

```
1 using System.Collections;
2 using System.Collections.Generic;
3 using Photon.Pun;
4 using UnityEngine;
5 using UnityEngine.SceneManagement;
6
7 public class MenuEstado : EstadoPartida
8 {
9     public override void IniciarEstado(ControlPartida controlPartida)
10    {
11        Time.timeScale = 1;
12        controlPartida.controladorReproductorMusical.IniciarModoMenu();
13        controlPartida.controladorMejoras.ReiniciarPoolMejoras();
14
15        if (controlPartida.controladorSala)
16        {
17            if (PhotonNetwork.InRoom)
18                controlPartida.controladorSala.desconectar();
19        }
20
21        controlPartida.controladorPuntos.reiniciarPunros();
22
23        SceneManager.LoadScene("MainMenu");
24    }
25
26    public override void ActualizarEstado(ControlPartida controlPartida)
27    {
28        if (controlPartida.controladorSala)
29        {
30            if (!PhotonNetwork.InRoom)
31                controlPartida.controladorSala.volverOffline();
32            if (!PhotonNetwork.IsConnected && !PhotonNetwork.InRoom)
33                controlPartida.controladorSala.Autodestruccion();
34        }
35    }
36 }
```

```
1 using System;
2 using System.Collections;
3 using System.Collections.Generic;
4 using Photon.Pun;
5 using UnityEngine;
6
7 public class SpawnPoint : MonoBehaviour
8 {
9     [SerializeField] private GameObject jugador;
10    private bool spawned = false;
11
12    // Start is called before the first frame update
13    void Start()
14    {
15    }
16
17    private void Update()
18    {
19        if (ControlPartida.Instancia.mapaGenerado && !spawned)
20        {
21            ControlPartida.Instancia.controladorReproductorMusical.IniciarModoPartida();
22            spawned = true;
23            GameObject _jugador;
24            if (ControlPartida.Instancia.controladorSala)
25            {
26                ControladorSala sala = ControlPartida.Instancia.controladorSala;
27                _jugador = PhotonNetwork.Instantiate(sala.jugador.name, transform.position, Quaternion.identity);
28            }
29            else
30            {
31                _jugador = Instantiate(jugador, transform.position, Quaternion.identity);
32            }
33
34            _jugador.GetComponent<SetupJugador>().Jugador_Local();
35        }
36    }
37 }
```

```
1 using System.Collections;
2 using System.Collections.Generic;
3 using UnityEngine;
4
5 public class CanvaInGame : MonoBehaviour
6 {
7     Camera camera;
8
9     // Start is called before the first frame update
10    void Start()
11    {
12        if (FindObjectOfType<CamaraJugador>())
13        {
14            camera = FindObjectOfType<CamaraJugador>().gameObject.GetComponent<Camera>();
15        }
16    }
17
18    // Update is called once per frame
19    void Update()
20    {
21        if (FindObjectOfType<CamaraJugador>())
22        {
23            camera = FindObjectOfType<CamaraJugador>().gameObject.GetComponent<Camera>();
24        }
25
26        if (!camera)
27        {
28            return;
29        }
30
31        transform.LookAt(transform.position + camera.transform.rotation * Vector3.back,
32            camera.transform.rotation * Vector3.up);
33    }
34 }
```

```
1 using System.Collections;
2 using System.Collections.Generic;
3 using UnityEngine;
4 using UnityEngine.SceneManagement;
5
6 public class FinalEstado : EstadoPartida
7 {
8     public override void IniciarEstado(ControlPartida controlPartida)
9     {
10         SceneManager.LoadScene("EndGame");
11     }
12
13     public override void ActualizarEstado(ControlPartida controlPartida)
14     {
15     }
16 }
17 }
```

```
1 using System.Collections;
2 using System.Collections.Generic;
3 using UnityEngine;
4 using UnityEngine.SceneManagement;
5
6 public class LobbyEstado : EstadoPartida
7 {
8     public override void IniciarEstado(ControlPartida controlPartida)
9     {
10         Time.timeScale = 1;
11         if (controlPartida.controladorSala)
12         {
13             controlPartida.controladorSala.Autodestruccion();
14             controlPartida.controladorSala = null;
15         }
16
17         controlPartida.controladorPuntos.reiniciarPuntos();
18         controlPartida.controladorMejoras.ReiniciarPoolMejoras();
19
20         SceneManager.LoadScene("Lobby");
21     }
22
23     public override void ActualizarEstado(ControlPartida controlPartida)
24     {
25     }
26 }
```

```
1 using UnityEngine;
2 using Random = System.Random;
3
4 public class Aleatoriedad
5 {
6     private int semilla;
7
8     private Random aleatorio;
9
10    public Aleatoriedad(int semilla)
11    {
12        this.semilla = semilla;
13        aleatorio = new Random(semilla);
14    }
15
16    public int nuevoEntero(int min, int max)
17    {
18        int rango = max - min;
19        semilla--;
20        return Mathf.Abs(semilla % rango) + min;
21    }
22
23    public float nuevoDecimal(float min, float max)
24    {
25        float rango = max - min;
26        semilla--;
27        return Mathf.Abs(semilla % rango) + min;
28    }
29 }
30 }
```

```
1 using System.Collections;
2 using System.Collections.Generic;
3 using UnityEngine;
4 using UnityEngine.SceneManagement;
5
6 public class InicioEstado : EstadoPartida
7 {
8     public override void IniciarEstado(ControlPartida controlPartida)
9     {
10         controlPartida.cambiarEstado(controlPartida.menuEstado);
11     }
12
13     public override void ActualizarEstado(ControlPartida controlPartida)
14     {
15     }
16 }
17 }
18
```

```
1 using System.Collections;
2 using System.Collections.Generic;
3 using UnityEngine;
4
5 public class SetupJugador : MonoBehaviour
6 {
7     [SerializeField] private Camera camaraPrincipal;
8     [SerializeField] private Camera camaraMinimapa;
9     [SerializeField] private AudioListener audioJugador;
10    [SerializeField] private Canvas canvas;
11    [SerializeField] private MoverAlPunto mover;
12    [SerializeField] private ControladorClicks clicks;
13    [SerializeField] private ControladorPausa pausa;
14    [SerializeField] private GameObject canvaPropio;
15
16    public void Jugador_local()
17    {
18        camaraPrincipal.enabled = true;
19        audioJugador.enabled = true;
20        camaraMinimapa.enabled = true;
21        mover.enabled = true;
22        clicks.enabled = true;
23        canvas.enabled = true;
24        canvaPropio.SetActive(false);
25        //pausa.enabled = true;
26
27
28        foreach (ControladorHover controlador in FindObjectsOfType<ControladorHover>())
29        {
30            controlador.enabled = true;
31        }
32    }
33 }
34
35 }
36
37
```

```
1 using System.Collections;
2 using System.Collections.Generic;
3 using UnityEngine;
4
5 public abstract class EstadoPartida
6 {
7     public abstract void IniciarEstado(ControlPartida controlPartida);
8     public abstract void ActualizarEstado(ControlPartida controlPartida);
9 }
```

```

1 using System.Collections;
2 using System.Collections.Generic;
3 using Photon.Pun;
4 using Unity.Mathematics;
5 using UnityEngine;
6 using Random = UnityEngine.Random;
7
8 public class GeneradorTile : MonoBehaviour
9 {
10     public int tier;
11     float radius = 1.5f;
12     public LayerMask mask;
13     Aleatoriedad random;
14     private PhotonView photonView;
15
16     List<string> sitiosYaPuestos = new List<string>();
17
18     // Start is called before the first frame update
19     void Start()
20     {
21         photonView = gameObject.AddComponent<PhotonView>();
22         PhotonNetwork.AllocateViewID(photonView);
23
24         if (ControlPartida.Instancia)
25         {
26             if (ControlPartida.Instancia.controladorSala)
27             {
28                 random = new Aleatoriedad(ControlPartida.Instancia.controladorSala.semilla);
29             }
30             else
31             {
32                 random = new Aleatoriedad(Random.Range(1000, 9999));
33             }
34         }
35         else
36         {
37             random = new Aleatoriedad(Random.Range(1000, 9999));
38         }
39     }
40
41     public void generarEnemigos()
42     {
43         string forma = "";
44         while (tier > 0)
45         {
46             float posX = random.nuevoEntero(-4, 5);
47             float posZ = random.nuevoEntero(-4, 5);
48             Vector3 position = transform.position + new Vector3(posX, 0.5f, posZ);
49             Collider[] things = Physics.OverlapSphere(position, radius, mask);
50             if (things.Length == 0)
51             {
52                 int tierBicho = random.nuevoEntero(1, tier + 1);
53                 //TODO: ajustar esto bien segun los tier de los enemigos disponibles
54                 while (tierBicho > 4)
55                 {
56                     tierBicho = random.nuevoEntero(1, tier + 1);
57                 }
58
59                 tier -= tierBicho;
60
61                 while (sitiosYaPuestos.Contains("'" + posX + "," + posZ + "'"))
62                 {
63                     posX = random.nuevoEntero(-4, 5);
64                     posZ = random.nuevoEntero(-4, 5);
65                 }
66
67                 sitiosYaPuestos.Add("'" + posX + "," + posZ + "'");
68                 GameObject g = Instantiate(ControlPartida.Instancia.controladorEnemigos.devolverEnemigo(tierBicho),
69                     transform.position + new Vector3(posX, 0f, posZ), quaternion.identity);
70
71                 forma += posX + "," + posZ + "," + tierBicho + "," + g.name + "|";
72             }
73         }
74     }
75
76     [PunRPC]
77     public void cargarEnemigos(string forma)
78     {
79         string[] torretas = forma.Split("|");
80         foreach (string torreta in torretas)
81         {
82             if (torreta.Length > 0)
83             {
84                 string[] partes = torreta.Split(",");
85
86                 float posX = float.Parse(partes[0]);
87                 float posZ = float.Parse(partes[1]);
88                 int tier = int.Parse(partes[2]);
89                 string nombre = (partes[3]);
90
91                 Instantiate(ControlPartida.Instancia.controladorEnemigos.devolverEnemigo(tier, nombre),
92                     transform.position + new Vector3(posX, 0f, posZ), quaternion.identity);
93             }
94         }
95     }
96 }

```

```
1 using System.Collections;
2 using System.Collections.Generic;
3 using UnityEngine;
4 using UnityEngine.SceneManagement;
5
6 public class PartidaEstado : EstadoPartida
7 {
8     public override void IniciarEstado(ControlPartida controlPartida)
9     {
10         controlPartida.controladorReproductorMusical.ActualizarEscuchadorGlobal(true);
11         SceneManager.LoadScene("PruebaProcedural");
12     }
13
14     public override void ActualizarEstado(ControlPartida controlPartida)
15     {
16     }
17 }
```

```
1 using System;
2 using System.Collections;
3 using System.Collections.Generic;
4 using ExitGames.Client.Photon.StructWrapping;
5 using UnityEngine;
6
7 public class ConsumibleVida : MonoBehaviour
8 {
9     AudioSource audioSource;
10
11     private void Start()
12     {
13         audioSource = GetComponent<AudioSource>();
14     }
15
16     private void OnTriggerEnter(Collider other)
17     {
18         if (other.CompareTag("Player"))
19         {
20             if (other.GetType<BoxCollider>())
21             {
22                 other.GetComponent<ControladorVida>().curarVida(20);
23                 StartCoroutine(sonido());
24                 Destroy(GetComponent<BoxCollider>());
25                 Destroy(GetComponent<MeshFilter>());
26             }
27         }
28     }
29
30     IEnumerator sonido()
31     {
32         audioSource.Play();
33         yield return new WaitForSeconds(1);
34         Destroy(this.gameObject);
35     }
36 }
```

```

1 using System;
2 using System.Collections;
3 using System.Collections.Generic;
4 using System.IO;
5 using UnityEngine;
6
7 public class ControlPartida : MonoBehaviour
8 {
9     public EstadoPartida estadoActualPartida;
10    public InicioEstado inicioEstado = new InicioEstado();
11    public MenuEstado menuEstado = new MenuEstado();
12    public LobbyEstado lobbyEstado = new LobbyEstado();
13    public PartidaEstado partidaEstado = new PartidaEstado();
14    public FinalEstado finalEstado = new FinalEstado();
15
16    //Core
17    public ControladorMejoras controladorMejoras;
18    public ControladorConsumible controladorConsumible;
19    public ControladorReproductorMusical controladorReproductorMusical;
20    public ControladorNotificaciones controladorNotificaciones;
21    public ControladorLogros controladorLogros;
22
23    //Lobby
24    public ControladorSala controladorSala;
25    public ControladorPuntos controladorPuntos;
26
27    //Partida
28    public ControladorPausa controladorPausa;
29    public ControladorEnemigos controladorEnemigos;
30    public bool mapaGenerado = false;
31    public bool sePuedeEmpezarADisparar = false;
32    public bool saMuerto = false;
33
34    private static ControlPartida instancia;
35
36    public static ControlPartida Instancia
37    {
38        get { return instancia; }
39    }
40
41    void Awake()
42    {
43        if (instancia == null)
44        {
45            instancia = this;
46            DontDestroyOnLoad(gameObject);
47        }
48        else
49        {
50            Destroy(gameObject);
51        }
52
53        if (Application.platform == RuntimePlatform.Android)
54        {
55            QualitySettings.vSyncCount = 0;
56            Application.targetFrameRate = 60;
57        }
58        else
59        {
60            QualitySettings.vSyncCount = 1;
61            Application.targetFrameRate = Screen.currentResolution.refreshRate;
62        }
63    }
64
65    // Start is called before the first frame update
66    void Start()
67    {
68        DontDestroyOnLoad(this);
69
70        if (Convert.ToBoolean(PlayerPrefs.GetInt("Cheats", 0)))
71        {
72            Application.Quit();
73        }
74    }
75
76    // Update is called once per frame
77    void Update()
78    {
79        estadoActualPartida.ActualizarEstado(this);
80    }
81
82    public void cambiarEstado(EstadoPartida estadoPartida)
83    {
84        estadoActualPartida = estadoPartida;
85        estadoActualPartida.IniciarEstado(this);
86    }
87 }
88

```

```

1 using System;
2 using System.Collections;
3 using System.Collections.Generic;
4 using System.IO;
5 using System.Linq;
6 using Photon.Pun;
7 using Unity.VisualScripting;
8 using UnityEngine;
9 using UnityEngine.AI;
10 using Random = UnityEngine.Random;
11
12 public class GeneradorMapas : MonoBehaviour
13 {
14     [SerializeField] private GameObject escenario;
15     [SerializeField] private GameObject tiles;
16     [SerializeField] private GameObject suelo;
17
18     private Dictionary<string, GameObject> tilesEscenario;
19
20     [SerializeField] private ControladorEnemigos controladorEnemigos;
21     [SerializeField] private Light luzMapa;
22     private ControladorSala controladorSala;
23     private int semilla;
24
25     public int ronda = 1;
26     private int rondaCalculo = 1;
27     public int mapaHecho = 0;
28     public bool mapeando = false;
29
30     Aleatoriedad aleatoriedad;
31
32     private List<GameObject> objetos = new List<GameObject>();
33     private Dictionary<GameObject, int> esquinas = new Dictionary<GameObject, int>();
34     private Dictionary<GameObject, int> bordes = new Dictionary<GameObject, int>();
35     string pathTilesEscenario = "Tiles/Escenario";
36     string pathTilesBordes = "Tiles/Bordes";
37     string pathTilesEsquinas = "Tiles/Esquinas";
38
39     private PhotonView photonView;
40
41
42     // Start is called before the first frame update
43     void Start()
44     {
45         photonView = GetComponent<PhotonView>();
46
47
48         objetos = Resources.LoadAll(pathTilesEscenario, typeof(GameObject))
49             .Cast<GameObject>()
50             .ToList();
51
52         esquinas = Resources.LoadAll(pathTilesEsquinas, typeof(GameObject))
53             .Cast<GameObject>()
54             .ToDictionary(m => m, m => 0);
55
56         bordes = Resources.LoadAll(pathTilesBordes, typeof(GameObject))
57             .Cast<GameObject>()
58             .ToDictionary(m => m, m => 0);
59
60         if (ControlPartida.Instancia)
61         {
62             ControlPartida.Instancia.mapaGenerado = false;
63             ControlPartida.Instancia.sePuedeEmpezarADisparar = false;
64         }
65
66
67         if (ControlPartida.Instancia)
68         {
69             if (ControlPartida.Instancia.controladorSala)
70             {
71                 controladorSala = ControlPartida.Instancia.controladorSala;
72                 aleatoriedad = new Aleatoriedad(ControlPartida.Instancia.controladorSala.semilla);
73                 StartCoroutine(generarUnMapaCorrecto());
74             }
75             else
76             {
77                 aleatoriedad = new Aleatoriedad(Random.Range(1000, 9999));
78                 StartCoroutine(generarUnMapaCorrecto());
79             }
80         }
81         else
82         {
83             aleatoriedad = new Aleatoriedad(Random.Range(1000, 9999));
84             StartCoroutine(generarUnMapaCorrecto());
85         }
86     }
87
88     IEnumerator generarUnMapaCorrecto()
89     {
90         Time.timeScale = 0.1f;
91         if (!mapeando)
92         {
93             mapaHecho = 0;
94
95             if (ControlPartida.Instancia)
96             {
97                 rondaCalculo = (ControlPartida.Instancia.controladorPuntos.ronda) * 5;
98             }
99             else
100             {
101                 rondaCalculo = ronda;

```

```

102     }
103
104     escenario.transform.eulerAngles = new Vector3(0, 0, 0);
105     Transform[] childs = tiles.GetComponentsInChildren<Transform>();
106
107     foreach (Transform child in childs)
108     {
109         if (child != tiles.transform)
110         {
111             Destroy(child.gameObject);
112         }
113     }
114
115     foreach (ControladorVida ene in FindObjectsOfType<ControladorVida>())
116     {
117         Destroy(ene.gameObject);
118     }
119
120     StartCoroutine(colocarPiezas());
121 }
122
123 yield break;
124 }
125
126 string mapa = "";
127
128 IEnumerator colocarPiezas()
129 {
130     mapa = "";
131     mapeando = true;
132     float max = 20;
133
134     if (ControlPartida.Instancia)
135     {
136         switch (ControlPartida.Instancia.controladorPuntos.ronda)
137         {
138             case < 5:
139                 break;
140             case < 15 and > 5:
141                 max += 10;
142                 break;
143             case < 30 and > 15:
144                 max += 20;
145                 break;
146         }
147     }
148
149     float y = max;
150     float x = max;
151
152     for (int i = 0; i < max * 2 / 10 + 1; i++)
153     {
154         y = max;
155         for (int j = 0; j < max * 2 / 10 + 1; j++)
156         {
157             if (Mathf.Abs(x) < max - 5 && Mathf.Abs(y) < max - 5)
158             {
159                 if (Mathf.Abs(x) < 0.5f && Mathf.Abs(y) < 0.5f)
160                 {
161                     y -= 10;
162                     continue;
163                 }
164
165                 int casillaAponer = aleatoriedad.nuevoEntero(0, objetos.Count);
166                 //TODO: limitar el valor de verdad
167
168                 int tier = aleatoriedad.nuevoEntero(0, 10);
169                 while (rondaCalculo - tier < 0)
170                 {
171                     tier = aleatoriedad.nuevoEntero(0, 10);
172                     print(rondaCalculo + " :while: " + tier);
173                 }
174
175                 print(rondaCalculo + " : " + tier);
176                 int rotacion = aleatoriedad.nuevoEntero(0, 4);
177                 rondaCalculo -= tier;
178                 mapa += x + "," + y + "," + casillaAponer + "," + tier + "," + rotacion + "|";
179             }
180             else
181             {
182                 if (x < -(max - 5))
183                 {
184                     if (y > max - 5)
185                     {
186                         int casillaAponer = aleatoriedad.nuevoEntero(0, esquinas.Count);
187                         mapa += x + "," + y + "," + casillaAponer + "," + 0 + "," + 1 + "|";
188                     }
189                     else if (y < -(max - 5))
190                     {
191                         int casillaAponer = aleatoriedad.nuevoEntero(0, esquinas.Count);
192                         mapa += x + "," + y + "," + casillaAponer + "," + 0 + "," + 0 + "|";
193                     }
194                     else
195                     {
196                         int casillaAponer = aleatoriedad.nuevoEntero(0, bordes.Count);
197                         mapa += x + "," + y + "," + casillaAponer + "," + 0 + "," + 2 + "|";
198                     }
199                 }
200                 else if (x > max - 5)
201                 {
202                     if (y > max - 5)

```

```

203         {
204             int casillaAponer = aleatoriedad.nuevoEntero(0, esquinas.Count);
205             mapa += x + "," + y + "," + casillaAponer + "," + 0 + "," + 2 + "|";
206         }
207         else if (y < -(max - 5))
208         {
209             int casillaAponer = aleatoriedad.nuevoEntero(0, esquinas.Count);
210             mapa += x + "," + y + "," + casillaAponer + "," + 0 + "," + -1 + "|";
211         }
212         else
213         {
214             int casillaAponer = aleatoriedad.nuevoEntero(0, bordes.Count);
215             mapa += x + "," + y + "," + casillaAponer + "," + 0 + "," + 0 + "|";
216         }
217     }
218     else
219     {
220         int casillaAponer = aleatoriedad.nuevoEntero(0, bordes.Count);
221
222         if (y > max - 5)
223         {
224             mapa += x + "," + y + "," + casillaAponer + "," + 0 + "," + -1 + "|";
225         }
226         else if (y < -(max - 5))
227         {
228             mapa += x + "," + y + "," + casillaAponer + "," + 0 + "," + 1 + "|";
229         }
230     }
231 }
232
233     y -= 10;
234 }
235
236     x -= 10;
237 }
238
239     yield return null;
240     StartCoroutine(colocarMapa(mapa));
241
242     yield break;
243 }
244
245 IEnumerator comprobarMapa()
246 {
247     NavMeshPath path = new NavMeshPath();
248     yield return new WaitForSecondsRealtime(0.01f);
249
250     int enemigosAlcanzables = 0;
251     ControladorVida[] enemigos = FindObjectsOfType<ControladorVida>();
252     foreach (ControladorVida vida in enemigos)
253     {
254         NavMesh.CalculatePath(transform.position, vida.transform.position, NavMesh.AllAreas, path);
255
256         if (path.status == NavMeshPathStatus.PathComplete)
257         {
258             enemigosAlcanzables++;
259         }
260     }
261
262     yield return new WaitForSecondsRealtime(0.01f);
263
264     mapeando = false;
265
266     if (rondaCalculo == 0 && enemigosAlcanzables == enemigos.Length)
267     {
268         StartCoroutine(empezarRonda());
269         yield break;
270     }
271     else
272     {
273         print(enemigosAlcanzables);
274         print(enemigos.Length);
275         StartCoroutine(generarUnMapaCorrecto());
276         yield break;
277     }
278 }
279
280 IEnumerator empezarRonda()
281 {
282     Time.timeScale = 1f;
283     ControlPartida.Instancia.mapaGenerado = true;
284     yield return new WaitForSecondsRealtime(1f);
285     controladorEnemigos.iniciar();
286     yield return new WaitForSecondsRealtime(1f);
287     ControlPartida.Instancia.sePuedeEmpezarADisparar = true;
288 }
289
290 IEnumerator colocarMapa(string mapaString)
291 {
292     mapeando = true;
293     float max = 20;
294
295     if (ControlPartida.Instancia)
296     {
297         switch (ControlPartida.Instancia.controladorPuntos.ronda)
298         {
299             case < 5:
300                 break;
301             case < 15 and > 5:
302                 max += 10;
303         }

```

```

304         luzMapa.transform.Rotate(180, 0, 0);
305         break;
306     case < 30 and > 15:
307         max += 20;
308         break;
309     }
310 }
311
312 suelo.transform.localScale = new Vector3(max * 2 / 10 + 1, max * 2 / 10 + 1, max * 2 / 10 + 1);
313
314 string[] piezas = mapaString.Split("|");
315 foreach (string pieza in piezas)
316 {
317     if (pieza.Length > 0)
318     {
319         string[] parte = pieza.Split(",");
320         int x = Int32.Parse(parte[0]);
321         int y = Int32.Parse(parte[1]);
322         int objeto = Int32.Parse(parte[2]);
323         int tier = Int32.Parse(parte[3]);
324         int rotacion = Int32.Parse(parte[4]);
325
326
327         if (Mathf.Abs(x) < max - 5 && Mathf.Abs(y) < max - 5)
328         {
329             if (Mathf.Abs(x) < 0.5f && Mathf.Abs(y) < 0.5f)
330             {
331                 y -= 10;
332                 continue;
333             }
334
335             GameObject tile = objetos.ElementAt(objeto);
336             GeneradorTile generadorTile = tile.GetComponent<GeneradorTile>();
337
338             generadorTile.tier = tier;
339
340             GameObject gameObject = Instantiate(tile,
341                 new Vector3(x, 0, y),
342                 Quaternion.identity, tiles.transform);
343
344             gameObject.transform.eulerAngles = new Vector3(0, 90 * rotacion, 0);
345         }
346         else
347         {
348             if (x < -(max - 5))
349             {
350                 if (y > max - 5)
351                 {
352                     GameObject gameObject = Instantiate(
353                         esquinas.ElementAt(objeto).Key,
354                         new Vector3(x, 0, y),
355                         Quaternion.identity, tiles.transform);
356
357                     gameObject.transform.eulerAngles = new Vector3(0, 90, 0);
358                 }
359                 else if (y < -(max - 5))
360                 {
361                     GameObject gameObject = Instantiate(
362                         esquinas.ElementAt(objeto).Key,
363                         new Vector3(x, 0, y),
364                         Quaternion.identity, tiles.transform);
365
366                     gameObject.transform.eulerAngles = new Vector3(0, 0, 0);
367                 }
368                 else
369                 {
370                     GameObject gameObject = Instantiate(
371                         bordes.ElementAt(objeto).Key,
372                         new Vector3(x, 0, y),
373                         Quaternion.identity, tiles.transform);
374
375                     gameObject.transform.eulerAngles = new Vector3(0, 180, 0);
376                 }
377             }
378             else if (x > max - 5)
379             {
380                 if (y > max - 5)
381                 {
382                     GameObject gameObject = Instantiate(
383                         esquinas.ElementAt(objeto).Key,
384                         new Vector3(x, 0, y),
385                         Quaternion.identity, tiles.transform);
386
387                     gameObject.transform.eulerAngles = new Vector3(0, 180, 0);
388                 }
389                 else if (y < -(max - 5))
390                 {
391                     GameObject gameObject = Instantiate(
392                         esquinas.ElementAt(objeto).Key,
393                         new Vector3(x, 0, y),
394                         Quaternion.identity, tiles.transform);
395
396                     gameObject.transform.eulerAngles = new Vector3(0, -90, 0);
397                 }
398                 else
399                 {
400                     GameObject gameObject = Instantiate(
401                         bordes.ElementAt(objeto).Key,
402                         new Vector3(x, 0, y),
403                         Quaternion.identity, tiles.transform);
404

```

```

405         gameObject.transform.eulerAngles = new Vector3(0, 0, 0);
406     }
407 }
408 else
409 {
410     GameObject gameObject = Instantiate(
411         bordes.ElementAt(objeto).Key,
412         new Vector3(x, 0, y),
413         Quaternion.identity, tiles.transform);
414
415     if (y > max - 5)
416     {
417         gameObject.transform.eulerAngles = new Vector3(0, -90, 0);
418     }
419     else if (y < -(max - 5))
420     {
421         gameObject.transform.eulerAngles = new Vector3(0, 90, 0);
422     }
423 }
424 }
425 }
426 }
427
428 escenario.transform.eulerAngles = new Vector3(0, 45, 0);
429
430 yield return null;
431
432 foreach (GeneradorTile generador in FindObjectsOfType<GeneradorTile>())
433 {
434     generador.generarEnemigos();
435     yield return null;
436 }
437
438 StartCoroutine(comprobarMapa());
439 }
440 }

```

```

1 using System.Collections;
2 using System.Collections.Generic;
3 using Unity.VisualScripting;
4 using UnityEngine;
5 using UnityEngine.AI;
6
7 public class MoverAIEnemigo : MonoBehaviour
8 {
9     private bool EnemigoAvistado;
10    [SerializeField] private string tagEnemigo;
11    public GameObject enemigoDisparable;
12    private float distanciaEnemigo;
13
14    private NavMeshAgent agente;
15
16    // Start is called before the first frame update
17    void Start()
18    {
19        agente = GetComponent<NavMeshAgent>();
20    }
21
22    // Update is called once per frame
23    void Update()
24    {
25        if (enemigoDisparable.IsDestroyed())
26        {
27            EnemigoAvistado = false;
28            enemigoDisparable = null;
29        }
30
31        if (EnemigoAvistado)
32        {
33            Vector3 relativePos = enemigoDisparable.transform.position - transform.position;
34            relativePos.y = 0f;
35            distanciaEnemigo = relativePos.magnitude;
36            agente.SetDestination(enemigoDisparable.transform.position);
37        }
38    }
39
40    private void OnTriggerStay(Collider other)
41    {
42        if (other.tag.Equals(tagEnemigo))
43        {
44            Vector3 relativePos = other.transform.position - transform.position;
45            relativePos.y = 0f;
46            float nuevaDistancia = relativePos.magnitude;
47            if (nuevaDistancia < distanciaEnemigo)
48            {
49                enemigoDisparable = other.gameObject;
50            }
51        }
52    }
53
54    private void OnTriggerEnter(Collider other)
55    {
56        if (other.tag.Equals(tagEnemigo))
57        {
58            EnemigoAvistado = true;
59            enemigoDisparable = other.gameObject;
60        }
61    }
62
63    private void OnTriggerExit(Collider other)
64    {
65        if (other.tag.Equals(tagEnemigo))
66        {
67            if (other.gameObject == enemigoDisparable)
68            {
69                EnemigoAvistado = false;
70                enemigoDisparable = null;
71            }
72        }
73    }
74 }
75 }

```

```

1 using System;
2 using System.Collections;
3 using System.Collections.Generic;
4 using UnityEngine;
5
6 public class ControladorBala : MonoBehaviour
7 {
8     public float tiempoVida;
9     public float maxTiempoVida = 5f;
10    public float velocidad = 4f;
11    public float dañoBala = 1f;
12    public GameObject hit;
13    public string tagEnemigo;
14    public string tagMio;
15    public TrailRenderer trail;
16    public GuiadoBalas guiado;
17    public BalaExplosiva explosiva;
18    public bool explota;
19
20    public void iniciarDatos(float maTiVida, float ve, float daBa, string taEn, string taMi, float taBa, bool gui,
21        bool explo)
22    {
23        maxTiempoVida = maTiVida;
24        velocidad = ve;
25        dañoBala = daBa;
26        tagEnemigo = taEn;
27        tagMio = taMi;
28        transform.localScale *= taBa;
29        trail.startWidth *= taBa;
30        guiado.enabled = gui;
31        explota = explo;
32    }
33
34    // Start is called before the first frame update
35    void Start()
36    {
37        trail = GetComponent<TrailRenderer>();
38        tiempoVida = maxTiempoVida;
39    }
40
41    // Update is called once per frame
42    void Update()
43    {
44        transform.Translate(Vector3.forward * (Time.deltaTime * velocidad));
45        tiempoVida -= Time.deltaTime;
46        if (tiempoVida < 0f)
47        {
48            Destroy(this.gameObject);
49        }
50    }
51
52    private void OnCollisionEnter(Collision collision)
53    {
54        if (collision.gameObject.tag.Equals(tagEnemigo))
55        {
56            if (explota)
57            {
58                explosiva.daño = dañoBala;
59                explosiva.explotar();
60            }
61            else
62            {
63                Instantiate(hit, transform.position, transform.rotation);
64                collision.gameObject.GetComponent<ControladorVida>().recibirDaño(dañoBala);
65                Destroy(this.gameObject);
66            }
67        }
68
69        if (!collision.gameObject.tag.Equals(tagMio) && !collision.gameObject.tag.Equals("Bala"))
70        {
71            if (tagMio == "Player" && collision.gameObject.tag.Equals("Escudo"))
72            {
73                return;
74            }
75
76            Destroy(this.gameObject);
77        }
78    }
79 }

```

```

1 using System;
2 using System.Collections;
3 using System.Collections.Generic;
4 using MoreMountains.Tools;
5 using Photon.Pun;
6 using Photon.Realtime;
7 using TMPPro;
8 using UnityEngine;
9 using UnityEngine.SceneManagement;
10 using UnityEngine.UI;
11 using Hashtable = ExitGames.Client.Photon.Hashtable;
12 using Random = System.Random;
13
14 public class ControladorSala : MonoBehaviourPunCallbacks
15 {
16     public GameObject jugador;
17
18     [SerializeField] private GameObject objetoBotonUnirse;
19     [SerializeField] private GameObject objetoBotonDesconectar;
20     [SerializeField] private GameObject objetoBotonEmpezar;
21
22     private Button botonUnirse;
23
24     private ControladorHover controladorHoverBotonUnirse;
25
26     [SerializeField] private GameObject objetoInputNombreSala;
27     [SerializeField] private GameObject objetoInputNombreUsuario;
28     [SerializeField] private TextMeshProUGUI textoArribaInput;
29     private TMP_InputField inputNombreSala;
30     private TMP_InputField inputNombreUsuario;
31     [SerializeField] private TextMeshProUGUI nombreSala;
32     [SerializeField] private TextMeshProUGUI genteConectada;
33     [SerializeField] private TextMeshProUGUI displayNombreUsuario;
34     [SerializeField] private TextMeshProUGUI textoError;
35     public List<string> listaGenteConectada;
36     public List<string> rutasCompartidas;
37
38     [Space] public Transform spawn_point;
39
40     private PhotonView photonView;
41
42     private bool empezoPartida;
43     // Start is called before the first frame update
44
45     public string nombreUsuario;
46     public int semilla;
47
48     public Random random;
49
50
51     void Awake()
52     {
53         if (ControlPartida.Instancia.controladorSala == null)
54         {
55             ControlPartida.Instancia.controladorSala = this;
56             DontDestroyOnLoad(gameObject);
57         }
58         else
59         {
60             Destroy(gameObject);
61         }
62     }
63
64     public void Autodestruccion()
65     {
66         Destroy(gameObject);
67     }
68
69     void Start()
70     {
71         inputNombreSala = objetoInputNombreSala.GetComponent<TMP_InputField>();
72         inputNombreUsuario = objetoInputNombreUsuario.GetComponent<TMP_InputField>();
73
74         botonUnirse = objetoBotonUnirse.GetComponent<Button>();
75         botonUnirse.interactable = false;
76         controladorHoverBotonUnirse = botonUnirse.GetComponent<ControladorHover>();
77         controladorHoverBotonUnirse.enabled = false;
78
79         textoError.gameObject.SetActive(false);
80
81         photonView = GetComponent<PhotonView>();
82
83         listaGenteConectada = new List<string>();
84
85         Debug.Log("Conectando...");
86         PhotonNetwork.ConnectUsingSettings();
87
88
89         random = new Random();
90
91         empezoPartida = false;
92     }
93
94     private void Update()
95     {
96         //Debug.Log(PhotonNetwork.NetworkClientState);
97         if (ControlPartida.Instancia.estadoActualPartida == ControlPartida.Instancia.lobbyEstado)
98         {
99             if (PhotonNetwork.IsMasterClient)
100             {
101                 objetoBotonEmpezar.SetActive(PhotonNetwork.InRoom);

```

```

102         objetoBotonDesconectar.SetActive(false);
103     }
104     else
105     {
106         objetoBotonEmpezar.SetActive(false);
107         objetoBotonDesconectar.SetActive(PhotonNetwork.InRoom);
108     }
109
110     actualizarLista();
111 }
112
113
114 public override void OnConnectedToMaster()
115 {
116     base.OnConnectedToMaster();
117     Debug.Log("Conectado al servidor");
118     botonUnirse.interactable = true;
119     controladorHoverBotonUnirse.enabled = true;
120 }
121
122 public override void OnCreatedRoom()
123 {
124     base.OnCreatedRoom();
125     semilla = random.Next(1000, 9999);
126     ;
127     PhotonNetwork.CurrentRoom.SetCustomProperties(new Hashtable
128     { { "Seed", semilla }, { "EnPartida", false } });
129     enviarRutas();
130     Debug.Log("creada una sala");
131 }
132
133 string customPropertyName = "EnPartida";
134
135 public void conectarseSala()
136 {
137     nombreUsuario = inputNombreUsuario.text.ToLower();
138     displayNombreUsuario.text = "bienvenido: " + nombreUsuario;
139
140     nombreSala.text = inputNombreSala.text.ToLower();
141     if (PhotonNetwork.NetworkClientState == ClientState.ConnectedToMasterServer)
142     {
143         PhotonNetwork.JoinLobby();
144     }
145 }
146
147 public override void OnJoinedLobby()
148 {
149     base.OnJoinedLobby();
150     PhotonNetwork.JoinOrCreateRoom(nombreSala.text, null, null);
151     Debug.Log("Conectados y unidos a un lobby " + nombreSala.text);
152 }
153
154 public override void OnJoinedRoom()
155 {
156     if (PhotonNetwork.CurrentRoom.CustomProperties.ContainsKey("EnPartida"))
157     {
158         bool enPartida = (bool)PhotonNetwork.CurrentRoom.CustomProperties["EnPartida"];
159
160         if (enPartida)
161         {
162             Debug.Log("Ya hay una partida en curso");
163             PhotonNetwork.LeaveRoom();
164             textoError.gameObject.SetActive(true);
165             return;
166         }
167     }
168
169     base.OnJoinedRoom();
170     textoError.gameObject.SetActive(false);
171
172     textoArribaInput.text = "unido a:";
173     objetoInputNombreSala.SetActive(false);
174     objetoInputNombreUsuario.SetActive(false);
175     objetoBotonUnirse.SetActive(false);
176
177     if (PhotonNetwork.CurrentRoom.CustomProperties.ContainsKey("Seed"))
178     {
179         int seed = (int)PhotonNetwork.CurrentRoom.CustomProperties["Seed"];
180
181         semilla = seed;
182     }
183
184     Debug.Log("Conectados y unidos a una sala");
185     photonView.RPC("aniadirNombre", RpcTarget.AllBufferedViaServer, nombreUsuario);
186 }
187
188 private void OnApplicationQuit()
189 {
190     Debug.Log("Cerrado");
191     if (PhotonNetwork.IsConnected)
192     {
193         photonView.RPC("quitarNombre", RpcTarget.AllBufferedViaServer, nombreUsuario);
194     }
195 }
196
197 public void desconectar()
198 {
199     photonView.RPC("quitarNombre", RpcTarget.AllBufferedViaServer, nombreUsuario);
200     Debug.Log("Desconectar...");
201     if (PhotonNetwork.InRoom)
202     {

```

```

203         Debug.Log("Cerrando");
204         PhotonNetwork.LeaveRoom();
205     }
206 }
207
208 public void volverOffline()
209 {
210     PhotonNetwork.Disconnect();
211 }
212
213
214 public override void OnLeftRoom()
215 {
216     base.OnLeftRoom();
217     Debug.Log("Saliendo de la sala...");
218     if (!empezoPartida && ControlPartida.Instancia.estadoActualPartida == ControlPartida.Instancia.lobbyEstado)
219     {
220         listaGenteConectada.Clear();
221         actualizarLista();
222         textoArribaInput.text = "unirse a:";
223         objetoInputNombreSala.SetActive(true);
224         displayNombreUsuario.text = "Escribe el nombre: ";
225         objetoInputNombreUsuario.SetActive(true);
226         objetoBotonUnirse.SetActive(true);
227         objetoBotonDesconectar.SetActive(false);
228     }
229 }
230
231 [PunRPC]
232 public void quitarNombre(string nombre)
233 {
234     Debug.Log("Quitando nombre...");
235     listaGenteConectada.Remove(nombre);
236     actualizarLista();
237     if (nombre == nombreUsuario)
238     {
239         Debug.Log("Desconectado");
240     }
241     else
242     {
243         Debug.Log("Quitado");
244     }
245 }
246
247 [PunRPC]
248 public void aniadirNombre(string nombre)
249 {
250     listaGenteConectada.Add(nombre);
251     actualizarLista();
252 }
253
254 public void actualizarLista()
255 {
256     genteConectada.text = "";
257     foreach (string nombre in listaGenteConectada)
258     {
259         genteConectada.text += Environment.NewLine + nombre;
260     }
261 }
262
263 [PunRPC]
264 public void cargarEscenaPartida(string a)
265 {
266     ControlPartida.Instancia.cambiarEstado(ControlPartida.Instancia.partidaEstado);
267     empezoPartida = true;
268 }
269
270 [PunRPC]
271 public void cargarFinalPartida()
272 {
273     // ControlPartida.Instancia.cambiarEstado(ControlPartida.Instancia.partidaEstado);
274     SceneManager.LoadScene("EndGame");
275     empezoPartida = true;
276 }
277
278 public void empezarPartida()
279 {
280     PhotonNetwork.CurrentRoom.SetCustomProperties(new Hashtable
281     { { "EnPartida", true } });
282     photonView.RPC("cargarEscenaPartida", RpcTarget.AllViaServer, "as");
283 }
284
285 public void acabarPartida()
286 {
287     photonView.RPC("cargarFinalPartida", RpcTarget.AllBufferedViaServer);
288 }
289
290
291 public int genteLista = 0;
292
293 [PunRPC]
294 public void usuarioListo()
295 {
296     genteLista++;
297     if (listaGenteConectada.Count == (genteLista))
298     {
299         photonView.RPC("cargarEscenaPartida", RpcTarget.AllBufferedViaServer, "PruebaProcedural");
300         genteLista = 0;
301     }
302 }
303

```

```
304 public void usuarioLocalListo()
305 {
306     photonView.RPC("usuarioListo", RpcTarget.AllBufferedViaServer);
307 }
308
309 public void enviarRutas()
310 {
311     string rutas = "Ruta base,";
312     if (Convert.ToBoolean(PlayerPrefs.GetInt("Ruta base plus", 0))) rutas += "Ruta base plus,";
313     if (Convert.ToBoolean(PlayerPrefs.GetInt("Ruta ciencia", 0))) rutas += "Ruta ciencia,";
314     if (Convert.ToBoolean(PlayerPrefs.GetInt("Ruta ciencia plus", 0))) rutas += "Ruta ciencia plus,";
315     if (Convert.ToBoolean(PlayerPrefs.GetInt("Ruta divina", 0))) rutas += "Ruta divina,";
316     if (Convert.ToBoolean(PlayerPrefs.GetInt("Ruta her6ica", 0))) rutas += "Ruta her6ica,";
317     photonView.RPC("compartirRutas", RpcTarget.AllBufferedViaServer, rutas);
318 }
319
320 [PunRPC]
321 public void compartirRutas(string rutas)
322 {
323     rutasCompartidas = new List<string>();
324     foreach (string ruta in rutas.Split(","))
325     {
326         if (ruta.Length > 0)
327         {
328             rutasCompartidas.Add(ruta);
329         }
330     }
331
332     ControlPartida.Instancia.controladorMejoras.ReiniciarPoolMejoras();
333 }
334 }
```

```

1 using System;
2 using System.Collections;
3 using System.Collections.Generic;
4 using Photon.Pun;
5 using UnityEngine;
6 using UnityEngine.Events;
7 using Random = UnityEngine.Random;
8
9 public class ControladorVida : MonoBehaviour
10 {
11     public float vida;
12
13     public float vidaMaxima = 100f;
14
15     public UnityEvent cuandoRecibaImaecto;
16
17     System.Random random;
18
19     void Start()
20     {
21         vida = vidaMaxima;
22
23         if (ControlPartida.Instancia.controladorSala)
24         {
25             random = new System.Random(ControlPartida.Instancia.controladorSala.semilla);
26         }
27         else
28         {
29             random = new System.Random(Random.Range(0, 99999));
30         }
31     }
32
33     public void curarVida(float puntosARecuperar)
34     {
35         vida += puntosARecuperar;
36         if (vida > vidaMaxima)
37         {
38             vida = vidaMaxima;
39         }
40     }
41
42     public void recibirDaño(float daño)
43     {
44         vida -= daño;
45         cuandoRecibaImaecto?.Invoke();
46
47         if (vida < 0)
48         {
49             if (this.gameObject.tag.Equals("Enemigo"))
50             {
51                 int puntosQueDa = 0;
52                 ApuntarAlEnemigo stats = GetComponentInChildren<ApuntarAlEnemigo>();
53                 float dps = stats.daño * stats.balasADisparar / stats.tasaDisparo;
54                 puntosQueDa = (int)dps + (int)vidaMaxima;
55                 ControlPartida.Instancia.controladorPuntos.aumentarPuntos(puntosQueDa);
56                 FindObjectOfType<ControladorEnemigos>().enemigos.Remove(this);
57
58                 if (90 < random.Next(0, 100))
59                 {
60                     Instantiate(
61                         ControlPartida.Instancia.controladorConsumible.devolverConsumible(),
62                         transform.position, Quaternion.identity);
63                 }
64             }
65
66             if (this.gameObject.tag.Equals("Player"))
67             {
68                 FindObjectOfType<ControladorPausa>().seMurioAlguien();
69             }
70
71             Destroy(this.gameObject);
72         }
73     }
74 }

```

```

1 using System;
2 using System.Collections;
3 using System.Collections.Generic;
4 using ExitGames.Client.Photon.StructWrapping;
5 using Photon.Pun;
6 using Unity.VisualScripting;
7 using UnityEngine;
8 using UnityEngine.AI;
9 using Random = UnityEngine.Random;
10
11 public class ApuntarAlEnemigo : MonoBehaviour
12 {
13     private bool EnemigoAvistado;
14     [SerializeField] private List<Transform> spawnerBalas;
15     private int spawnerEnUso = 0;
16     private int balaEnUso = 0;
17     [SerializeField] public List<GameObject> tiposBalas;
18     [SerializeField] public float tasaDisparo;
19     [SerializeField] public float rangoDisparo;
20     [SerializeField] public float dispersion;
21     [SerializeField] public float tiempoVidaBala;
22     private bool disparando = false;
23     [SerializeField] private string tagEnemigo;
24     public float daño;
25     public float velocidadBala;
26     public float tamañoBala;
27     public float balasADisparar = 1;
28     private GameObject enemigoDisparable;
29     private float distanciaEnemigo;
30     public bool balasGuiadas = false;
31     public bool balasExplosivas = false;
32     private List<GameObject> enemigosEnRango;
33
34     private NavMeshAgent agente;
35
36     PhotonView photonView;
37
38     [SerializeField] private AudioSource sonidoDisparar;
39     Aleatoriedad random;
40     private SphereCollider _sphereCollider;
41
42     // Start is called before the first frame update
43     void Start()
44     {
45         _sphereCollider = GetComponent<SphereCollider>();
46         photonView = GetComponent<PhotonView>();
47         GetComponent<SphereCollider>().radius = rangoDisparo;
48         enemigosEnRango = new List<GameObject>();
49
50         if (ControlPartida.Instancia)
51         {
52             if (ControlPartida.Instancia.controladorSala)
53             {
54                 random = new Aleatoriedad(ControlPartida.Instancia.controladorSala.semilla);
55             }
56             else
57             {
58                 random = new Aleatoriedad(Random.Range(1000, 9999));
59             }
60         }
61         else
62         {
63             random = new Aleatoriedad(Random.Range(1000, 9999));
64         }
65
66         if (GetComponent<Rigidbody>())
67         {
68             agente = GetComponent<NavMeshAgent>();
69         }
70     }
71
72     // Update is called once per frame
73     void Update()
74     {
75         _sphereCollider.radius = rangoDisparo;
76         enemigoDisparable = null;
77         EnemigoAvistado = false;
78         distanciaEnemigo = float.MaxValue;
79
80         foreach (GameObject o in enemigosEnRango)
81         {
82             if (o.IsDestroyed())
83             {
84                 enemigosEnRango.Remove(o);
85                 break;
86             }
87
88             Vector3 relativePos = o.transform.position - transform.position;
89             relativePos.y = 0f;
90             float nuevaDistancia = relativePos.magnitude;
91
92             if (nuevaDistancia > rangoDisparo) continue;
93             if (!(nuevaDistancia < distanciaEnemigo)) continue;
94
95             if (Physics.Raycast(transform.position, relativePos, out RaycastHit hit, nuevaDistancia,
96                 LayerMask.GetMask("Default"), QueryTriggerInteraction.Ignore)) continue;
97
98             EnemigoAvistado = true;
99             distanciaEnemigo = nuevaDistancia;
100             enemigoDisparable = o.gameObject;
101         }

```

```

102
103
104     if (EnemigoAvistado)
105     {
106         Vector3 relativePos = enemigoDisparable.transform.position - transform.position;
107         relativePos.y = 0f;
108         distanciaEnemigo = relativePos.magnitude;
109         Quaternion rotation = Quaternion.LookRotation(relativePos, Vector3.up);
110         transform.rotation = rotation;
111     }
112
113     if (ControlPartida.Instancia)
114     {
115         if (EnemigoAvistado && !disparando && ControlPartida.Instancia.sePuedeEmpezarADisparar)
116         {
117             StartCoroutine(Disparar());
118         }
119     }
120     else if (EnemigoAvistado && !disparando)
121     {
122         StartCoroutine(Disparar());
123     }
124 }
125
126 public IEnumerator Disparar()
127 {
128     disparando = true;
129     spawnerEnUso++;
130     if (spawnerEnUso >= spawnerBalas.Count)
131     {
132         spawnerEnUso = 0;
133     }
134
135     balaEnUso--;
136     if (balaEnUso < 0)
137     {
138         balaEnUso = tiposBalas.Count - 1;
139     }
140
141     for (int i = 1; i <= balasADisparar; i++)
142     {
143         GameObject balaObjeto = GameObject.Instantiate(tiposBalas[balaEnUso], spawnerBalas[spawnerEnUso].position,
144             spawnerBalas[spawnerEnUso].rotation * Quaternion.Euler(0,
145                 random.nuevoDecimal(-dispersion, dispersion), 0));
146
147         float mivelosidad = velocidadBala;
148         if (agente)
149         {
150             Quaternion rotation = transform.rotation;
151             Vector3 baseDirection = Vector3.forward;
152             Vector3 direction = rotation * baseDirection;
153
154
155             float productoPunto = Vector3.Dot(agente.velocity.normalized, direction.normalized);
156
157             mivelosidad += productoPunto * agente.velocity.magnitude / 2f;
158         }
159
160         balaObjeto.GetComponent<ControladorBala>()
161             .iniciarDatos(tiempoVidaBala, mivelosidad, daño, tagEnemigo, tag,
162                 tamañoBala, balasGuiadas, balasExplosivas);
163
164         spawnerEnUso++;
165         if (spawnerEnUso >= spawnerBalas.Count)
166         {
167             spawnerEnUso = 0;
168         }
169     }
170
171     if (sonidoDisparar)
172     {
173         sonidoDisparar.pitch = random.nuevoDecimal(0.98f, 1.05f);
174         sonidoDisparar.Play();
175     }
176
177     yield return new WaitForSeconds(tasaDisparo);
178     disparando = false;
179 }
180
181
182 private void OnTriggerEnter(Collider other)
183 {
184     if (other.IsType<SphereCollider>())
185     {
186         return;
187     }
188
189     if (other.tag.Equals(tagEnemigo))
190     {
191         enemigosEnRango.Add(other.gameObject);
192     }
193 }
194
195 private void OnTriggerExit(Collider other)
196 {
197     if (other.IsType<SphereCollider>())
198     {
199         return;
200     }
201
202     if (other.tag.Equals(tagEnemigo))

```

```
203         {  
204             enemigosEnRango.Remove(other.gameObject);  
205         }  
206     }  
207 }
```

```

1 using System;
2 using System.Collections;
3 using System.Collections.Generic;
4 using ExitGames.Client.Photon.StructWrapping;
5 using UnityEngine;
6 using UnityEngine.UI;
7 using Random = UnityEngine.Random;
8
9 public class ConsumibleMejora : MonoBehaviour
10 {
11     public Mejora mejora;
12     private bool seleccionada = false;
13     AudioSource audioSource;
14
15     private void Start()
16     {
17         audioSource = GetComponent();
18
19         if (ControlPartida.Instancia.controladorMejoras.poolMejoras.Count > 0)
20         {
21             mejora = ControlPartida.Instancia.controladorMejoras.poolMejoras[
22                 Random.Range(0, ControlPartida.Instancia.controladorMejoras.poolMejoras.Count)];
23             ControlPartida.Instancia.controladorMejoras.poolMejoras.Remove(mejora);
24         }
25         else
26         {
27             mejora = ControlPartida.Instancia.controladorMejoras.baseMejoras["Analisi sintactico"];
28         }
29     }
30
31
32
33     private void OnTriggerEnter(Collider other)
34     {
35         if (other.CompareTag("Player"))
36         {
37             if (other.IsType<BoxCollider>())
38             {
39                 seleccionada = true;
40                 ControlPartida.Instancia.controladorMejoras.ActivarMejoraInGame(mejora);
41                 ControlPartida.Instancia.controladorNotificaciones.enviarNotificacion(mejora.name, mejora.descripcion,
42                     mejora.icono);
43                 StartCoroutine(sonido());
44                 Destroy(GetComponent<BoxCollider>());
45                 Destroy(GetComponent<MeshFilter>());
46             }
47         }
48     }
49
50     private void OnDisable()
51     {
52         if (!seleccionada && mejora.name != "Analisi sintactico")
53         {
54             ControlPartida.Instancia.controladorMejoras.poolMejoras.Add(mejora);
55         }
56     }
57
58     IEnumerator sonido()
59     {
60         audioSource.Play();
61         yield return new WaitForSeconds(1);
62         Destroy(this.gameObject);
63     }
64 }

```

```

1 using System;
2 using System.Collections;
3 using System.Collections.Generic;
4 using LootLocker.Requests;
5 using TMPro;
6 using UnityEngine;
7 using UnityEngine.UI;
8
9 public class ControladorFinal : MonoBehaviour
10 {
11     // Start is called before the first frame update
12     // Start is called before the first frame update
13
14     [SerializeField] private TextMeshProUGUI textoPuntos;
15     [SerializeField] private TMP_InputField nombre;
16     [SerializeField] private TextMeshProUGUI info;
17     [SerializeField] private TextMeshProUGUI puntos;
18     [SerializeField] private TextMeshProUGUI textoFinal;
19     [SerializeField] private Image fondoFinal;
20     [SerializeField] private Button botonEnviar;
21
22     int leaderboardID = 10024;
23     private string memberID;
24     int prescore = 1000;
25     private int miradoDesde = 0;
26     private int posicionJugador = 0;
27
28     private void Start()
29     {
30         LootLockerSDKManager.StartGuestSession((response) =>
31         {
32             if (!response.success)
33             {
34                 Debug.Log("error starting LootLocker session");
35
36                 return;
37             }
38
39             LootLockerSDKManager.GetScoreList(leaderboardID, 1000, miradoDesde, cargarLootLocker);
40         });
41
42         ControlPartida.Instancia.controladorPuntos.puntos += ControlPartida.Instancia.controladorPuntos.puntosPartida;
43         textoPuntos.text = "Puntos: " + ControlPartida.Instancia.controladorPuntos.puntosPartida;
44         prescore = ControlPartida.Instancia.controladorPuntos.puntosPartida;
45
46
47         ControlPartida.Instancia.controladorLogros.comprobarLogros();
48
49         if (ControlPartida.Instancia.controladorSala)
50         {
51             nombre.text = ControlPartida.Instancia.controladorSala.nombreUsuario;
52         }
53
54         if (ControlPartida.Instancia.saMuerto)
55         {
56             textoFinal.text = "Has muerto";
57             fondoFinal.color = new Color(0.9921569f, 0.3411765f, 0.282353f);
58         }
59     }
60
61     public void cargarLootLocker(LootLockerGetScoreListResponse response)
62     {
63         if (response.items[^1].score > prescore)
64         {
65             miradoDesde += 1000;
66             LootLockerSDKManager.GetScoreList(leaderboardID, 1000, miradoDesde, cargarLootLocker);
67             return;
68         }
69
70         foreach (LootLockerLeaderboardMember member in response.items)
71         {
72             if (member.score >= prescore)
73             {
74                 posicionJugador = member.rank + 1;
75             }
76         }
77
78         if (posicionJugador - 5 <= 0)
79         {
80             posicionJugador = 0;
81         }
82
83         LootLockerSDKManager.GetScoreList(leaderboardID, 10, posicionJugador - 5, cargarListaLootLocker);
84     }
85
86     public void cargarListaLootLocker(LootLockerGetScoreListResponse response)
87     {
88         puntos.text = "";
89         foreach (LootLockerLeaderboardMember member in response.items)
90         {
91             puntos.text += member.rank + ":" + member.member_id + " " + member.score + Environment.NewLine;
92         }
93     }
94
95
96     public void cargarLootLockerPostEnvio(LootLockerGetScoreListResponse response)
97     {
98         puntos.text = "";
99         foreach (LootLockerLeaderboardMember member in response.items)
100         {
101             if (member.member_id == memberID &&

```

```

102         member.score == ControlPartida.Instancia.controladorPuntos.puntosPartida)
103     {
104         puntos.text += "<color=#f7fff7>" + member.rank + ":" + member.member_id + " " + member.score +
105             Environment.NewLine + "<color=#FFBE0B>";
106         continue;
107     }
108
109     puntos.text += member.rank + ":" + member.member_id + " " + member.score + Environment.NewLine;
110 }
111 }
112
113 public void enviar()
114 {
115     if (ControlPartida.Instancia.controladorPuntos.puntosPartida < 0)
116     {
117         PlayerPrefs.SetInt("Cheats", 1);
118         Application.Quit();
119         return;
120     }
121
122     botonEnviar.interactable = false;
123     botonEnviar.GetComponent<ControladorHover>().enabled = false;
124
125     ControlPartida.Instancia.controladorMejoras.ReiniciarPoolMejoras();
126
127     memberID = nombre.text;
128     int score = ControlPartida.Instancia.controladorPuntos.puntosPartida;
129
130     if (score != prescore)
131     {
132         PlayerPrefs.SetInt("Cheats", 1);
133         Application.Quit();
134         return;
135     }
136
137     LootLockerSDKManager.StartGuestSession((response) =>
138     {
139         if (!response.success)
140         {
141             Debug.Log("error starting LootLocker session");
142
143             return;
144         }
145
146         Debug.Log("successfully started LootLocker session");
147
148         LootLockerSDKManager.SubmitScore(memberID, score, leaderboardID, (response2) =>
149         {
150             if (response2.statusCode == 200)
151             {
152                 Debug.Log("Successful");
153                 info.text = "Subido correctamente";
154                 botonEnviar.GetComponentInChildren<TextMeshProUGUI>().text = "Subido";
155                 LootLockerSDKManager.StartGuestSession((response) =>
156                 {
157                     if (!response.success)
158                     {
159                         Debug.Log("error starting LootLocker session");
160
161                         return;
162                     }
163
164                     LootLockerSDKManager.GetScoreList(leaderboardID, 10, posicionJugador - 5,
165                         cargarLootLockerPostEnvio);
166                 });
167             }
168             else
169             {
170                 Debug.Log("failed: " + response2.Error);
171                 info.text = "Hubo un error vuelve a intentarlo";
172                 botonEnviar.GetComponentInChildren<TextMeshProUGUI>().text = "Reenviar";
173                 botonEnviar.interactable = true;
174                 botonEnviar.GetComponent<ControladorHover>().enabled = true;
175             }
176         });
177     });
178 }
179 }

```

```

1 using System;
2 using System.Collections;
3 using System.Collections.Generic;
4 using MoreMountains.Feedbacks;
5 using Photon.Pun;
6 using UnityEngine;
7 using UnityEngine.SceneManagement;
8
9 public class ControladorPausa : MonoBehaviour
10 {
11     [SerializeField] private GameObject menuPausa;
12     [SerializeField] private GameObject menuMejora;
13     [SerializeField] private GameObject gui;
14     [SerializeField] private MMF_Player feedback;
15     public bool pausado = false;
16     public bool finRonda = false;
17     public bool carga = false;
18
19     PhotonView photonView;
20
21     private void Start()
22     {
23         photonView = GetComponent<PhotonView>();
24         ControlPartida.Instancia.controladorPausa = this;
25         menuPausa.SetActive(false);
26         menuMejora.SetActive(false);
27         gui.SetActive(true);
28         modoCarga(false);
29     }
30
31     private void Update()
32     {
33         if (Input.GetKeyDown(KeyCode.Escape) && !finRonda)
34         {
35             if (pausado)
36             {
37                 reanudar();
38             }
39             else
40             {
41                 pausar();
42             }
43         }
44     }
45
46     public void modoCarga(bool carga)
47     {
48         carga = carga;
49         if (FindObjectsOfType<FakeCargar>().Length > 0)
50         {
51             foreach (FakeCargar fakeCargar in FindObjectsOfType<FakeCargar>())
52             {
53                 fakeCargar.gameObject.SetActive(carga);
54             }
55         }
56     }
57
58     public void finDeRonda()
59     {
60         finRonda = true;
61         StartCoroutine(efectoFinalRonda());
62     }
63
64     IEnumerator efectoFinalRonda()
65     {
66         print("Aaaaaaaaaaaaaaaaaaaaaa");
67         feedback.PlayFeedbacks();
68         Time.timeScale = 0.1f;
69         gui.SetActive(false);
70         yield return new WaitForSecondsRealtime(0.3f);
71         Time.timeScale = 1f;
72         menuMejora.SetActive(true);
73     }
74
75     public void pausar()
76     {
77         if (ControlPartida.Instancia.controladorSala)
78         {
79             photonView.RPC("alternarPausa", RpcTarget.AllBufferedViaServer, true);
80         }
81         else
82         {
83             alternarPausa(true);
84         }
85     }
86
87     public void reanudar()
88     {
89         if (ControlPartida.Instancia.controladorSala)
90         {
91             photonView.RPC("alternarPausa", RpcTarget.AllBufferedViaServer, false);
92         }
93         else
94         {
95             alternarPausa(false);
96         }
97     }
98
99     public void seMurioAlguien()
100     {
101         if (ControlPartida.Instancia.controladorSala)

```

```
102     {
103         photonView.RPC("alternarMorir", RpcTarget.AllBufferedViaServer);
104     }
105     else
106     {
107         alternarMorir();
108     }
109 }
110
111 [PunRPC]
112 public void alternarPausa(bool pausa)
113 {
114     if (!pausa)
115     {
116         Time.timeScale = 1;
117     }
118     else
119     {
120         Time.timeScale = 0.1f;
121     }
122
123     foreach (ControladorPausa controladorPausa in FindObjectsOfType<ControladorPausa>())
124     {
125         controladorPausa.pausado = pausa;
126     }
127
128     pausado = pausa;
129     menuPausa.SetActive(pausa);
130     gui.SetActive(!pausa);
131     if (!pausado)
132     {
133         for (int i = 0; i < SceneManager.sceneCount; ++i)
134         {
135             var scene = SceneManager.GetSceneAt(i);
136
137             if (scene.name == "Ajustes")
138             {
139                 SceneManager.UnloadSceneAsync("Ajustes");
140             }
141         }
142     }
143 }
144
145 [PunRPC]
146 public void alternarMorir()
147 {
148     ControlPartida.Instancia.saMuerto = true;
149     ControlPartida.Instancia.cambiarEstado(ControlPartida.Instancia.finalEstado);
150 }
151 }
```

```

1 using System;
2 using System.Collections;
3 using System.Collections.Generic;
4 using Unity.VisualScripting;
5 using UnityEngine;
6 using UnityEngine.EventSystems;
7 using UnityEngine.UI;
8
9 public class ControladorClicks : MonoBehaviour
10 {
11     // Start is called before the first frame update
12     //transform para el sprite de la mira
13     //para que sea asignable en el editor
14     [SerializeField] Transform marca;
15     [SerializeField] Transform nave;
16     [SerializeField] public Joystick Joystick;
17
18     private ControladorPausa controladorPausa;
19
20     [SerializeField] Camera camara;
21     [SerializeField] Transform _transformCamara;
22     [SerializeField] private float alturaCamara = 10;
23     public float velocidadCamara;
24
25     public Vector2 poseClickRelativa;
26     public Vector2 poseMouseRelativa;
27
28     Vector3 prevMousePos = Vector3.zero;
29
30     private bool primerClick3 = true;
31     private bool camaraFija = true;
32
33     private Vector3 _mousePos;
34
35     [SerializeField] GraphicRaycaster raycaster;
36     PointerEventData pointerEventData;
37     EventSystem eventSystem;
38
39     // Start is called before the first frame update
40     void Start()
41     {
42         controladorPausa = FindObjectOfType<ControladorPausa>();
43         eventSystem = FindObjectOfType<EventSystem>();
44     }
45
46     // Update is called once per frame
47
48
49     // Update is called once per frame
50     void Update()
51     {
52         if (Input.GetKeyDown(KeyCode.Mouse2))
53         {
54             _mousePos = Input.mousePosition;
55
56             prevMousePos = _mousePos;
57         }
58
59         if (Input.GetKeyDown(KeyCode.Y))
60         {
61             camaraFija = !camaraFija;
62         }
63
64         if (Input.GetKey(KeyCode.Mouse2))
65         {
66             camaraFija = false;
67             _mousePos = Input.mousePosition;
68
69             {
70                 //Debug.Log(poseClickRelativa);
71
72                 Vector3 moveMouse = _mousePos - prevMousePos;
73                 Vector3 moveMouseRelativa =
74                     new Vector3(moveMouse.x / Screen.width, moveMouse.z, moveMouse.y / Screen.height);
75
76                 _transformCamara.position += moveMouseRelativa * (Time.deltaTime * velocidadCamara);
77             }
78         }
79     }
80
81     void LateUpdate()
82     {
83         {
84             _mousePos = Input.mousePosition;
85             {
86                 poseMouseRelativa.x = _mousePos.x / Screen.width;
87                 poseMouseRelativa.y = _mousePos.y / Screen.height;
88             }
89
90             if (poseMouseRelativa.x > 0.95f)
91             {
92                 _transformCamara.position += new Vector3(1, 0, 0) * (Time.deltaTime * velocidadCamara);
93             }
94
95             if (poseMouseRelativa.x < 0.05f)
96             {
97                 _transformCamara.position -= new Vector3(1, 0, 0) * (Time.deltaTime * velocidadCamara);
98             }
99
100             if (poseMouseRelativa.y > 0.95f)
101             {

```

```

102     _transformCamara.position += new Vector3(0, 0, 1) * (Time.deltaTime * velocidadCamara);
103 }
104
105 if (poseMouseRelativa.y < 0.05f)
106 {
107     _transformCamara.position -= new Vector3(0, 0, 1) * (Time.deltaTime * velocidadCamara);
108 }
109
110 if ((Input.GetKey(KeyCode.Space) || camaraFija) && !nave.IsDestroyed())
111 {
112     Vector3 positionMarca = nave.position;
113     _transformCamara.position = new Vector3(positionMarca.x, alturaCamara,
114         positionMarca.z - Mathf.Tan(Mathf.PI / 4) * alturaCamara);
115 }
116
117
118 if (Input.GetButton("Fire1") && Application.platform != RuntimePlatform.Android)
119 {
120     {
121         poseClickRelativa.x = _mousePos.x / Screen.width;
122         poseClickRelativa.y = _mousePos.y / Screen.height;
123     }
124
125     pointerEventData = new PointerEventData(eventSystem);
126     pointerEventData.position = Input.mousePosition;
127     List<RaycastResult> results = new List<RaycastResult>();
128     raycaster.Raycast(pointerEventData, results);
129     foreach (RaycastResult result in results)
130     {
131         if (result.gameObject.GetComponent<Button>())
132         {
133             return;
134         }
135     }
136
137     Ray rayo_desde_camara = camara.ViewportPointToRay(new Vector3(poseClickRelativa.x, poseClickRelativa.y, 0));
138
139     if (!controladorPausa.pausado && Physics.Raycast(rayo_desde_camara, out RaycastHit hit, alturaCamara * 2,
140         LayerMask.GetMask("Suelo")))
141     {
142         //Debug.Log(hit.point);
143         if (Physics.Raycast(hit.point, Vector3.up, out RaycastHit HitCielo))
144         {
145             return;
146         }
147     }
148
149     marca.position = hit.point + new Vector3(0, 0.1f, 0);
150 }
151
152
153 if (Input.GetAxis("Horizontal") != 0 || Input.GetAxis("Vertical") != 0)
154 {
155     marca.position = nave.position +
156         new Vector3(Input.GetAxis("Horizontal"), 0, Input.GetAxis("Vertical")) * 2;
157 }
158
159 Joystick.enabled = (Application.platform == RuntimePlatform.Android);
160 if (Joystick.Direction.magnitude != 0)
161 {
162     marca.position = nave.position +
163         new Vector3(Joystick.Direction.x, 0, Joystick.Direction.y);
164 }
165 }
166 }

```

```

1 using System;
2 using System.Collections;
3 using System.Collections.Generic;
4 using TMPro;
5 using Unity.VisualScripting;
6 using UnityEditor;
7 using UnityEngine;
8 using UnityEngine.Serialization;
9 using UnityEngine.UI;
10 using Random = UnityEngine.Random;
11
12 public class ControladorMejora : MonoBehaviour
13 {
14     [SerializeField] private List<Color> listaColorTier;
15     [SerializeField] private List<Color> listaColorTextoTier;
16
17     [SerializeField] private List<Image> listaIconosStats;
18     [SerializeField] private List<Image> listaStats;
19     [SerializeField] private List<Image> listaStatsB;
20     [SerializeField] private Sprite spriteSube;
21     [SerializeField] private Sprite spriteBaja;
22
23     [SerializeField] private Mejora mejora;
24
25     [SerializeField] private Image imagenFondo;
26     [SerializeField] private TextMeshProUGUI titulo;
27     [SerializeField] private TextMeshProUGUI descripcion;
28     [SerializeField] private TextMeshProUGUI siguienteMejora;
29     [SerializeField] private TextMeshProUGUI siguienteMejoraTitulo;
30     [SerializeField] private Image icono;
31     [SerializeField] private Image estrellasTier;
32     private bool seleccionada = false;
33
34     private Aleatoriedad aleatoriedad;
35
36
37     // Start is called before the first frame update
38     void OnEnable()
39     {
40         if (ControlPartida.Instancia.controladorSala)
41         {
42             aleatoriedad = new Aleatoriedad(ControlPartida.Instancia.controladorSala.semilla);
43         }
44         else
45         {
46             aleatoriedad = new Aleatoriedad(Random.Range(1000, 9999));
47         }
48
49         if (ControlPartida.Instancia.controladorMejoras.poolMejoras.Count > 0)
50         {
51             mejora = ControlPartida.Instancia.controladorMejoras.poolMejoras[
52                 aleatoriedad.nuevoEntero(0, ControlPartida.Instancia.controladorMejoras.poolMejoras.Count)];
53             ControlPartida.Instancia.controladorMejoras.poolMejoras.Remove(mejora);
54         }
55         else
56         {
57             mejora = ControlPartida.Instancia.controladorMejoras.baseMejoras["Analisi sintactico"];
58         }
59
60
61         imagenFondo.color = (listaColorTier[mejora.tierMejora]);
62
63         titulo.color = (listaColorTextoTier[mejora.tierMejora]);
64         titulo.text = (mejora).name;
65
66         descripcion.color = listaColorTextoTier[mejora.tierMejora];
67         descripcion.text = (mejora).descripcion;
68
69         icono.sprite = mejora.icono;
70         //icono.color = listaColorTextoTier[mejora.tierMejora];
71
72         estrellasTier.color = listaColorTextoTier[mejora.tierMejora];
73
74         Vector2 size = estrellasTier.rectTransform.sizeDelta;
75         size.x = 100f * (mejora.tierMejora + 1);
76         estrellasTier.rectTransform.sizeDelta = size;
77
78         estrellasTier.transform.localPosition = new Vector3(0, estrellasTier.transform.localPosition.y,
79             estrellasTier.transform.localPosition.z);
80
81         foreach (Image iconosStat in listaIconosStats)
82         {
83             iconosStat.color = listaColorTextoTier[mejora.tierMejora];
84         }
85
86         actualizarStat(listaStats[0], listaStatsB[0], mejora.modAtq);
87         actualizarStat(listaStats[1], listaStatsB[1], mejora.modRango);
88         actualizarStat(listaStats[2], listaStatsB[2], mejora.modVida);
89         actualizarStat(listaStats[3], listaStatsB[3], mejora.modVel);
90         actualizarStat(listaStats[4], listaStatsB[4], mejora.modAtqSpd);
91
92         if (mejora.siguienteMejora)
93         {
94             siguienteMejora.text = mejora.siguienteMejora.name;
95         }
96         else
97         {
98             siguienteMejora.text = "Ninguna";
99         }
100
101         siguienteMejora.color = listaColorTextoTier[mejora.tierMejora];

```

```
102     siguienteMejoraTitulo.color = listaColorTextoTier[mejora.tierMejora];
103 }
104
105 public void actualizarStat(Image i, Image ib, float f)
106 {
107     i.enabled = true;
108     ib.enabled = true;
109
110     if (f > 1)
111     {
112         i.sprite = spriteSube;
113         ib.sprite = spriteSube;
114         i.color = Color.green;
115     }
116     else if (f < 1)
117     {
118         i.sprite = spriteBaja;
119         ib.sprite = spriteBaja;
120         i.color = Color.red;
121     }
122     else
123     {
124         i.enabled = false;
125         ib.enabled = false;
126     }
127 }
128
129 private void OnDisable()
130 {
131     if (!seleccionada && mejora.name != "Analisi sintactico")
132     {
133         ControlPartida.Instancia.controladorMejoras.poolMejoras.Add(mejora);
134     }
135 }
136
137 // Update is called once per frame
138 public void activarMejora()
139 {
140     seleccionada = true;
141
142     ControlPartida.Instancia.controladorMejoras.ActivarMejora(mejora);
143 }
144 }
```

```

1 using System;
2 using System.Collections;
3 using System.Collections.Generic;
4 using TMPro;
5 using UnityEngine;
6
7 public class ControladorPuntos : MonoBehaviour
8 {
9     void Awake()
10    {
11        if (ControlPartida.Instancia.controladorPuntos == null)
12        {
13            ControlPartida.Instancia.controladorPuntos = this;
14            DontDestroyOnLoad(gameObject);
15        }
16        else
17        {
18            Destroy(gameObject);
19        }
20    }
21
22    public void Autodestruccion()
23    {
24        Destroy(gameObject);
25    }
26
27    private void Start()
28    {
29        puntos = PlayerPrefs.GetInt("Puntos", 0);
30    }
31
32    public int puntos = 0;
33    public int puntosPartida = 0;
34    public int combo = 1;
35    public int ronda = 1;
36    public bool nohited = true;
37    public bool rondanohited = true;
38
39    public void manPegao()
40    {
41        if (ronda == ControlPartida.Instancia.controladorMejoras.cantidadBaseRondas)
42        {
43            rondanohited = false;
44        }
45
46        nohited = false;
47        combo--;
48        if (combo < 1)
49        {
50            combo = 1;
51        }
52    }
53
54    public void aumentarPuntos(int puntosBase)
55    {
56        puntosPartida += puntosBase * combo;
57        combo++;
58    }
59
60    public void reiniciarPunros()
61    {
62        rondanohited = true;
63        nohited = true;
64        combo = 1;
65        ronda = 1;
66        puntosPartida = 0;
67    }
68
69    private void OnApplicationQuit()
70    {
71        PlayerPrefs.SetInt("Puntos", puntos);
72    }
73 }

```

```
1 using System.Collections;
2 using System.Collections.Generic;
3 using UnityEngine;
4 using UnityEngine.Audio;
5 using UnityEngine.UI;
6
7 public class ControladorAjustes : MonoBehaviour
8 {
9     [SerializeField] private AudioMixer mixer;
10
11     [SerializeField] private Slider sliderVolumenMusica;
12     [SerializeField] private Slider sliderVolumenEfectos;
13
14     public float volumenEfectos;
15     public float volumenMusica;
16
17     // Start is called before the first frame update
18     void Start()
19     {
20         mixer.GetFloat("Musico", out volumenMusica);
21         mixer.GetFloat("Sonidos", out volumenEfectos);
22
23         sliderVolumenEfectos.value = PlayerPrefs.GetFloat("VolumenEfectos", 0.75f);
24         sliderVolumenMusica.value = PlayerPrefs.GetFloat("VolumenMusica", 0.75f);
25     }
26
27     // Update is called once per frame
28     void Update()
29     {
30         volumenEfectos = sliderVolumenEfectos.value;
31         volumenMusica = sliderVolumenMusica.value;
32
33         mixer.SetFloat("Musico", Mathf.Log10(volumenMusica) * 20);
34         mixer.SetFloat("Sonidos", Mathf.Log10(volumenEfectos) * 20);
35
36         PlayerPrefs.SetFloat("VolumenMusica", volumenMusica);
37         PlayerPrefs.SetFloat("VolumenEfectos", volumenEfectos);
38     }
39 }
```

```

1 using System.Collections;
2 using System.Collections.Generic;
3 using System.Linq;
4 using Photon.Pun;
5 using TMPro;
6 using UnityEngine;
7 using UnityEngine.AI;
8 using Object = System.Object;
9
10 public class ControladorJugador : MonoBehaviour
11 {
12     [SerializeField] public Transform sitioSombrero;
13     [SerializeField] public Transform centroArea;
14     [SerializeField] public Material matChasisNave;
15     [SerializeField] public GameObject modeloNave;
16
17     [SerializeField] private GameObject jugador;
18
19     [SerializeField] private TextMeshProUGUI textoCuantos;
20     [SerializeField] private GameObject cartelImpacto;
21     [SerializeField] private GameObject camara;
22     [SerializeField] ControladorVida controladorVida;
23     [SerializeField] ApuntarAlEnemigo apuntarAlEnemigo;
24     [SerializeField] NavMeshAgent agente;
25
26     public List<Mejora> listaMejorasObtenidas;
27
28     [SerializeField] private float duracionEfectoShake;
29     [SerializeField] private float intensidadEfectoShake;
30     private GameObject area;
31     private ControladorEnemigos controladorEnemigos;
32     private PhotonView photonView;
33     private GameObject sombrero;
34
35     // Start is called before the first frame update
36     void Start()
37     {
38         photonView = GetComponent<PhotonView>();
39
40
41         if (ControlPartida.Instancia.controladorSala)
42         {
43             if (photonView.IsMine)
44             {
45                 ControlPartida.Instancia.controladorMejoras.jugador = this;
46                 ControlPartida.Instancia.controladorMejoras.AplicarMejoras();
47             }
48         }
49         else
50         {
51             ControlPartida.Instancia.controladorMejoras.jugador = this;
52             ControlPartida.Instancia.controladorMejoras.AplicarMejoras();
53         }
54
55         controladorVida.cuandoRecibaImaoceto.AddListener(() =>
56         {
57             ControlPartida.Instancia.controladorPuntos.manPegao();
58             StartCoroutine(efectoImpactoGUI());
59             StartCoroutine(efectoImpactoCamara(duracionEfectoShake, intensidadEfectoShake));
60         });
61
62         controladorEnemigos = FindObjectOfType<ControladorEnemigos>();
63     }
64
65     void Update()
66     {
67         textoCuantos.text = controladorEnemigos.enemigos.Count + " Restantes";
68
69         centroArea.position = transform.position;
70     }
71
72     public void DesconectarJugador()
73     {
74         if (ControlPartida.Instancia.controladorSala)
75         {
76             photonView.RPC("borrarJugador", RpcTarget.AllBufferedViaServer);
77         }
78         else
79         {
80             borrarJugador();
81         }
82     }
83
84     public void ReiniciarPartida()
85     {
86         ControlPartida.Instancia.controladorMejoras.ReiniciarPoolMejoras();
87         ControlPartida.Instancia.cambiarEstado(ControlPartida.Instancia.partidaEstado);
88     }
89
90     [PunRPC]
91     public void borrarJugador()
92     {
93         Destroy(jugador);
94     }
95
96     IEnumerator efectoImpactoGUI()
97     {
98         cartelImpacto.SetActive(true);
99         yield return new WaitForSecondsRealtime(0.2f);
100        cartelImpacto.SetActive(false);
101    }

```

```

102
103     IEnumerator efectoImpactoCamara(float duracion, float intensidad)
104     {
105         Vector3 posicionOriginal = camara.transform.localPosition;
106
107         float transcurrido = 0;
108         while (transcurrido < duracion)
109         {
110             float x = Random.Range(-1f, 1f) * intensidad;
111             float y = Random.Range(-1f, 1f) * intensidad;
112
113             camara.transform.localPosition = new Vector3(x, y, 0);
114             transcurrido += Time.deltaTime;
115
116             yield return null;
117         }
118
119         camara.transform.localPosition = posicionOriginal;
120     }
121
122     public void activarMejoras(List<Mejora> mejoras)
123     {
124         string listaMejoras = "";
125         for (int i = 0; i < mejoras.Count; i++)
126         {
127             listaMejoras += (mejoras[i].name) + ",";
128         }
129
130
131         if (ControlPartida.Instancia.controladorSala)
132         {
133             photonView.RPC("AplicarMejoras", RpcTarget.AllBufferedViaServer, listaMejoras);
134         }
135         else
136         {
137             AplicarMejoras(listaMejoras);
138         }
139     }
140
141     [PunRPC]
142     public void AplicarMejoras(string mejorasArray)
143     {
144         // Filtrar las mejoras válidas y no repetidas del array
145         List<string> mejorasString = new List<string>(mejorasArray.Split(","));
146         List<Mejora> listaMejorasObtenidas = new List<Mejora>();
147         foreach (string s in mejorasString)
148         {
149             if (s.Length > 0)
150             {
151                 if (ControlPartida.Instancia.controladorMejoras.baseMejoras.ContainsKey(s))
152                 {
153                     listaMejorasObtenidas.Add(ControlPartida.Instancia.controladorMejoras.baseMejoras[s]);
154                 }
155                 else if (ControlPartida.Instancia.controladorMejoras.navesDisponibles.ContainsKey(s))
156                 {
157                     listaMejorasObtenidas.Add(ControlPartida.Instancia.controladorMejoras.navesDisponibles[s]);
158                 }
159                 else if (ControlPartida.Instancia.controladorMejoras.sombrerosDisponibles.ContainsKey(s))
160                 {
161                     listaMejorasObtenidas.Add(ControlPartida.Instancia.controladorMejoras.sombrerosDisponibles[s]);
162                 }
163             }
164         }
165
166
167         Mejora sumaMejoras = ScriptableObject.CreateInstance<Mejora>();
168
169         foreach (Mejora mejora in listaMejorasObtenidas)
170         {
171             if (listaMejorasObtenidas.Contains(mejora.siguienteMejora))
172             {
173                 continue;
174             }
175
176             switch (mejora.tipo)
177             {
178                 case TipoMejora.Bala:
179                     apuntarAlEnemigo.tiposBalas.Add(mejora.objeto);
180                     break;
181                 case TipoMejora.Chasis:
182                     Material[] materials = modeloNave.GetComponent<Renderer>().materials;
183                     materials[3] = new Material(mejora.objeto.GetComponent<Renderer>().sharedMaterial);
184                     modeloNave.GetComponent<Renderer>().materials = materials;
185
186                     break;
187                 case TipoMejora.Modelo:
188                     Mesh mesh = (mejora.objeto.GetComponent<MeshFilter>()).sharedMesh;
189                     modeloNave.GetComponent<MeshFilter>().mesh = mesh;
190                     continue;
191
192                     break;
193                 case TipoMejora.Sombreros:
194                     sombrero = mejora.objeto;
195                     Instantiate(sombrero,
196                             sitioSombrero.position,
197                             sitioSombrero.rotation,
198                             sitioSombrero);
199                     break;
200                 case TipoMejora.Área:
201                     area = mejora.objeto;
202                     Instantiate(area,

```

```

203         centroArea.position,
204         centroArea.rotation,
205         centroArea);
206     break;
207     case TipoMejora.Utilidad:
208         switch (mejora.name)
209         {
210             case "Balas guiadas":
211                 apuntarALEnemigo.balasGuiadas = true;
212                 break;
213             case "Balas explosivas":
214                 apuntarALEnemigo.balasExplosivas = true;
215                 break;
216         }
217     break;
218 }
219
220
221 sumaMejoras.modAtq *= mejora.modAtq;
222 sumaMejoras.modVida *= mejora.modVida;
223 sumaMejoras.modAtqSpd *= mejora.modAtqSpd;
224 sumaMejoras.modDispersion *= mejora.modDispersion;
225 sumaMejoras.modVel *= mejora.modVel;
226 sumaMejoras.modAcc *= mejora.modAcc;
227 sumaMejoras.modVelBala *= mejora.modVelBala;
228 sumaMejoras.modTamBala *= mejora.modTamBala;
229 sumaMejoras.modRango *= mejora.modRango;
230 sumaMejoras.modTiempoVidaBala *= mejora.modTiempoVidaBala;
231 sumaMejoras.balasAdicionales += mejora.balasAdicionales;
232 }
233
234
235 apuntarALEnemigo.daño = ControlPartida.Instancia.controladorMejoras.naveAciva.modAtq * sumaMejoras.modAtq;
236 apuntarALEnemigo.tasaDisparo =
237     ControlPartida.Instancia.controladorMejoras.naveAciva.modAtqSpd / sumaMejoras.modAtqSpd;
238 apuntarALEnemigo.velocidadBala =
239     ControlPartida.Instancia.controladorMejoras.naveAciva.modVelBala * sumaMejoras.modVelBala;
240 apuntarALEnemigo.tamañoBala =
241     ControlPartida.Instancia.controladorMejoras.naveAciva.modTamBala * sumaMejoras.modTamBala;
242 apuntarALEnemigo.rangoDisparo =
243     ControlPartida.Instancia.controladorMejoras.naveAciva.modRango * sumaMejoras.modRango;
244 apuntarALEnemigo.tiempoVidaBala = ControlPartida.Instancia.controladorMejoras.naveAciva.modTiempoVidaBala *
245     sumaMejoras.modTiempoVidaBala;
246 apuntarALEnemigo.dispersion = ControlPartida.Instancia.controladorMejoras.naveAciva.modDispersion *
247     sumaMejoras.modDispersion;
248 apuntarALEnemigo.balasADisparar = ControlPartida.Instancia.controladorMejoras.naveAciva.balasAdicionales +
249     sumaMejoras.balasAdicionales;
250
251
252 controladorVida.vidaMaxima =
253     ControlPartida.Instancia.controladorMejoras.naveAciva.modVida * sumaMejoras.modVida;
254
255 controladorVida.vida = controladorVida.vidaMaxima;
256
257
258 agente.speed = ControlPartida.Instancia.controladorMejoras.naveAciva.modVel * sumaMejoras.modVel;
259 agente.acceleration = ControlPartida.Instancia.controladorMejoras.naveAciva.modAcc * sumaMejoras.modAcc;
260
261 camara.GetComponent<Camera>().orthographicSize = 5 + agente.speed / 10;
262 }

```

```

1 using System;
2 using System.Collections;
3 using System.Collections.Generic;
4 using System.Globalization;
5 using System.IO;
6 using System.Linq;
7 using System.Threading;
8 using Photon.Pun;
9 using Unity.VisualScripting;
10 using UnityEngine;
11 using UnityEngine.AI;
12 using UnityEngine.Rendering;
13 using UnityEngine.SceneManagement;
14 using UnityEngine.Serialization;
15
16 public class ControladorMejoras : MonoBehaviour
17 {
18     public ControladorJugador jugador;
19
20     public Dictionary<string, Mejora> navesDisponibles = new Dictionary<string, Mejora>();
21     public List<Mejora> navesObtenidas;
22     public Mejora naveActiva;
23
24     public Dictionary<string, Mejora> sombrerosDisponibles = new Dictionary<string, Mejora>();
25     public List<Mejora> sombrerosObtenidas;
26     public Mejora sombreroActivo;
27
28     public Dictionary<string, Mejora> baseMejoras = new Dictionary<string, Mejora>();
29     public List<Mejora> poolMejoras;
30     public List<Mejora> mejorasObtenidas;
31
32     public int cantidadBaseRondas = 5;
33
34     string pathMejoras = "Mejoras/Mejoras";
35     string pathIconos = "Mejoras/Iconos";
36     string pathObjetos = "Mejoras/Objetos";
37
38     void Awake()
39     {
40         if (ControlPartida.Instancia.controladorMejoras == null)
41         {
42             ControlPartida.Instancia.controladorMejoras = this;
43             DontDestroyOnLoad(gameObject);
44         }
45         else
46         {
47             Destroy(gameObject);
48         }
49     }
50
51     // Start is called before the first frame update
52     void Start()
53     {
54         PlayerPrefs.SetInt("Ruta base", 1);
55
56         if (Convert.ToBoolean(PlayerPrefs.GetInt("Ruta base plus", 0))) cantidadBaseRondas += 5;
57
58         CargarMejoras(pathMejoras);
59         CargarNaves();
60         CargarSombreros();
61         ControlPartida.Instancia.cambiarEstado(ControlPartida.Instancia.menuEstado);
62     }
63
64     void CargarMejoras(string pathArchivo)
65     {
66         Dictionary<string, Sprite> iconos = Resources.LoadAll(pathIconos, typeof(Sprite))
67             .Cast<Sprite>()
68             .ToDictionary(m => m.name);
69
70         Dictionary<string, GameObject> objetos = Resources.LoadAll(pathObjetos, typeof(GameObject))
71             .Cast<GameObject>()
72             .ToDictionary(m => m.name);
73
74         TextAsset file = Resources.Load(pathMejoras) as TextAsset;
75         string separador = ",";
76         string linea;
77         string[] lineas = file.ToString().Split(
78             new string[] { Environment.NewLine },
79             StringSplitOptions.None
80         );
81
82         // Leer la primera línea pero descartarla porque es el encabezado
83         for (int i = 1; i < lineas.Length; i++)
84         {
85             Mejora mejora = new Mejora();
86             linea = lineas[i];
87
88             string[] fila = linea.Split(separador);
89
90             mejora.name = fila[0];
91             if (objetos.TryGetValue(fila[0], out GameObject objeto))
92             {
93                 mejora.objeto = objeto;
94             }
95
96             if (iconos.TryGetValue(fila[0], out Sprite icono))
97             {
98                 mejora.icono = icono;
99             }
100
101             mejora.descripcion = fila[1];

```

```

102     mejora.tipo = Enum.Parse<TipoMejora>(fila[2]);
103     if (baseMejoras.TryGetValue(fila[3], out Mejora siguienteMejora))
104     {
105         mejora.siguienteMejora = siguienteMejora;
106     }
107
108     mejora.rutaNecesaria = fila[4];
109     mejora.modAtq = float.Parse(fila[5], CultureInfo.InvariantCulture);
110     mejora.modVida = float.Parse(fila[6], CultureInfo.InvariantCulture);
111     mejora.modAtqSpd = float.Parse(fila[7], CultureInfo.InvariantCulture);
112     mejora.modDispersion = float.Parse(fila[8], CultureInfo.InvariantCulture);
113     mejora.modVel = float.Parse(fila[9], CultureInfo.InvariantCulture);
114     mejora.modAcc = float.Parse(fila[10], CultureInfo.InvariantCulture);
115     mejora.modRango = float.Parse(fila[11], CultureInfo.InvariantCulture);
116     mejora.modTiempoVidaBala = float.Parse(fila[12], CultureInfo.InvariantCulture);
117     mejora.modVelBala = float.Parse(fila[13], CultureInfo.InvariantCulture);
118     mejora.modTamBala = float.Parse(fila[14], CultureInfo.InvariantCulture);
119     mejora.balasAdicionales = int.Parse(fila[15]);
120     mejora.tierMejora = int.Parse(fila[16]);
121
122     if (mejora.tipo == TipoMejora.Sombreros)
123     {
124         sombrerosDisponibles.Add(mejora.name, mejora);
125         continue;
126     }
127
128     if (mejora.tipo == TipoMejora.Modelo)
129     {
130         navesDisponibles.Add(mejora.name, mejora);
131         continue;
132     }
133
134     baseMejoras.Add(mejora.name, mejora);
135 }
136
137
138 public void ReiniciarPoolMejoras()
139 {
140     cantidadBaseRondas = 5;
141     if (ControlPartida.Instancia.controladorSala)
142     {
143         if (ControlPartida.Instancia.controladorSala.rutasCompartidas.Contains("Ruta base plus"))
144             cantidadBaseRondas += 5;
145     }
146     else
147     {
148         if (Convert.ToBoolean(PlayerPrefs.GetInt("Ruta base plus", 0))) cantidadBaseRondas += 5;
149     }
150
151     poolMejoras.Clear();
152     mejorasObtenidas.Clear();
153
154     foreach (KeyValuePair<string, Mejora> valuePair in baseMejoras)
155     {
156         if (ControlPartida.Instancia.controladorSala)
157         {
158             if (ControlPartida.Instancia.controladorSala.rutasCompartidas
159                 .Contains(valuePair.Value.rutaNecesaria) &&
160                 valuePair.Value.tierMejora == 0)
161             {
162                 poolMejoras.Add(valuePair.Value);
163             }
164         }
165         else
166         {
167             if (Convert.ToBoolean(PlayerPrefs.GetInt(valuePair.Value.rutaNecesaria, 0)) &&
168                 valuePair.Value.tierMejora == 0)
169             {
170                 poolMejoras.Add(valuePair.Value);
171             }
172         }
173     }
174
175     mejorasObtenidas.Add(naveActiva);
176     if (sombreroActivo)
177     {
178         mejorasObtenidas.Add(sombreroActivo);
179     }
180 }
181
182
183
184 public void CargarNaves()
185 {
186     string textoNavesObtenidas = PlayerPrefs.GetString("Naves obtenidas", "Nave base,");
187     string[] navesObtenidas = textoNavesObtenidas.Split(",");
188     this.navesObtenidas.Clear();
189     for (int i = 0; i < navesObtenidas.Length; i++)
190     {
191         if (navesObtenidas[i].Length < 1)
192         {
193             continue;
194         }
195
196         if (Convert.ToBoolean(PlayerPrefs.GetInt(navesDisponibles[navesObtenidas[i]].rutaNecesaria, 0)))
197         {
198             this.navesObtenidas.Add(navesDisponibles[navesObtenidas[i]]);
199         }
200     }
201
202     naveActiva = navesDisponibles[PlayerPrefs.GetString("Nave activa", "Nave base")];

```

```

203     }
204
205     public void CargarSombreros()
206     {
207         string textoSombrerosObtenidos = PlayerPrefs.GetString("Sombreros obtenidos", "");
208         string[] sombrerosObtenidos = textoSombrerosObtenidos.Split(",");
209
210         for (int i = 0; i < sombrerosObtenidos.Length; i++)
211         {
212             if (sombrosObtenidos[i].Length < 1)
213             {
214                 continue;
215             }
216
217             if (Convert.ToBoolean(PlayerPrefs.GetInt(sombrosDisponibles[sombrosObtenidos[i]].rutaNecesaria, 0)))
218             {
219                 this.sombrosObtenidos.Add(sombrosDisponibles[sombrosObtenidos[i]]);
220             }
221         }
222
223         if (PlayerPrefs.GetString("Sombrero activo", "").Length > 0)
224         {
225             sombreroActivo = sombrerosDisponibles[PlayerPrefs.GetString("Sombrero activo", "")];
226         }
227     }
228
229     public void AplicarMejoras()
230     {
231         jugador.activarMejoras(mejorasObtenidas);
232     }
233
234     public void ActivarMejora(Mejora mejora)
235     {
236         if (mejora.siguieteMejora)
237         {
238             if (Convert.ToBoolean(PlayerPrefs.GetInt(mejora.siguieteMejora.rutaNecesaria, 0)))
239             {
240                 poolMejoras.Add(mejora.siguieteMejora);
241             }
242         }
243
244         mejorasObtenidas.Add(mejora);
245
246         if (ControlPartida.Instancia.controladorPuntos.ronda >= cantidadBaseRondas)
247         {
248             if (ControlPartida.Instancia.controladorSala)
249             {
250                 ControlPartida.Instancia.controladorSala.acabarPartida();
251             }
252             else
253             {
254                 SceneManager.LoadScene("EndGame");
255             }
256         }
257         else
258         {
259             ControlPartida.Instancia.controladorPuntos.ronda++;
260             if (ControlPartida.Instancia.controladorSala)
261             {
262                 ControlPartida.Instancia.controladorPausa.modocarga(true);
263                 ControlPartida.Instancia.controladorSala.usuarioLocalListo();
264             }
265             else
266             {
267                 ControlPartida.Instancia.cambiarEstado(ControlPartida.Instancia.partidaEstado);
268             }
269         }
270     }
271
272     public void ActivarMejoraInGame(Mejora mejora)
273     {
274         if (mejora.siguieteMejora)
275         {
276             poolMejoras.Add(mejora.siguieteMejora);
277         }
278
279         mejorasObtenidas.Add(mejora);
280         poolMejoras.Remove(mejora);
281         AplicarMejoras();
282     }
283 }

```

```

1 using System;
2 using System.Collections;
3 using System.Collections.Generic;
4 using System.Linq;
5 using TMPro;
6 using UnityEngine;
7 using Random = UnityEngine.Random;
8
9 public class ControladorEnemigos : MonoBehaviour
10 {
11     public List<ControladorVida> enemigos;
12     [SerializeField] private TextMeshProUGUI textoCuantos;
13     [SerializeField] private GameObject[] bichosSpawn;
14     [SerializeField] private Transform[] sitiosSpawn;
15     [SerializeField] private int cunatosSpawnear;
16
17     private bool empezoRonda = false;
18
19     string pathEnemigos = "Enemigos/Tier ";
20     private List<GameObject> objetos = new List<GameObject>();
21     private List<List<GameObject>> enemigosObjetos = new List<List<GameObject>>();
22
23     Aleatoriedad random;
24
25     // Start is called before the first frame update
26     private void Awake()
27     {
28         ControlPartida.Instancia.controladorEnemigos = this;
29     }
30
31     void Start()
32     {
33         for (int i = 1; i < 10; i++)
34         {
35             objetos = Resources.LoadAll(pathEnemigos + i, typeof(GameObject))
36                 .Cast<GameObject>()
37                 .ToList();
38             enemigosObjetos.Add(objetos);
39         }
40
41         if (ControlPartida.Instancia.controladorSala)
42         {
43             random = new Aleatoriedad(ControlPartida.Instancia.controladorSala.semilla);
44         }
45         else
46         {
47             random = new Aleatoriedad(Random.Range(1000, 9999));
48         }
49     }
50
51     // Update is called once per frame
52     void Update()
53     {
54         if (empezoRonda)
55         {
56             if (enemigos.Count == 0)
57             {
58                 foreach (ControladorPausa pausa in FindObjectsOfType<ControladorPausa>())
59                 {
60                     if (!pausa.finRonda)
61                     {
62                         pausa.finDeRonda();
63                     }
64                 }
65
66                 foreach (ControladorBala bala in FindObjectsOfType<ControladorBala>())
67                 {
68                     Destroy(bala.gameObject);
69                 }
70             }
71         }
72     }
73
74     public void iniciar()
75     {
76         ControladorVida[] vidas = FindObjectsOfType<ControladorVida>();
77         for (int i = 0; i < vidas.Length; i++)
78         {
79             if (vidas[i].CompareTag("Enemigo"))
80             {
81                 enemigos.Add(vidas[i]);
82             }
83         }
84
85         empezoRonda = true;
86     }
87
88     public GameObject devolverEnemigo(int tier)
89     {
90         int aux = random.nuevoEntero(0, enemigosObjetos[tier - 1].Count);
91         return enemigosObjetos[tier - 1][aux];
92     }
93
94     public GameObject devolverEnemigo(int tier, string nombre)
95     {
96         foreach (GameObject o in enemigosObjetos[tier])
97         {
98             if (o.name == nombre)
99             {
100                 return o;
101             }
102         }
103     }

```

```
102         }
103     }
104
105     return null;
106 }
107 }
```

```
1 using System;
2 using System.Collections;
3 using System.Collections.Generic;
4 using ExitGames.Client.Photon.StructWrapping;
5 using UnityEngine;
6 using Random = UnityEngine.Random;
7
8
9 public class ControladorConsumible : MonoBehaviour
10 {
11     public List<GameObject> consumibles;
12     System.Random random;
13
14
15     void Awake()
16     {
17         if (ControlPartida.Instancia.controladorConsumible == null)
18         {
19             ControlPartida.Instancia.controladorConsumible = this;
20             DontDestroyOnLoad(gameObject);
21         }
22         else
23         {
24             Destroy(gameObject);
25         }
26     }
27
28     void Start()
29     {
30         if (ControlPartida.Instancia.controladorSala)
31         {
32             random = new System.Random(ControlPartida.Instancia.controladorSala.semilla);
33         }
34         else
35         {
36             random = new System.Random();
37         }
38     }
39
40     public GameObject devolverConsumible()
41     {
42         if (ControlPartida.Instancia.controladorMejoras.poolMejoras.Count > 0)
43         {
44             return consumibles[random.Next(0, consumibles.Count)];
45         }
46         return consumibles[0];
47     }
48 }
49 }
```

```
1 using System.Collections;
2 using System.Collections.Generic;
3 using UnityEngine;
4
5 public class CambiadorEstadoPartida : MonoBehaviour
6 {
7     public enum Estado
8     {
9         Inicio,
10        Menu,
11        Lobby,
12        Partida
13    }
14
15    public Estado estadoPartida;
16    // Start is called before the first frame update
17
18    public void CambiarEstado()
19    {
20        switch (estadoPartida)
21        {
22            case Estado.Inicio:
23                ControlPartida.Instancia.cambiarEstado(ControlPartida.Instancia.inicioEstado);
24                break;
25            case Estado.Menu:
26                ControlPartida.Instancia.cambiarEstado(ControlPartida.Instancia.menuEstado);
27                break;
28            case Estado.Lobby:
29                ControlPartida.Instancia.cambiarEstado(ControlPartida.Instancia.lobbyEstado);
30                break;
31            case Estado.Partida:
32                ControlPartida.Instancia.cambiarEstado(ControlPartida.Instancia.partidaEstado);
33                break;
34        }
35    }
36 }
```

```

1 using System;
2 using System.Collections;
3 using System.Collections.Generic;
4 using TMPro;
5 using UnityEngine;
6
7 public class ControladorDisplayPuntos : MonoBehaviour
8 {
9     [SerializeField] private TextMeshProUGUI textoPuntos;
10    [SerializeField] private TextMeshProUGUI textoCombo;
11    [SerializeField] private TextMeshProUGUI textoRonda;
12    private int belosidad = 1;
13    private int prevCombo;
14
15    private Vector3 baseEscala;
16
17    private ControladorPuntos controladorPuntos;
18
19    // Start is called before the first frame update
20    void Start()
21    {
22        controladorPuntos = ControlPartida.Instancia.controladorPuntos;
23
24        baseEscala = transform.localScale;
25
26
27        if (ControlPartida.Instancia.estadoActualPartida == ControlPartida.Instancia.menuEstado)
28        {
29            textoPuntos.text = controladorPuntos.puntos + "";
30        }
31        else
32        {
33            textoPuntos.text = controladorPuntos.puntosPartida + "";
34        }
35
36        textoCombo.text = controladorPuntos.combo + "x";
37    }
38
39    // Update is called once per frame
40    void Update()
41    {
42        belosidad = 10 * controladorPuntos.combo;
43        if (ControlPartida.Instancia.estadoActualPartida == ControlPartida.Instancia.menuEstado)
44        {
45            int puntos = int.Parse(textoPuntos.text);
46
47            if (puntos - belosidad * 1000 > controladorPuntos.puntos)
48            {
49                puntos -= belosidad * 1000;
50            }
51            else
52            {
53                puntos = controladorPuntos.puntos;
54            }
55
56            textoPuntos.text = puntos + "";
57        }
58        else
59        {
60            int puntos = int.Parse(textoPuntos.text);
61
62            if (puntos + belosidad < controladorPuntos.puntosPartida)
63            {
64                puntos += belosidad;
65            }
66            else
67            {
68                puntos = controladorPuntos.puntosPartida;
69            }
70
71            textoPuntos.text = puntos + "";
72            textoRonda.text = "Ronda : " + controladorPuntos.ronda;
73        }
74
75        if (prevCombo != controladorPuntos.combo)
76        {
77            StartCoroutine(efctoCombo());
78        }
79
80        textoCombo.text = controladorPuntos.combo + "x";
81        prevCombo = controladorPuntos.combo;
82    }
83
84    IEnumerator efctoCombo()
85    {
86        float tiempoRutina = 0f;
87        float duracionAnimacion = 0.1f;
88
89        while (tiempoRutina < duracionAnimacion)
90        {
91            tiempoRutina += Time.unscaledDeltaTime;
92            textoCombo.transform.localScale = new Vector3(
93                Mathf.Lerp(baseEscala.x, baseEscala.x * 1.5f, tiempoRutina / duracionAnimacion),
94                Mathf.Lerp(baseEscala.y, baseEscala.y * 1.5f, tiempoRutina / duracionAnimacion),
95                Mathf.Lerp(baseEscala.z, baseEscala.z * 1.5f, tiempoRutina / duracionAnimacion)
96            );
97            yield return null;
98        }
99
100        textoCombo.transform.localScale = baseEscala;
101

```

```
102
103
104     yield return null;
105 }
106 }
```

```

1 using System;
2 using System.Collections;
3 using System.Collections.Generic;
4 using System.Linq;
5 using MoreMountains.Feedbacks;
6 using TMPro;
7 using UnityEngine;
8 using UnityEngine.UI;
9
10 public class ControladorDisplayTienda : MonoBehaviour
11 {
12     [SerializeField] private Button botonComprar;
13     [SerializeField] private Button botonUsar;
14
15     [SerializeField] private Image imagenNave;
16
17     [SerializeField] private Image vidaStat;
18     private float llenadoVidaPrevio;
19     [SerializeField] private Image dañoStat;
20     private float llenadoDañoPrevio;
21     [SerializeField] private Image velocidadStat;
22     private float llenadoVelocidadPrevio;
23     [SerializeField] private Image RangoStat;
24     private float llenadoRangoPrevio;
25     [SerializeField] private Image velocidadAtqStat;
26     private float llenadoVelocidadAtqPrevio;
27
28     private float tiempo;
29
30     [SerializeField] private TextMeshProUGUI nombreNave;
31     [SerializeField] private TextMeshProUGUI precioNave;
32
33     [SerializeField] private MMF_Player nombreNaveFeedback;
34     MMF_TMPTextReveal nombreNaveFeedbackTMP;
35
36     int precio;
37     int precioFinal;
38
39     private int mejoraMostradaActualmente = 0;
40
41     // Start is called before the first frame update
42     void Start()
43     {
44         for (int i = 0; i < ControlPartida.Instancia.controladorMejoras.navesDisponibles.Count; i++)
45         {
46             if (ControlPartida.Instancia.controladorMejoras.navesDisponibles.ElementAt(i).Value ==
47                 ControlPartida.Instancia.controladorMejoras.naveAciva)
48             {
49                 mejoraMostradaActualmente = i;
50                 break;
51             }
52         }
53
54         nombreNaveFeedbackTMP = nombreNaveFeedback.GetFeedbackOfType<MMF_TMPTextReveal>("Revelar el título de la nave");
55
56         actualizarDisplay(
57             ControlPartida.Instancia.controladorMejoras.navesDisponibles[
58                 ControlPartida.Instancia.controladorMejoras.naveAciva.name]);
59     }
60
61     // Update is called once per frame
62     void Update()
63     {
64         tiempo += Time.deltaTime;
65
66         if (precio != precioFinal)
67         {
68             precio = (int)Mathf.Lerp(precio, precioFinal, tiempo);
69             precioNave.text = precio + "pts";
70         }
71     }
72
73     public void actualizarDisplay(Mejora mejora)
74     {
75         imagenNave.sprite = mejora.icono;
76         nombreNaveFeedbackTMP.NewText = mejora.name;
77         nombreNaveFeedback.PlayFeedbacks();
78         // nombreNave.text = mejora.name;
79         botonComprar.gameObject.SetActive(true);
80         precioNave.gameObject.SetActive(true);
81         botonUsar.gameObject.SetActive(false);
82         precioFinal = (ControlPartida.Instancia
83             .controladorMejoras
84             .navesDisponibles.ElementAt(mejoraMostradaActualmente).Value.tierMejora + 1) * 200000;
85
86
87         llenadoVidaPrevio = vidaStat.fillAmount;
88         llenadoDañoPrevio = dañoStat.fillAmount;
89         llenadoVelocidadPrevio = velocidadStat.fillAmount;
90         llenadoRangoPrevio = RangoStat.fillAmount;
91         llenadoVelocidadAtqPrevio = velocidadAtqStat.fillAmount;
92
93         // actualizar stats de forma fluida en vez de de golpe
94         StartCoroutine(rellenar(mejora));
95
96         botonComprar.interactable = ControlPartida.Instancia.controladorPuntos.puntos >= precioFinal;
97         botonComprar.gameObject.GetComponent<ControladorHover>().enabled =
98             ControlPartida.Instancia.controladorPuntos.puntos >= precioFinal;
99
100         if (!ControlPartida.Instancia.controladorMejoras.navesObtenidas.Contains(mejora)) return;
101         botonComprar.gameObject.SetActive(false);

```

```

102     precioNave.gameObject.SetActive(false);
103     botonUsar.gameObject.SetActive(true);
104
105     if (!ControlPartida.Instancia.controladorMejoras.naveAciva) return;
106     if (ControlPartida.Instancia.controladorMejoras.naveAciva.Equals(mejora))
107     {
108         botonUsar.gameObject.SetActive(false);
109     }
110 }
111
112 IEnumerator rellenar(Mejora mejora)
113 {
114     tiempo = 0;
115     while (tiempo < 0.1f)
116     {
117         vidaStat.fillAmount = Mathf.Lerp(llenadoVidaPrevio, (mejora.modVida) / 150, tiempo / 0.1f);
118         dañoStat.fillAmount = Mathf.Lerp(llenadoDañoPrevio, (mejora.modAtq) / 15, tiempo / 0.1f);
119         velocidadStat.fillAmount = Mathf.Lerp(llenadoVelocidadPrevio, (mejora.modVel) / 15, tiempo / 0.1f);
120         RangoStat.fillAmount = Mathf.Lerp(llenadoRangoPrevio, (mejora.modRango) / 30, tiempo / 0.1f);
121         velocidadAtqStat.fillAmount =
122             Mathf.Lerp(llenadoVelocidadAtqPrevio, 1 - (mejora.modAtqSpd) / 0.6f, tiempo / 0.1f);
123         yield return null;
124     }
125 }
126
127 public void siguienteMejora()
128 {
129     mejoraMostradaActualmente++;
130     if (mejoraMostradaActualmente >= ControlPartida.Instancia.controladorMejoras.navesDisponibles.Count)
131     {
132         mejoraMostradaActualmente = 0;
133     }
134
135     while (!Convert.ToBoolean(PlayerPrefs.GetInt(ControlPartida.Instancia.controladorMejoras.navesDisponibles
136         .ElementAt(mejoraMostradaActualmente).Value.rutaNecesaria, 0)))
137     {
138         mejoraMostradaActualmente++;
139         if (mejoraMostradaActualmente >= ControlPartida.Instancia.controladorMejoras.navesDisponibles.Count)
140         {
141             mejoraMostradaActualmente = 0;
142         }
143     }
144
145     actualizarDisplay(ControlPartida.Instancia.controladorMejoras.navesDisponibles
146         .ElementAt(mejoraMostradaActualmente).Value);
147 }
148
149 public void mejoraAnterior()
150 {
151     mejoraMostradaActualmente--;
152     if (mejoraMostradaActualmente < 0)
153     {
154         mejoraMostradaActualmente = ControlPartida.Instancia.controladorMejoras.navesDisponibles.Count - 1;
155     }
156
157     while (!Convert.ToBoolean(PlayerPrefs.GetInt(ControlPartida.Instancia.controladorMejoras.navesDisponibles
158         .ElementAt(mejoraMostradaActualmente).Value.rutaNecesaria, 0)))
159     {
160         mejoraMostradaActualmente--;
161         if (mejoraMostradaActualmente < 0)
162         {
163             mejoraMostradaActualmente = ControlPartida.Instancia.controladorMejoras.navesDisponibles.Count - 1;
164         }
165     }
166
167     actualizarDisplay(ControlPartida.Instancia.controladorMejoras.navesDisponibles
168         .ElementAt(mejoraMostradaActualmente).Value);
169 }
170
171 public void comprarMejora()
172 {
173     ControlPartida.Instancia.controladorPuntos.puntos -= precioFinal;
174
175     ControlPartida.Instancia.controladorMejoras.navesObtenidas.Add(
176         ControlPartida.Instancia.controladorMejoras.navesDisponibles.ElementAt(mejoraMostradaActualmente).Value);
177
178     string navesObtenidas = "";
179     foreach (Mejora nave in ControlPartida.Instancia.controladorMejoras.navesObtenidas)
180     {
181         navesObtenidas += nave.name + ",";
182     }
183
184     PlayerPrefs.SetString("Naves obtenidas", navesObtenidas);
185
186     activarMejora();
187 }
188
189 public void activarMejora()
190 {
191     ControlPartida.Instancia.controladorMejoras.naveAciva =
192         ControlPartida.Instancia.controladorMejoras.navesDisponibles.ElementAt(mejoraMostradaActualmente).Value;
193
194     actualizarDisplay(ControlPartida.Instancia.controladorMejoras.navesDisponibles
195         .ElementAt(mejoraMostradaActualmente).Value);
196
197     PlayerPrefs.SetString("Nave activa", ControlPartida.Instancia.controladorMejoras.naveAciva.name);
198     ControlPartida.Instancia.controladorMejoras.ReiniciarPoolMejoras();
199 }
200 }
201 }

```

```
1 using System.Collections;
2 using System.Collections.Generic;
3 using UnityEngine;
4 using UnityEngine.UI;
5
6 public class ControladorDisplayVidaJugador : MonoBehaviour
7 {
8     [SerializeField] private Image imageVida;
9     [SerializeField] private Image imageVidaFondo;
10    [SerializeField] private float belosidad;
11
12    [SerializeField] private ControladorVida _controladorVida;
13
14    // Start is called before the first frame update
15    void Start()
16    {
17        imageVida.fillAmount = _controladorVida.vida / _controladorVida.vidaMaxima;
18    }
19
20    // Update is called once per frame
21    void Update()
22    {
23        imageVida.fillAmount = _controladorVida.vida / _controladorVida.vidaMaxima;
24
25        if (imageVidaFondo.fillAmount - belosidad * Time.deltaTime >
26            _controladorVida.vida / _controladorVida.vidaMaxima)
27        {
28            imageVidaFondo.fillAmount -= belosidad * Time.deltaTime;
29        }
30        else if (imageVidaFondo.fillAmount + belosidad * Time.deltaTime <
31            _controladorVida.vida / _controladorVida.vidaMaxima)
32        {
33            imageVidaFondo.fillAmount += belosidad * Time.deltaTime;
34        }
35        else
36        {
37            imageVidaFondo.fillAmount = imageVida.fillAmount;
38        }
39    }
40 }
```

```

1 using System.Collections;
2 using System.Collections.Generic;
3 using System.Linq;
4 using UnityEngine;
5 using UnityEngine.Audio;
6
7 public class ControladorReproductorMusical : MonoBehaviour
8 {
9     [SerializeField] private AudioMixer mixer;
10    [SerializeField] private AudioSource ReproductorMusica;
11    [SerializeField] private AudioListener EscuchadorGlobal;
12    public Camera camaraGlobal;
13
14    [SerializeField] private AudioClip[] listaMusicaMenu;
15    [SerializeField] private AudioClip[] listaMusicaPartida;
16    [SerializeField] private AudioClip[] listaMusica;
17    private int cancionActual = 0;
18
19    // Start is called before the first frame update
20    void Start()
21    {
22        DontDestroyOnLoad(this);
23        ControlPartida.Instancia.controladorReproductorMusical = this;
24
25        mixer.SetFloat("Sonidos", Mathf.Log10(PlayerPrefs.GetFloat("VolumenEfectos", 0.75f)) * 20);
26        mixer.SetFloat("Musico", Mathf.Log10(PlayerPrefs.GetFloat("VolumenMusica", 0.75f)) * 20);
27    }
28
29    // Update is called once per frame
30    void Update()
31    {
32        if (!ReproductorMusica.isPlaying)
33        {
34            SiguienteCancion();
35        }
36    }
37
38    public void SiguienteCancion()
39    {
40        cancionActual++;
41        if (cancionActual >= listaMusica.Length)
42        {
43            cancionActual = 0;
44        }
45
46        ReproductorMusica.clip = listaMusica[cancionActual];
47        ReproductorMusica.Play();
48    }
49
50    public void ActualizarEscuchadorGlobal(bool encendido)
51    {
52        EscuchadorGlobal.enabled = encendido;
53        camaraGlobal.enabled = encendido;
54    }
55
56    public void IniciarModoPartida()
57    {
58        if (!listaMusica.Contains(listaMusicaPartida[0]))
59        {
60            cancionActual = listaMusicaPartida.Length + 2;
61            listaMusica = listaMusicaPartida;
62            SiguienteCancion();
63        }
64
65        ActualizarEscuchadorGlobal(false);
66    }
67
68
69    public void IniciarModoMenu()
70    {
71        cancionActual = listaMusicaMenu.Length + 2;
72        listaMusica = listaMusicaMenu;
73        SiguienteCancion();
74        ActualizarEscuchadorGlobal(true);
75    }
76 }

```

```

1 using System;
2 using System.Collections;
3 using System.Collections.Generic;
4 using System.Linq;
5 using TMPro;
6 using UnityEngine;
7 using UnityEngine.UI;
8
9 public class ControladorDisplayTiendaSombreros : MonoBehaviour
10 {
11     [SerializeField] private Button botonComprar;
12     [SerializeField] private Button botonUsar;
13
14     [SerializeField] private Image imagenSombrero;
15     [SerializeField] private TextMeshProUGUI nombreSombrero;
16     [SerializeField] private TextMeshProUGUI precioSombrero;
17     int precio;
18     int precioFinal;
19
20     private float tiempo;
21
22     private int mejoraMostradaActualmente = 0;
23
24     // Start is called before the first frame update
25     void Start()
26     {
27         if (ControlPartida.Instancia.controladorMejoras.sombreroActivo)
28         {
29             for (int i = 0; i < ControlPartida.Instancia.controladorMejoras.navesDisponibles.Count; i++)
30             {
31                 if (ControlPartida.Instancia.controladorMejoras.sombrerosDisponibles.ElementAt(i).Value ==
32                     ControlPartida.Instancia.controladorMejoras.sombreroActivo)
33                 {
34                     mejoraMostradaActualmente = i;
35                     break;
36                 }
37             }
38
39             actualizarDisplay(
40                 ControlPartida.Instancia.controladorMejoras.sombrerosDisponibles[
41                     ControlPartida.Instancia.controladorMejoras.sombreroActivo.name]);
42         }
43         else
44         {
45             actualizarDisplay(ControlPartida.Instancia.controladorMejoras.sombrerosDisponibles
46                             .ElementAt(mejoraMostradaActualmente).Value);
47         }
48     }
49
50     // Update is called once per frame
51     void Update()
52     {
53         tiempo += Time.deltaTime;
54
55         if (precio != precioFinal)
56         {
57             precio = (int)Mathf.Lerp(precio, precioFinal, tiempo);
58             precioSombrero.text = precio + "pts";
59         }
60     }
61
62     public void actualizarDisplay(Mejora mejora)
63     {
64         tiempo = 0;
65         imagenSombrero.sprite = mejora.icono;
66         nombreSombrero.text = mejora.name;
67         botonComprar.gameObject.SetActive(true);
68         botonUsar.gameObject.SetActive(false);
69         precioFinal = (ControlPartida.Instancia
70             .controladorMejoras
71             .sombrerosDisponibles.ElementAt(mejoraMostradaActualmente).Value.tierMejora + 1) * 100000;
72
73         botonComprar.interactable = ControlPartida.Instancia.controladorPuntos.puntos >= precioFinal;
74         botonComprar.gameObject.GetComponent<ControladorHover>().enabled =
75             ControlPartida.Instancia.controladorPuntos.puntos >= precioFinal;
76
77         if (!ControlPartida.Instancia.controladorMejoras.sombrerosObtenidas.Contains(mejora)) return;
78         botonComprar.gameObject.SetActive(false);
79         precioSombrero.gameObject.SetActive(false);
80         botonUsar.gameObject.SetActive(true);
81
82         if (!ControlPartida.Instancia.controladorMejoras.sombreroActivo) return;
83         if (ControlPartida.Instancia.controladorMejoras.sombreroActivo.Equals(mejora))
84         {
85             botonUsar.gameObject.SetActive(false);
86         }
87     }
88
89     public void siguienteMejora()
90     {
91         mejoraMostradaActualmente++;
92         if (mejoraMostradaActualmente >= ControlPartida.Instancia.controladorMejoras.sombrerosDisponibles.Count)
93         {
94             mejoraMostradaActualmente = 0;
95         }
96
97         while (!Convert.ToBoolean(PlayerPrefs.GetInt(ControlPartida.Instancia.controladorMejoras.sombrerosDisponibles
98             .ElementAt(mejoraMostradaActualmente).Value.rutaNecesaria, 0)))
99         {
100             mejoraMostradaActualmente++;
101             if (mejoraMostradaActualmente >= ControlPartida.Instancia.controladorMejoras.sombrerosDisponibles.Count)

```

```

102     {
103         mejoraMostradaActualmente = 0;
104     }
105 }
106
107 actualizarDisplay(ControlPartida.Instancia.controladorMejoras.sombrerosDisponibles
108     .ElementAt(mejoraMostradaActualmente).Value);
109 }
110
111 public void mejoraAnterior()
112 {
113     mejoraMostradaActualmente--;
114     if (mejoraMostradaActualmente < 0)
115     {
116         mejoraMostradaActualmente = ControlPartida.Instancia.controladorMejoras.sombrerosDisponibles.Count - 1;
117     }
118
119     while (!Convert.ToBoolean(PlayerPrefs.GetInt(ControlPartida.Instancia.controladorMejoras.sombrerosDisponibles
120         .ElementAt(mejoraMostradaActualmente).Value.rutaNecesaria, 0)))
121     {
122         mejoraMostradaActualmente--;
123         if (mejoraMostradaActualmente < 0)
124         {
125             mejoraMostradaActualmente = ControlPartida.Instancia.controladorMejoras.sombrerosDisponibles.Count - 1;
126         }
127     }
128
129     actualizarDisplay(ControlPartida.Instancia.controladorMejoras.sombrerosDisponibles
130         .ElementAt(mejoraMostradaActualmente).Value);
131 }
132
133 public void comprarMejora()
134 {
135     ControlPartida.Instancia.controladorPuntos.puntos -= precio;
136     ControlPartida.Instancia.controladorMejoras.sombreroActivo =
137         ControlPartida.Instancia.controladorMejoras.sombrerosDisponibles.ElementAt(mejoraMostradaActualmente).Value;
138     ControlPartida.Instancia.controladorMejoras.sombrerosObtenidas.Add(
139         ControlPartida.Instancia.controladorMejoras.sombrerosDisponibles.ElementAt(mejoraMostradaActualmente)
140             .Value);
141
142     actualizarDisplay(ControlPartida.Instancia.controladorMejoras.sombrerosDisponibles
143         .ElementAt(mejoraMostradaActualmente).Value);
144
145     string sombrerosObtenidas = "";
146     foreach (Mejora sombrero in ControlPartida.Instancia.controladorMejoras.sombrerosObtenidas)
147     {
148         sombrerosObtenidas += sombrero.name + ",";
149     }
150
151     PlayerPrefs.SetString("Sombrero activo", ControlPartida.Instancia.controladorMejoras.sombreroActivo.name);
152
153     PlayerPrefs.SetString("Sombreros obtenidos", sombrerosObtenidas);
154 }
155
156 public void activarMejora()
157 {
158     ControlPartida.Instancia.controladorMejoras.sombreroActivo =
159         ControlPartida.Instancia.controladorMejoras.sombrerosDisponibles.ElementAt(mejoraMostradaActualmente).Value;
160
161     actualizarDisplay(ControlPartida.Instancia.controladorMejoras.sombrerosDisponibles
162         .ElementAt(mejoraMostradaActualmente).Value);
163
164     PlayerPrefs.SetString("Sombrero activo", ControlPartida.Instancia.controladorMejoras.sombreroActivo.name);
165 }
166 }

```

```

1  @font-face {
2      font-family: myFirstFont;
3      src: url(MajorMonoDisplay-Regular.ttf);
4  }
5
6  html {
7      font-family: myFirstFont, monospace;
8      font-size: 16px;
9      text-transform: lowercase;
10     color: #292f36;
11 }
12
13 body {
14     height: 100vh;
15     background-image: url("img/Fondos/fondo.png");
16     background-repeat: no-repeat;
17     background-size: cover;
18     background-position: right;
19     overflow: clip;
20 }
21
22 .boton-jugar {
23     font-family: myFirstFont, monospace;
24     background-color: #ffa100;
25     border: 32px solid #292f36;
26     color: #292f36;
27     padding: 10px;
28     font-size: 140px;
29     font-weight: bold;
30     position: fixed;
31     cursor: pointer;
32     bottom: -235px;
33     right: -107px;
34     width: 911px;
35     height: 543px;
36     rotate: -16deg;
37     transition: bottom 0.2s ease, right 0.2s ease;
38     text-align: center;
39     text-decoration: none;
40     animation: mover-abajo-derecha 0.2s;
41 }
42 }
43
44 .boton-jugar span {
45     display: block;
46     margin-top: 100px;
47 }
48
49 .boton-jugar:hover {
50     bottom: -165px;
51     right: -60px;
52 }
53
54 .cartel-pium {
55     position: fixed;
56     top: -399px;
57     left: -240px;
58     rotate: -16deg;
59     width: 1044px;
60     height: 700px;
61     background-color: #f7fff7;
62     animation: mover-arriba-izquierda 0.2s;
63 }
64 }
65
66 .cartel-pium h1 {
67     font-size: 160px;
68     margin: 32px;
69     width: 100%;
70     text-align: center;
71     position: fixed;
72     bottom: 0;
73     right: 0;
74 }
75
76 .botones {
77     position: fixed;
78     bottom: 0;
79     left: 0;
80     display: flex;
81     flex-wrap: wrap-reverse;
82     justify-content: start;
83     align-items: end;
84     width: 450px;
85     animation: mover-abajo-izquierda 0.2s;
86 }
87
88 .botones a {
89
90     margin-left: 16px;
91     margin-bottom: 16px;
92     border: 16px solid #292f36;
93     width: 160px;
94     height: 160px;
95     transition: transform 0.1s ease;
96     border-radius: 4px;
97 }
98
99 .boton-descargar {
100     background-color: #ffa100;
101 }

```

```
102
103 .boton-ranking {
104     background-color: #CB48B7;
105 }
106
107 .boton-wiki {
108     background-color: #48ACF0;
109 }
110
111 .botones a:hover {
112     transform: scale(1.1);
113 }
114
115 .botones a img {
116     width: 100%;
117     height: auto;
118 }
119
120 @keyframes mover-abajo-izquierda {
121     from {
122         bottom: -1000px;
123         left: -1000px /* El elemento comienza 100px más abajo de su posición normal */
124     }
125     to { /* El elemento vuelve a su posición normal */
126     }
127 }
128
129 @keyframes mover-arriba-izquierda {
130     from {
131         top: -1000px;
132         left: -1000px /* El elemento comienza 100px más abajo de su posición normal */
133     }
134     to { /* El elemento vuelve a su posición normal */
135     }
136 }
137
138 @keyframes mover-abajo-derecha {
139     from {
140         bottom: -1000px;
141         right: -1000px /* El elemento comienza 100px más abajo de su posición normal */
142     }
143     to {
144     }
145 }
146 }
147
148
149 @media screen and (max-width: 1400px) {
150
151
152     .boton-jugar {
153         font-size: 120px;
154         bottom: -105px;
155         right: -107px;
156         width: 600px;
157         height: 300px;
158     }
159 }
160
161 .boton-jugar span {
162     display: block;
163     margin-top: 40px;
164 }
165
166 .boton-jugar:hover {
167     bottom: -85px;
168     right: -80px;
169 }
170
171 .cartel-pium {
172     top: -150px;
173     left: -140px;
174     width: 800px;
175     height: 400px;
176 }
177 }
178
179 .cartel-pium h1 {
180     font-size: 120px;
181 }
182
183 .botones {
184     width: 260px;
185 }
186
187 .botones a {
188
189     width: 80px;
190     height: 80px;
191 }
192
193 .botones a:hover {
194     transform: scale(1.1);
195 }
196
197
198 }
199
200 @media screen and (max-width: 900px) {
201
202     body {
```

```
203     background-image: url("img/Fondos/fondoMovil.png");
204     background-position: center;
205 }
206
207 .boton-jugar:hover {
208     bottom: -65px;
209     right: -107px;
210 }
211
212
213 .botones {
214     width: 144px;
215     justify-content: center;
216     align-items: start;
217 }
218
219 .botones a {
220     margin: 8px;
221     height: 96px;
222     width: 96px;
223 }
224
225 }
226
227 @media screen and (max-width: 750px) {
228
229     .cartel-pium {
230         top: -200px;
231         left: -50vw;
232         width: 200vw;
233         height: 400px;
234     }
235 }
236
237 .cartel-pium h1 {
238     font-size: 80px;
239 }
240
241
242 .botones {
243     margin: 0;
244     width: 100%;
245     height: 100px;
246
247     rotate: -16deg;
248     bottom: 140px;
249 }
250
251 .botones a {
252     margin: 8px;
253     border: 8px solid #292F36;
254     height: 72px;
255     width: 72px;
256 }
257
258 .boton-jugar {
259     font-size: 60px;
260     bottom: -220px;
261     left: -50vw;
262     width: 200vw;
263     height: 300px;
264     border: 16px solid #292F36;
265 }
266
267
268 .boton-jugar span {
269     display: block;
270     margin-top: 0px;
271 }
272
273 }
```

```
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4   <meta charset="UTF-8">
5   <title>Pium - Wiki</title>
6   <link rel="stylesheet" href="estilo.css">
7   <link rel="stylesheet" href="estiloWiki.css">
8 </head>
9 <body>
10
11 <header>
12   <a class="logo" href="index.html">
13     
14     <h1>Pium</h1>
15   </a>
16   <nav>
17     <ul>
18       <li><a href="paginaRanking.html">Clasificación</a></li>
19       <li><a href="wiki.html">Wiki</a></li>
20       <li><a href="informacion.html#descargas">Descargar</a></li>
21     </ul>
22   </nav>
23 </header>
24
25 <h1 style="margin-top: 88px">Wiki</h1>
26
27 <div class="botonesMejoras">
28   <a class="botonMejoras" href="wikiMejoras.html">Mejoras</a>
29   <a class="botonMejoras" href="wikiNaves.html">Naves</a>
30 </div>
31
32 </body>
33
34
35 </html>
```

```

1 :root {
2   --color-principal: #ffa100;
3   --color-secundario: #292F36;
4 }
5
6 header {
7   display: flex;
8   justify-content: space-between;
9   align-items: center;
10  background-color: #ffffff;
11  padding: 8px;
12  box-shadow: 0px 2px 5px rgba(0, 0, 0, 0.1);
13  position: fixed;
14  top: 0;
15  left: 0;
16  width: calc(100vw - 16px);
17  z-index: 1;
18 }
19
20 .logo {
21   display: flex;
22   align-items: center;
23   text-decoration: none;
24   color: var(--color-secundario);
25   transition: transform 0.1s ease-in;
26 }
27
28 .logo:hover {
29   color: var(--color-principal);
30   transform: scale(1.1);
31 }
32
33 .logo img {
34   width: 50px;
35   height: 50px;
36   margin-right: 10px;
37 }
38
39 nav ul {
40   list-style: none;
41   display: flex;
42 }
43
44 nav li {
45   margin-left: 20px;
46 }
47
48 nav a {
49   text-decoration: none;
50   color: #333333;
51   font-weight: bold;
52   padding: 10px;
53   border-radius: 5px;
54   display: block;
55   transform: scale(1);
56   transition: background-color 0.3s ease, transform 0.3s ease;
57 }
58
59 nav a:hover {
60   background-color: #333333;
61   color: var(--color-principal);
62   transform: scale(1.1);
63 }
64
65 .tituloPagina{
66   padding: 100px 0 16px 0;
67   background-color: #fff;
68 }
69
70 h1 {
71   text-align: center;
72   margin-top: 16px;
73   color: #292f36;
74 }
75
76 h2 {
77   text-align: center;
78   margin-top: 96px;
79 }
80
81 .menu {
82   background-image: url("img/Fondos/fondo.png");
83   background-size: cover;
84 }
85 }
86
87 .vignette {
88   height: 100%;
89   background: radial-gradient(ellipse at center, rgba(0, 0, 0, 0.5) 0%, rgba(0, 0, 0, 0.8) 100%);
90   pointer-events: none;
91
92   display: flex;
93   justify-content: center;
94   align-items: center;
95 }
96
97 .menu .titulo {
98   width: 60%;
99   text-shadow: black 0.1em 0.1em 1em;
100
101 }

```

```
102 .titulo h1{
103     color: #f7fff7;
104 }
105 }
106 .menu .captura-menu img {
107 }
108     max-height: 100%;
109     width: auto;
110 }
111 }
112 .cosoTabla {
113     background-color: #f7fff7;
114     border-radius: 8px;
115     max-width: 1000px;
116     margin: 0 auto;
117     padding: 16px;
118 }
119 }
120 .fondoTabla {
121     padding: 16px;
122     min-height: calc(100vh - 64px);
123     background-color: #292F36;
124 }
125 }
126 }
127 table {
128     width: 100%;
129     text-align: center;
130     font-size: 32px;
131 }
132 }
133 body {
134     margin: 0px;
135     padding: 0px;
136 }
137 }
138 html {
139     font-family: myFirstFont, monospace;
140     font-size: 16px;
141     text-transform: lowercase;
142 }
143 }
144 @font-face {
145     font-family: myFirstFont;
146     src: url(MajorMonoDisplay-Regular.ttf);
147 }
148 }
149 table {
150     border: none;
151 }
152 }
153 tr {
154     transition: transform 0.1s ease, color 0.1s ease, font-weight 0.1s ease;
155 }
156 }
157 tr:hover {
158     transform: scale(1.01);
159     color: #ffa100;
160     font-weight: bolder;
161 }
162 }
163 .posicion-1 {
164     font-size: 64px;
165     font-weight: bold;
166     color: #b10e0e;
167     height: 80px;
168 }
169 }
170 .posicion-2 {
171     font-size: 54px;
172     font-weight: bold;
173     height: 64px;
174 }
175 }
176 }
177 .posicion-3 {
178     font-size: 48px;
179     height: 54px;
180 }
181 }
182 }
183 .apartado {
184     height: 40rem;
185 }
186 }
187 iframe {
188     height: 100%;
189     width: 100%;
190     border: none;
191 }
192 }
193 .principal {
194     height: 40vh;
195     margin-top: 80px;
196     background-color: #292F36;
197     display: flex;
198 }
199 }
200 .principal .titulo {
201     width: 40%;
202     display: flex;
```

```
203     justify-content: center;
204     align-items: center;
205 }
206
207 .principal .titulo h1 {
208     color: #ffa100;
209     font-weight: bolder;
210 }
211
212 .boton {
213     margin: 0 32px;
214     padding: 16px;
215     font-size: 72px;
216     height: 120px;
217     line-height: 120px;
218     font-weight: bold;
219     background-color: #ffa100;
220     color: #292F36;
221     border: 8px solid #292F36;
222     border-radius: 8px;
223     text-decoration: none;
224     transition: transform 0.3s ease;
225 }
226
227 .boton:hover {
228     background-color: #292F36;
229     color: #ffa100;
230     transform: scale(1.1);
231 }
232
233 .categorias {
234     display: flex;
235     justify-content: center;
236     align-items: center;
237 }
238
239 .categorias a {
240     margin: 0 8px;
241     height: 16px;
242     font-weight: bold;
243     background-color: #ffa100;
244     color: #292F36;
245     border: 8px solid #292F36;
246     border-radius: 8px;
247 }
248
249 .descargas {
250     display: flex;
251     justify-content: center;
252     align-items: center;
253     flex-wrap: wrap;
254     background-color: #f7fff7;
255 }
256 .descargas h2{
257     width: 100%;
258 }
```

```
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4   <meta charset="UTF-8">
5   <title>Pium - roguelite multijugador</title>
6   <link rel="stylesheet" href="main.css">
7   <meta name="viewport" content="width=device-width, initial-scale=1">
8 </head>
9 <body>
10
11 <a href="informacion.html" class="boton-jugar">
12   <span>
13     jugar
14   </span>
15 </a>
16
17 <div class="cartel-pium">
18   <h1>Pium</h1>
19 </div>
20
21 <div class="botones">
22   <a class="boton-wiki" href="wiki.html">
23     
24   </a>
25   <a href="paginaRanking.html" class="boton-ranking">
26     
27   </a>
28   <a href="informacion.html#descargas" class="boton-descargar">
29     
30   </a>
31 </div>
32
33 </body>
34 </html>
```

```
1 body {
2   background-color: #292F36;
3   color: #f7fff7;
4 }
5
6 .botonesMejoras {
7   display: flex;
8   flex-wrap: wrap;
9   justify-content: space-evenly;
10  align-items: center;
11  height: 600px;
12 }
13
14 .botonMejoras {
15   border-radius: 2px;
16   border: 8px solid #fdbd0d;
17   height: 160px;
18   width: 320px;
19   transition: transform 0.3s cubic-bezier(0.68, -0.55, 0.13, 3.31);
20   text-align: center;
21   color: #f7fff7;
22   text-decoration: none;
23   font-size: 40px;
24   font-weight: bold;
25   line-height: 160px;
26 }
27
28 .botonMejoras:hover {
29   transform: scale(1.1);
30 }
31
32
33 .TarjetaMejora {
34
35   background-color: #ffffff7;
36   color: #292F36;
37   border-radius: 2px;
38   max-width: 1200px;
39   padding: 16px;
40   margin: 16px auto;
41   display: flex;
42   justify-content: space-between;
43   height: 200px;
44   transition: transform 0.1s ease-in;
45
46 }
47
48 .TarjetaMejora:hover {
49   transform: scale(1.1);
50 }
51
52 .Tier1 {
53   border: 4px solid #1b998b;
54   color: #000000;
55 }
56
57 .Tier2 {
58   border: 4px solid #48acf0;
59   color: #000000;
60 }
61
62 .Tier3 {
63   border: 4px solid #cb48b7;
64   color: #000000;
65 }
66
67 .Tier4 {
68   border: 4px solid #ffbe0b;
69   color: #000000;
70 }
71
72 .Tier5 {
73   border: 4px solid #fd5748;
74   color: #000000;
75 }
76
77 .Tier6 {
78   border: 4px solid #292f36;
79   background-color: #292f36;
80   color: #f7fff7;
81 }
82
83 .IconoMejora {
84   width: 200px;
85   height: 200px;
86 }
87
88 .IconoMejora img {
89   max-width: 200px;
90   height: auto;
91 }
92
93 .TextoMejora {
94   width: 600px;
95   height: auto;
96   display: flex;
97   flex-direction: column;
98   justify-content: center;
99 }
100
101 .TextoMejora span {
```

```
102     font-size: 32px;
103     font-weight: bold;
104 }
105
106 .TextoMejora p {
107     font-size: 16px;
108     font-weight: bold;
109 }
110
111 .TextoMejora .Ruta {
112     font-size: 16px;
113 }
114
115 .Stats {
116     width: 360px;
117     display: flex;
118     flex-direction: column;
119     justify-content: space-evenly;
120 }
121
122 .GrupoHorizontal {
123     display: flex;
124     justify-content: space-between;
125     align-items: center;
126     margin: 8px 0;
127 }
128
129 .Stat {
130     width: 108px;
131     display: flex;
132     align-items: center;
133     flex-direction: column;
134 }
135
136 .Stat span {
137     font-size: 24px;
138     font-weight: bold;
139 }
140
141 .Stat .tituloStat {
142     font-size: 8px;
143     font-weight: bold;
144 }
145
146
147 @media screen and (max-width: 500px) {
148     .TarjetaMejora {
149         flex-direction: column;
150         height: auto;
151     }
152
153     .TextoMejora {
154         width: 100%;
155         margin: 8px 0;
156     }
157
158     .IconoMejora {
159         width: 100%;
160         height: 100%;
161         display: flex;
162         justify-content: center;
163         align-items: center;
164     }
165
166     .IconoMejora img {
167         max-width: 100%;
168         height: auto;
169     }
170 }
```

```

1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4   <meta charset="UTF-8">
5   <title>Pium - Wiki</title>
6   <link rel="stylesheet" href="estilo.css">
7   <link rel="stylesheet" href="estiloWiki.css">
8 </head>
9 <body>
10
11 <header>
12   <a class="logo" href="index.html">
13     
14     <h1>Pium</h1>
15   </a>
16   <nav>
17     <ul>
18       <li><a href="paginaRanking.html">Clasificación</a></li>
19       <li><a href="wiki.html">Wiki</a></li>
20       <li><a href="informacion.html#descargas">Descargar</a></li>
21     </ul>
22   </nav>
23 </header>
24
25 <h1 class="tituloPagina">Wiki - Pagina en desarrollo</h1>
26
27 </body>
28
29 <script !src="">
30
31   //rellenar con los datos de la lista de mejoras en Mejoras.csv
32   //cada mejora es una tarjeta con su icono, su nombre, su descripcion y sus stats
33
34   var request = new XMLHttpRequest();
35   request.open('GET', 'Mejoras.csv', true);
36   request.onload = function () {
37     var data = this.response;
38     var lines = data.split("\n");
39     for (var i = 1; i < lines.length; i++) {
40       var cells = lines[i].split(",");
41
42       let tier = parseFloat(cells[16]);
43
44       //generar la tarjeta de mejora con los datos de la mejora
45       if (cells[2] !== "Modelo") continue;
46
47       var tarjetaMejora = document.createElement("div");
48       // cambiar el color de la tarjeta segun el tier
49       if (tier === 0) {
50         tarjetaMejora.setAttribute("class", "TarjetaMejora");
51       } else if (tier === 1) {
52         tarjetaMejora.setAttribute("class", "TarjetaMejora Tier1");
53       } else if (tier === 2) {
54         tarjetaMejora.setAttribute("class", "TarjetaMejora Tier2");
55       } else if (tier === 3) {
56         tarjetaMejora.setAttribute("class", "TarjetaMejora Tier3");
57       } else if (tier === 4) {
58         tarjetaMejora.setAttribute("class", "TarjetaMejora Tier4");
59       } else if (tier === 5) {
60         tarjetaMejora.setAttribute("class", "TarjetaMejora Tier5");
61       } else if (tier === 6) {
62         tarjetaMejora.setAttribute("class", "TarjetaMejora Tier6");
63         continue;
64       }
65
66       var iconoMejora = document.createElement("div");
67       iconoMejora.setAttribute("class", "IconoMejora");
68       var imgIcono = document.createElement("img");
69       imgIcono.src = "img/IconosMejoras/" + cells[0] + ".png";
70       iconoMejora.appendChild(imgIcono);
71       tarjetaMejora.appendChild(iconoMejora);
72
73       var textoMejora = document.createElement("div");
74       textoMejora.setAttribute("class", "TextoMejora");
75       var titulo = document.createElement("span");
76       titulo.textContent = cells[0];
77       var descripcion = document.createElement("p");
78       descripcion.textContent = cells[1];
79       var ruta = document.createElement("span");
80       ruta.setAttribute("class", "Ruta");
81       ruta.textContent = cells[4];
82       textoMejora.appendChild(titulo);
83       textoMejora.appendChild(descripcion);
84       textoMejora.appendChild(ruta);
85       tarjetaMejora.appendChild(textoMejora);
86
87       var stats = document.createElement("div");
88       stats.setAttribute("class", "Stats");
89
90       var grupoHorizontal = document.createElement("div");
91       grupoHorizontal.setAttribute("class", "GrupoHorizontal");
92       var stat = document.createElement("div");
93       stat.setAttribute("class", "Stat");
94       var nombre = document.createElement("span");
95       nombre.setAttribute("class", "tituloStat");
96       nombre.textContent = "Daño";
97       var valor = document.createElement("span");
98       valor.textContent = cells[5];
99       // put the color of grupoHorizontal in red if the damage is greater than 1
100      if (parseFloat(cells[5]) > 10) {
101        stat.style.color = "#11724a";

```

```

102     } else if (parseFloat(cells[5]) < 10) {
103         stat.style.color = "#a11e1e";
104     }
105     stat.appendChild(nombre);
106     stat.appendChild(valor);
107     grupoHorizontal.appendChild(stat);
108
109     var stat = document.createElement("div");
110     stat.setAttribute("class", "Stat");
111     var nombre = document.createElement("span");
112     nombre.setAttribute("class", "tituloStat");
113     nombre.textContent = "Vida";
114     var valor = document.createElement("span");
115     valor.textContent = cells[6];
116     if (parseFloat(cells[6]) > 50) {
117         stat.style.color = "#11724a";
118     } else if (parseFloat(cells[6]) < 50) {
119         stat.style.color = "#a11e1e";
120     }
121     stat.appendChild(nombre);
122     stat.appendChild(valor);
123     grupoHorizontal.appendChild(stat);
124
125     var stat = document.createElement("div");
126     stat.setAttribute("class", "Stat");
127     var nombre = document.createElement("span");
128     nombre.setAttribute("class", "tituloStat");
129     nombre.textContent = "Tasa de disparo";
130     var valor = document.createElement("span");
131     if (parseFloat(cells[7]) > 0.4) {
132         stat.style.color = "#11724a";
133     } else if (parseFloat(cells[7]) < 0.4) {
134         stat.style.color = "#a11e1e";
135     }
136     valor.textContent = cells[7];
137     stat.appendChild(nombre);
138     stat.appendChild(valor);
139     grupoHorizontal.appendChild(stat);
140
141     stats.appendChild(grupoHorizontal);
142
143     var grupoHorizontal = document.createElement("div");
144     grupoHorizontal.setAttribute("class", "GrupoHorizontal");
145     var stat = document.createElement("div");
146     stat.setAttribute("class", "Stat");
147     var nombre = document.createElement("span");
148     nombre.setAttribute("class", "tituloStat");
149     nombre.textContent = "Dispersion";
150     var valor = document.createElement("span");
151     valor.textContent = cells[8];
152     if (parseFloat(cells[8]) < 5) {
153         stat.style.color = "#11724a";
154     } else if (parseFloat(cells[8]) > 5) {
155         stat.style.color = "#a11e1e";
156     }
157     stat.appendChild(nombre);
158     stat.appendChild(valor);
159     grupoHorizontal.appendChild(stat);
160
161     var stat = document.createElement("div");
162     stat.setAttribute("class", "Stat");
163     var nombre = document.createElement("span");
164     nombre.setAttribute("class", "tituloStat");
165     nombre.textContent = "Velocidad";
166     var valor = document.createElement("span");
167     valor.textContent = cells[9];
168     if (parseFloat(cells[9]) > 7) {
169         stat.style.color = "#11724a";
170     } else if (parseFloat(cells[9]) < 7) {
171         stat.style.color = "#a11e1e";
172     }
173     stat.appendChild(nombre);
174     stat.appendChild(valor);
175     grupoHorizontal.appendChild(stat);
176
177     var stat = document.createElement("div");
178     stat.setAttribute("class", "Stat");
179     var nombre = document.createElement("span");
180     nombre.setAttribute("class", "tituloStat");
181     nombre.textContent = "Aceleracion";
182     var valor = document.createElement("span");
183     valor.textContent = cells[10];
184     if (parseFloat(cells[10]) > 25) {
185         stat.style.color = "#11724a";
186     } else if (parseFloat(cells[10]) < 25) {
187         stat.style.color = "#a11e1e";
188     }
189     stat.appendChild(nombre);
190     stat.appendChild(valor);
191     grupoHorizontal.appendChild(stat);
192
193     stats.appendChild(grupoHorizontal);
194
195     var grupoHorizontal = document.createElement("div");
196     grupoHorizontal.setAttribute("class", "GrupoHorizontal");
197     var stat = document.createElement("div");
198     stat.setAttribute("class", "Stat");
199     var nombre = document.createElement("span");
200     nombre.setAttribute("class", "tituloStat");
201     nombre.textContent = "Rango";
202     var valor = document.createElement("span");

```

```

203     valor.textContent = cells[11];
204     if (parseFloat(cells[11]) > 25) {
205         stat.style.color = "#11724a";
206     } else if (parseFloat(cells[11]) < 25) {
207         stat.style.color = "#a11e1e";
208     }
209     stat.appendChild(nombre);
210     stat.appendChild(valor);
211     grupoHorizontal.appendChild(stat);
212
213     var stat = document.createElement("div");
214     stat.setAttribute("class", "Stat");
215     var nombre = document.createElement("span");
216     nombre.setAttribute("class", "tituloStat");
217     nombre.textContent = "Tiempo vida bala";
218     var valor = document.createElement("span");
219     valor.textContent = cells[12];
220     if (parseFloat(cells[12]) > 5) {
221         stat.style.color = "#11724a";
222     } else if (parseFloat(cells[12]) < 5) {
223         stat.style.color = "#a11e1e";
224     }
225     stat.appendChild(nombre);
226     stat.appendChild(valor);
227     grupoHorizontal.appendChild(stat);
228
229     var stat = document.createElement("div");
230     stat.setAttribute("class", "Stat");
231     var nombre = document.createElement("span");
232     nombre.setAttribute("class", "tituloStat");
233     nombre.textContent = "Velocidad bala";
234     var valor = document.createElement("span");
235     valor.textContent = cells[13];
236     if (parseFloat(cells[13]) > 12) {
237         stat.style.color = "#11724a";
238     } else if (parseFloat(cells[13]) < 12) {
239         stat.style.color = "#a11e1e";
240     }
241     stat.appendChild(nombre);
242     stat.appendChild(valor);
243     grupoHorizontal.appendChild(stat);
244
245     stats.appendChild(grupoHorizontal);
246
247     var grupoHorizontal = document.createElement("div");
248     grupoHorizontal.setAttribute("class", "GrupoHorizontal");
249     var stat = document.createElement("div");
250     stat.setAttribute("class", "Stat");
251     var nombre = document.createElement("span");
252     nombre.setAttribute("class", "tituloStat");
253     nombre.textContent = "Tamaño bala";
254     var valor = document.createElement("span");
255     valor.textContent = cells[14];
256     if (parseFloat(cells[14]) > 1) {
257         stat.style.color = "#11724a";
258     } else if (parseFloat(cells[14]) < 1) {
259         stat.style.color = "#a11e1e";
260     }
261     stat.appendChild(nombre);
262     stat.appendChild(valor);
263     grupoHorizontal.appendChild(stat);
264
265     var stat = document.createElement("div");
266     stat.setAttribute("class", "Stat");
267     var nombre = document.createElement("span");
268     nombre.setAttribute("class", "tituloStat");
269     nombre.textContent = "Balas";
270     var valor = document.createElement("span");
271     valor.textContent = cells[15];
272     if (parseFloat(cells[15]) > 1) {
273         stat.style.color = "#11724a";
274     }
275     stat.appendChild(nombre);
276     stat.appendChild(valor);
277     grupoHorizontal.appendChild(stat);
278
279     var stat = document.createElement("div");
280     stat.setAttribute("class", "Stat");
281     var nombre = document.createElement("span");
282     nombre.setAttribute("class", "tituloStat");
283     nombre.textContent = "Tier mejora";
284     var valor = document.createElement("span");
285     valor.textContent = cells[16];
286     stat.appendChild(nombre);
287     stat.appendChild(valor);
288     grupoHorizontal.appendChild(stat);
289
290     stats.appendChild(grupoHorizontal);
291
292     tarjetaMejora.appendChild(stats);
293
294     document.body.appendChild(tarjetaMejora);
295
296 }
297 }
298 }
299 request.send();
300
301 </script>
302
303

```



```
1 <html>
2 <head>
3   <title>Leaderboard</title>
4   <link rel="stylesheet" href="estilo.css">
5   <meta charset="utf-8">
6   <meta name="viewport" content="width=device-width, initial-scale=1">
7
8 </head>
9 <body>
10 <header>
11   <a class="logo" href="index.html">
12     
13     <h1>Pium</h1>
14   </a>
15   <nav>
16     <ul>
17       <li><a href="paginaRanking.html">Clasificación</a></li>
18       <li><a href="wiki.html">Wiki</a></li>
19       <li><a href="informacion.html#descargas">Descargar</a></li>
20     </ul>
21   </nav>
22 </header>
23 <div class="apartado principal">
24   <iframe src="https://www.youtube.com/embed/x-ZqZqr_PmE" title="Pium"
25     allow="accelerometer; autoplay; clipboard-write; encrypted-media; gyroscope; picture-in-picture; web-share"
26     allowfullscreen></iframe>
27
28 </div>
29
30 <div class="apartado menu">
31   <div class="vignette">
32     <div class="titulo">
33       <h1>Pium, juega con tus amigos en este rogulite con sistema de puntuacion global</h1>
34     </div>
35   </div>
36 </div>
37
38 <div class="apartado">
39   <!-- Pantalla donde muestra todas las mejoras en dos columnas verticales -->
40
41   <div class="apartado descargas" id="descargas">
42     <h2>Descarga para la plataforma que mas te guste</h2>
43     <a class="boton" href="Pium.zip">Windows</a>
44     <a class="boton" href="a.apk">Android</a>
45   </div>
46 </div>
47 </body>
48 </html>
49
```

```
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4   <meta charset="UTF-8">
5   <title>Pium - Wiki</title>
6   <link rel="stylesheet" href="estilo.css">
7   <link rel="stylesheet" href="estiloWiki.css">
8   <meta name="viewport" content="width=device-width, initial-scale=1">
9
10 </head>
11 <body>
12
13 <header>
14   <a class="logo" href="index.html">
15     
16     <h1>Pium</h1>
17   </a>
18   <nav>
19     <ul>
20       <li><a href="paginaRanking.html">Clasificación</a></li>
21       <li><a href="wiki.html">Wiki</a></li>
22       <li><a href="informacion.html#descargas">Descargar</a></li>
23     </ul>
24   </nav>
25 </header>
26
27 <h1 class="tituloPagina">Wiki - Pagina en desarrollo</h1>
28
29 </body>
30
31 <script !src="">
32
33   //rellenar con los datos de la lista de mejoras en Mejoras.csv
34   //cada mejora es una tarjeta con su icono, su nombre, su descripcion y sus stats
35
36   var request = new XMLHttpRequest();
37   request.open('GET', 'Mejoras.csv', true);
38   request.onload = function () {
39     var data = this.response;
40     var lines = data.split("\n");
41     for (var i = lines.length - 1; i > 0; i--) {
42       var cells = lines[i].split(",");
43
44       //generar la tarjeta de mejora con los datos de la mejora
45       if (cells[2] === "Modelo") continue;
46       if (cells[2] === "Sombreros") continue;
47
48       let tier = parseFloat(cells[16]);
49
50
51       var tarjetaMejora = document.createElement("div");
52
53       // cambiar el color de la tarjeta segun el tier
54       if (tier === 0) {
55         tarjetaMejora.setAttribute("class", "TarjetaMejora");
56       } else if (tier === 1) {
57         tarjetaMejora.setAttribute("class", "TarjetaMejora Tier1");
58       } else if (tier === 2) {
59         tarjetaMejora.setAttribute("class", "TarjetaMejora Tier2");
60       } else if (tier === 3) {
61         tarjetaMejora.setAttribute("class", "TarjetaMejora Tier3");
62       } else if (tier === 4) {
63         tarjetaMejora.setAttribute("class", "TarjetaMejora Tier4");
64       } else if (tier === 5) {
65         tarjetaMejora.setAttribute("class", "TarjetaMejora Tier5");
66       } else if (tier === 6) {
67         tarjetaMejora.setAttribute("class", "TarjetaMejora Tier6");
68         continue;
69       }
70
71       var iconoMejora = document.createElement("div");
72       iconoMejora.setAttribute("class", "IconoMejora");
73       var imgIcono = document.createElement("img");
74       imgIcono.src = "img/IconosMejoras/" + cells[0] + ".png";
75       iconoMejora.appendChild(imgIcono);
76       tarjetaMejora.appendChild(iconoMejora);
77
78       var textoMejora = document.createElement("div");
79       textoMejora.setAttribute("class", "TextoMejora");
80       var titulo = document.createElement("span");
81       titulo.textContent = cells[0];
82       var descripcion = document.createElement("p");
83       descripcion.textContent = cells[1];
84       var ruta = document.createElement("span");
85       ruta.setAttribute("class", "Ruta");
86       ruta.textContent = cells[4];
87       textoMejora.appendChild(titulo);
88       textoMejora.appendChild(descripcion);
89       textoMejora.appendChild(ruta);
90       tarjetaMejora.appendChild(textoMejora);
91
92       var stats = document.createElement("div");
93       stats.setAttribute("class", "Stats");
94
95       var grupoHorizontal = document.createElement("div");
96       grupoHorizontal.setAttribute("class", "GrupoHorizontal");
97       var stat = document.createElement("div");
98       stat.setAttribute("class", "Stat");
99       var nombre = document.createElement("span");
100      nombre.setAttribute("class", "tituloStat");
101      nombre.textContent = "Daño";
```

```

102     var valor = document.createElement("span");
103     valor.textContent = cells[5];
104     // put the color of grupoHorizontal in red if the damage is greater than 1
105     if (parseFloat(cells[5]) > 1) {
106         stat.style.color = "#11724a";
107     } else if (parseFloat(cells[5]) < 1) {
108         stat.style.color = "#a11e1e";
109     }
110     stat.appendChild(nombre);
111     stat.appendChild(valor);
112     grupoHorizontal.appendChild(stat);
113
114     var stat = document.createElement("div");
115     stat.setAttribute("class", "Stat");
116     var nombre = document.createElement("span");
117     nombre.setAttribute("class", "tituloStat");
118     nombre.textContent = "Vida";
119     var valor = document.createElement("span");
120     valor.textContent = cells[6];
121     if (parseFloat(cells[6]) > 1) {
122         stat.style.color = "#11724a";
123     } else if (parseFloat(cells[6]) < 1) {
124         stat.style.color = "#a11e1e";
125     }
126     stat.appendChild(nombre);
127     stat.appendChild(valor);
128     grupoHorizontal.appendChild(stat);
129
130     var stat = document.createElement("div");
131     stat.setAttribute("class", "Stat");
132     var nombre = document.createElement("span");
133     nombre.setAttribute("class", "tituloStat");
134     nombre.textContent = "Tasa de disparo";
135     var valor = document.createElement("span");
136     if (parseFloat(cells[7]) > 1) {
137         stat.style.color = "#11724a";
138     } else if (parseFloat(cells[7]) < 1) {
139         stat.style.color = "#a11e1e";
140     }
141     valor.textContent = cells[7];
142     stat.appendChild(nombre);
143     stat.appendChild(valor);
144     grupoHorizontal.appendChild(stat);
145
146     stats.appendChild(grupoHorizontal);
147
148     var grupoHorizontal = document.createElement("div");
149     grupoHorizontal.setAttribute("class", "GrupoHorizontal");
150     var stat = document.createElement("div");
151     stat.setAttribute("class", "Stat");
152     var nombre = document.createElement("span");
153     nombre.setAttribute("class", "tituloStat");
154     nombre.textContent = "Dispersion";
155     var valor = document.createElement("span");
156     valor.textContent = cells[8];
157     if (parseFloat(cells[8]) < 1) {
158         stat.style.color = "#11724a";
159     } else if (parseFloat(cells[8]) > 1) {
160         stat.style.color = "#a11e1e";
161     }
162     stat.appendChild(nombre);
163     stat.appendChild(valor);
164     grupoHorizontal.appendChild(stat);
165
166     var stat = document.createElement("div");
167     stat.setAttribute("class", "Stat");
168     var nombre = document.createElement("span");
169     nombre.setAttribute("class", "tituloStat");
170     nombre.textContent = "Velocidad";
171     var valor = document.createElement("span");
172     valor.textContent = cells[9];
173     if (parseFloat(cells[9]) > 1) {
174         stat.style.color = "#11724a";
175     } else if (parseFloat(cells[9]) < 1) {
176         stat.style.color = "#a11e1e";
177     }
178     stat.appendChild(nombre);
179     stat.appendChild(valor);
180     grupoHorizontal.appendChild(stat);
181
182     var stat = document.createElement("div");
183     stat.setAttribute("class", "Stat");
184     var nombre = document.createElement("span");
185     nombre.setAttribute("class", "tituloStat");
186     nombre.textContent = "Aceleracion";
187     var valor = document.createElement("span");
188     valor.textContent = cells[10];
189     if (parseFloat(cells[10]) > 1) {
190         stat.style.color = "#11724a";
191     } else if (parseFloat(cells[10]) < 1) {
192         stat.style.color = "#a11e1e";
193     }
194     stat.appendChild(nombre);
195     stat.appendChild(valor);
196     grupoHorizontal.appendChild(stat);
197
198     stats.appendChild(grupoHorizontal);
199
200     var grupoHorizontal = document.createElement("div");
201     grupoHorizontal.setAttribute("class", "GrupoHorizontal");
202     var stat = document.createElement("div");

```

```

203     stat.setAttribute("class", "Stat");
204     var nombre = document.createElement("span");
205     nombre.setAttribute("class", "tituloStat");
206     nombre.textContent = "Rango";
207     var valor = document.createElement("span");
208     valor.textContent = cells[11];
209     if (parseFloat(cells[11]) > 1) {
210         stat.style.color = "#11724a";
211     } else if (parseFloat(cells[11]) < 1) {
212         stat.style.color = "#a11e1e";
213     }
214     stat.appendChild(nombre);
215     stat.appendChild(valor);
216     grupoHorizontal.appendChild(stat);
217
218     var stat = document.createElement("div");
219     stat.setAttribute("class", "Stat");
220     var nombre = document.createElement("span");
221     nombre.setAttribute("class", "tituloStat");
222     nombre.textContent = "Tiempo vida bala";
223     var valor = document.createElement("span");
224     valor.textContent = cells[12];
225     if (parseFloat(cells[12]) > 1) {
226         stat.style.color = "#11724a";
227     } else if (parseFloat(cells[12]) < 1) {
228         stat.style.color = "#a11e1e";
229     }
230     stat.appendChild(nombre);
231     stat.appendChild(valor);
232     grupoHorizontal.appendChild(stat);
233
234     var stat = document.createElement("div");
235     stat.setAttribute("class", "Stat");
236     var nombre = document.createElement("span");
237     nombre.setAttribute("class", "tituloStat");
238     nombre.textContent = "Velocidad bala";
239     var valor = document.createElement("span");
240     valor.textContent = cells[13];
241     if (parseFloat(cells[13]) > 1) {
242         stat.style.color = "#11724a";
243     } else if (parseFloat(cells[13]) < 1) {
244         stat.style.color = "#a11e1e";
245     }
246     stat.appendChild(nombre);
247     stat.appendChild(valor);
248     grupoHorizontal.appendChild(stat);
249
250     stats.appendChild(grupoHorizontal);
251
252     var grupoHorizontal = document.createElement("div");
253     grupoHorizontal.setAttribute("class", "GrupoHorizontal");
254     var stat = document.createElement("div");
255     stat.setAttribute("class", "Stat");
256     var nombre = document.createElement("span");
257     nombre.setAttribute("class", "tituloStat");
258     nombre.textContent = "Tamaño bala";
259     var valor = document.createElement("span");
260     valor.textContent = cells[14];
261     if (parseFloat(cells[14]) > 1) {
262         stat.style.color = "#11724a";
263     } else if (parseFloat(cells[14]) < 1) {
264         stat.style.color = "#a11e1e";
265     }
266     stat.appendChild(nombre);
267     stat.appendChild(valor);
268     grupoHorizontal.appendChild(stat);
269
270     var stat = document.createElement("div");
271     stat.setAttribute("class", "Stat");
272     var nombre = document.createElement("span");
273     nombre.setAttribute("class", "tituloStat");
274     nombre.textContent = "Balas adicionales";
275     var valor = document.createElement("span");
276     valor.textContent = cells[15];
277     if (parseFloat(cells[15]) > 0) {
278         stat.style.color = "#11724a";
279     }
280     stat.appendChild(nombre);
281     stat.appendChild(valor);
282     grupoHorizontal.appendChild(stat);
283
284     var stat = document.createElement("div");
285     stat.setAttribute("class", "Stat");
286     var nombre = document.createElement("span");
287     nombre.setAttribute("class", "tituloStat");
288     nombre.textContent = "Tier mejora";
289     var valor = document.createElement("span");
290     valor.textContent = cells[16];
291     stat.appendChild(nombre);
292     stat.appendChild(valor);
293     grupoHorizontal.appendChild(stat);
294
295     stats.appendChild(grupoHorizontal);
296
297     tarjetaMejora.appendChild(stats);
298
299     document.body.appendChild(tarjetaMejora);
300
301 }
302 }
303

```

```
304     request.send();
305
306 </script>
307
308
309 </html>
```

```

1 <html>
2 <head>
3   <title>Leaderboard</title>
4   <link rel="stylesheet" href="estilo.css">
5   <meta charset="utf-8">
6 </head>
7 <body>
8 <header>
9   <a class="logo" href="index.html">
10     
11     <h1>Pium</h1>
12   </a>
13   <nav>
14     <ul>
15       <li><a href="paginaRanking.html">Clasificación</a></li>
16       <li><a href="wiki.html">Wiki</a></li>
17       <li><a href="informacion.html#descargas">Descargar</a></li>
18     </ul>
19   </nav>
20 </header>
21 <h2>Clasificación</h2>
22 <div class="fondoTabla">
23   <div class="cosoTabla">
24     <table>
25       <thead>
26         <tr>
27           <th>Rango</th>
28           <th>Nombre</th>
29           <th>Puntuación</th>
30         </tr>
31       </thead>
32       <tbody id="leaderboard-table"></tbody>
33     </table>
34   </div>
35 </div>
36
37 <script>
38
39   function Authentication() {
40     // Game API key
41     const gameAPIKey = 'dev_e40175ee00034400bc562bbcf016cadc'
42
43     // Leaderboard key
44     const leaderboardKey = '10024'
45
46     // Development mode true/false
47     const developmentMode = 'true'
48
49     // Authentication request
50     const AuthHttp = new XMLHttpRequest()
51     const auth_url = 'https://api.lootlocker.io/game/v2/session/guest'
52     AuthHttp.open('POST', auth_url)
53     AuthHttp.setRequestHeader('Content-Type', 'application/json')
54
55     // No player identifier was found, start new session
56     AuthHttp.send(
57       JSON.stringify({game_key: gameAPIKey, game_version: '0.1.0.0', development_mode: developmentMode})
58     )
59
60     AuthHttp.onreadystatechange = e => {
61       // Log server response
62       if (AuthHttp.responseText.length == 0) return
63
64
65       const arr = JSON.parse(AuthHttp.responseText);
66       console.log(arr);
67       let session_token = arr.session_token
68
69       // Get leaderboard data
70       const leaderboardHTTP = new XMLHttpRequest()
71       const amountOfEntries = 100
72       const leaderboardUrl =
73         'https://api.lootlocker.io/game/leaderboards/' + leaderboardKey + '/list?count=' + amountOfEntries
74       leaderboardHTTP.open('GET', leaderboardUrl)
75       leaderboardHTTP.setRequestHeader('Content-Type', 'application/json')
76
77       // Add session token
78       leaderboardHTTP.setRequestHeader(
79         'x-session-token',
80         session_token
81       )
82       // Send leaderboard request
83       leaderboardHTTP.send('')
84       leaderboardHTTP.onreadystatechange = e => {
85         let leaderboardData = ''
86
87         if (leaderboardHTTP.responseText.length == 0) return
88         console.log(leaderboardHTTP.responseText)
89         leaderboardData = JSON.parse(leaderboardHTTP.responseText)
90         const leaderboardTable = document.getElementById('leaderboard-table');
91
92         leaderboardTable.innerHTML = '';
93
94         leaderboardData.items.forEach(player => {
95           const row = document.createElement('tr');
96           const rankCell = document.createElement('td');
97           const nameCell = document.createElement('td');
98           const scoreCell = document.createElement('td');
99
100           rankCell.innerText = player.rank + "°"
101           nameCell.innerText = player.member_id;

```

```
102         scoreCell.innerText = player.score;
103
104         row.appendChild(rankCell);
105         row.appendChild(nameCell);
106         row.appendChild(scoreCell);
107         row.className = "posicion-" + player.rank;
108
109         leaderboardTable.appendChild(row);
110         //const leaderboardResponseText = JSON.parse(leaderboardHTTP.responseText)
111     })
112 }
113 }
114 }
115
116 // Ejecutamos la función principal
117 Authentication();
118
119 </script>
120 </body>
121 </html>
122
```