs

## 26. CameraFollow.cs

```

1 using UnityEngine;
2
3 public class CameraFollow : MonoBehaviour
4 {
5     public Transform target;           // El jugador
6     public Vector3 offset;             // Para ajustar la posicin de la
7                                         c mara
8
9     public Vector2 minLimits;           // L mite inferior izquierdo del
10    mapa
11    public Vector2 maxLimits;           // L mite superior derecho del mapa
12
13    void LateUpdate()
14    {
15        // Si el target es nulo, buscar al jugador nuevamente
16        if (target == null)
17        {
18            #if UNITY_2023_1_OR_NEWER
19                PlayerMovement foundPlayer = Object.FindFirstObjectByType<
20                PlayerMovement>();
21            #else
22                PlayerMovement foundPlayer = FindObjectOfTypes<PlayerMovement>
23                >();
24            #endif
25
26            if (foundPlayer != null)
27            {
28                target = foundPlayer.transform;
29            }
30            else
31            {
32                return; // Si no se encuentra el jugador, salir del
33                m todo
34            }
35        }
36    }
37 }

```



```

29     }
30
31     // Calcular la nueva posici n deseada con el offset
32     Vector3 desiredPosition = target.position + offset;
33
34     // Limitar el movimiento de la c mara dentro de los bordes
    definidos
35     float clampedX = Mathf.Clamp(desiredPosition.x, minLimits.x,
    maxLimits.x);
36     float clampedY = Mathf.Clamp(desiredPosition.y, minLimits.y,
    maxLimits.y);
37
38     // Establecer la nueva posici n de la c mara
39     transform.position = new Vector3(clampedX, clampedY, transform.
    position.z);
40 }
41 }

```

Listing 26: CameraFollow.cs

## 27. GameManager.cs

```

1 using UnityEngine;
2
3 public static class GameManager
4 {
5     public static int municion = 0;
6     public static int puntuacion = 0;
7     public static int botiquines = 0;
8 }

```

Listing 27: GameManager.cs

## 28. PlayerData.cs

```

1 using UnityEngine;
2
3 public static class PlayerData
4 {
5     public static int health;
6     public static int currentAmmo;
7     public static int totalAmmo;
8     public static int medkitCount;
9     public static int score;
10
11     public static void SaveData(PlayerMovement player)
12     {
13         health = player.GetCurrentHealth();
14         currentAmmo = player.GetCurrentAmmo();
15         totalAmmo = player.GetTotalAmmo();
16         medkitCount = player.GetMedkitCount();
17         score = player.GetScore();
18     }
19
20     public static void LoadData(PlayerMovement player)
21     {
22         player.SetCurrentHealth(health);

```

```

23     player.SetCurrentAmmo(currentAmmo);
24     player.SetTotalAmmo(totalAmmo);
25     player.SetMedkitCount(medkitCount);
26     player.SetScore(score);
27
28     // Update UI after loading data
29     player.UpdateUI();
30     player.UpdateHealthUI();
31 }
32 }

```

Listing 28: PlayerData.cs

## 29. DialogoManager.cs

```

1  using UnityEngine;
2  using UnityEngine.UI;
3  using System.Collections;
4  using TMPro;
5
6  public class DialogoManager : MonoBehaviour
7  {
8      public GameObject panelDialogo;
9      public TMP_Text textoDialogo;
10     public string[] lineas;
11     private int indice;
12
13
14     void Start()
15     {
16         panelDialogo.SetActive(false);
17     }
18
19     public void ActivarDialogo(string[] nuevoDialogo)
20     {
21         lineas = nuevoDialogo;
22         indice = 0;
23         panelDialogo.SetActive(true);
24         StartCoroutine(MostrarTexto());
25     }
26
27     IEnumerator MostrarTexto()
28     {
29         textoDialogo.text = "";
30         foreach (char letra in lineas[indice].ToCharArray())
31         {
32             textoDialogo.text += letra;
33             yield return new WaitForSeconds(0.03f); // efecto m quina
34         }
35     }
36
37     void Update()
38     {
39         if (panelDialogo.activeSelf && Input.GetKeyDown(KeyCode.E))
40         {
41             indice++;
42             if (indice < lineas.Length)

```

```

43         {
44             StartCoroutine(MostrarTexto());
45         }
46         else
47         {
48             panelDialogo.SetActive(false);
49         }
50     }
51 }
52
53 }

```

Listing 29: DialogoManager.cs

## 30. MissingScripts.cs

```

1 using UnityEngine;
2 using UnityEditor;
3
4 public class MissingScriptFinder : MonoBehaviour
5 {
6     #if UNITY_EDITOR
7         [MenuItem("Tools/Find Missing Scripts")]
8         public static void FindMissingScripts()
9         {
10             GameObject[] go = Object.FindObjectsByType<GameObject>(
11                 FindObjectsSortMode.None);
12             int goCount = 0, componentsCount = 0, missingCount = 0;
13
14             foreach (GameObject g in go)
15             {
16                 goCount++;
17                 Component[] components = g.GetComponents<Component>();
18                 for (int i = 0; i < components.Length; i++)
19                 {
20                     componentsCount++;
21                     if (components[i] == null)
22                     {
23                         missingCount++;
24                         Debug.Log($"GameObject '{g.name}' in scene has a
25                         missing script!", g);
26                     }
27                 }
28             }
29
30             Debug.Log($"Searched {goCount} GameObjects, {componentsCount}
31             components, found {missingCount} missing.");
32         }
33     #endif
34 }

```

Listing 30: MissingScripts.cs