```
using System.Collections;
using UnityEngine;
using UnityEngine.UI;
using Photon.Pun;
using System.Collections.Generic;
using PlayFab;
using PlayFab.ClientModels;
using UnityEngine.SceneManagement;

public class Coche : MonoBehaviour
{

    ExitGames.Client.Photon.Hashtable propiedadesJugador = new ExitGames.Client.Photon.Hashtable();
    //ESTA CLASE RECOGE LAS CARACTERISTICAS COMUNES A TODOS LOS COCHES TALES COMO EL MOVIMIENTO, EFECTOS, ETC
    public Rigidbody esfera;
    public string nickJugador;

    #region ESTADISTICAS BASICAS
    [Header("Estadísticas Básicas")]
    public float aceleracion = 8f;
    public float marchaAtras = 4f;
    public float fuerzaGiro = 80f;
    public float fuerzaGiroOriginal = 80f;
    public float gravedad = 10f;
    public float dragEnSuelo = 3f;
    public int hp = 100;
    public int maxHP = 100;
    public int posicion;
    public bool boosted = false;
    public float cantidadBoost;

    public float tiempoCircuito;

    #endregion

    #region UTILIDADES MOVIMIENTO
    [Header("Movimiento")]
    public LayerMask suelo;
    public LayerMask carretera;
    public float longitudRayoSuelo = 0.5f;
    public Transform puntoRayoSuelo;
    public Transform ruedaIzquierda, ruedaDerecha;
    public float maximaRotacionRuedas = 25f;
    private bool tocandoSuelo;
    private float multiplicador = 1000f;
    private float velocidadInput, giroInput;

    #endregion

    #region UTILIDADES DERRAPE
    private bool drifting;
    private int direccionDrift;
    private float tiempoDrift = 0;
    private GameObject carModel;

    #endregion

    #region COOLDOWNS
    [Header("CoolDowns")]
    protected bool[] isCD = new bool[4];
    protected float[] coolDowns = new float[4];
    protected float[] timerCD = new float[4];

    #endregion
```

```
#region HUD
[Header("HUD")]
public GameObject habilidades;
public Image[] imagenes = new Image[4];

public Text textoVida;
public Image barraVida;
public Text contadorVueltas;
public GameObject panelGas;
public Spawn spawn;
private GameObject camaras;

#endregion

#region INFORMACION JUGADOR
private string nombreJugador;
private int vuelta = 0;
private int numPuntoControl;
private float distanciaSiguientePunto;
private GameObject puntosControl;
private GameObject rank;
#endregion

#region UTILIDADES HABILIDADES
[Header("Habilidades")]
public Transform puntoPrefabs;
public bool invencible = false;
public bool protegido = false;
public bool ralentizado = false;
public bool resbalado = false;
public bool stuneado = false;
#endregion

#region ONLINE
protected PhotonView vista;
#endregion

#region VISUAL
[Header("Partículas")]
public GameObject particulasParent;
private Color[] colores = new Color[3];
private ParticleSystem[] particulas = new ParticleSystem[4];
#endregion

private bool acabado = false;

//================FUNCIONES================

#region MOVIMIENTO COCHE
public void RecogerInputMovimientoBasico() {
    velocidadInput = 0f;
    //RECOGIDA ACELERACION
    if (Input.GetAxis("Vertical") > 0)
    {
        //ACELERA HACIA DELANTE
        velocidadInput = Input.GetAxis("Vertical") * aceleracion * multiplicador;
    }
    else if (Input.GetAxis("Vertical") < 0)
    {
        //ACELERA MARCHA ATRAS
        velocidadInput = Input.GetAxis("Vertical") * marchaAtras * multiplicador;
    }
```

```
        //RECOGIDA DIRECCION
        giroInput = Input.GetAxis("Horizontal");

        if (tocandoSuelo)
        {
            transform.rotation = Quaternion.Euler(transform.eulerAngles + new Vector3(0, giroInput * fuerzaGiro * Time.deltaTime *
Input.GetAxis("Vertical"), 0f));
        }

        //ROTACION RUEDAS #ESTETICO
        ruedaIzquierda.localRotation = Quaternion.Euler(ruedaIzquierda.localRotation.eulerAngles.x, (giroInput *
maximaRotacionRuedas) - 180, ruedaIzquierda.localRotation.eulerAngles.z);
        ruedaDerecha.localRotation = Quaternion.Euler(ruedaDerecha.localRotation.eulerAngles.x, giroInput *
maximaRotacionRuedas, ruedaDerecha.localRotation.eulerAngles.z);

        transform.position = esfera.position - new Vector3(0, 0.275f, 0);


        //CAMBIO CAMARA (RETROVISOR)
        if (Input.GetKey(KeyCode.R))
        {
            camaras.transform.GetChild(0).gameObject.SetActive(false);
            camaras.transform.GetChild(2).gameObject.SetActive(true);
        } else {
            camaras.transform.GetChild(2).gameObject.SetActive(false);
            camaras.transform.GetChild(0).gameObject.SetActive(true);
        }
    }

    public void RecogerInputDerrape() {

        //EL JUGADOR ESTÁ PULSANDO ESPACIO, NO ESTÁ DERRAPANDO YA Y ESTÁ GIRANDO A IZQ O DRCH
        if (Input.GetButtonDown("Jump") && !drifting && Input.GetAxis("Horizontal")!=0) {
            drifting = true;
            direccionDrift = Input.GetAxis("Horizontal") > 0 ? 1 : -1;
            //GIRAMOS EL MODELO EN LA DIRECCION DEL DERRAPE
            carModel.transform.eulerAngles = carModel.transform.eulerAngles + new Vector3(0, 25 * direccionDrift, 0);
            particulasParent.transform.eulerAngles = carModel.transform.eulerAngles;
            fuerzaGiro /= 2f;
            for (int i = 0; i < particulas.Length; i++) {
                particulas[i].startColor = colores[0];
                particulas[i].Play();
            }
        }

        if (drifting)
        {

            tiempoDrift += Time.deltaTime;
            if (tiempoDrift > 2 && tiempoDrift <= 4)
            {
                for (int i = 0; i < particulas.Length; i++)
                {
                    particulas[i].startColor = colores[1];
                }
            }
            else if (tiempoDrift > 4)
            {
                for (int i = 0; i < particulas.Length; i++)
                {
                    particulas[i].startColor = colores[2];
                }
            }
        }
        else {
            carModel.transform.eulerAngles = this.transform.eulerAngles;
```

```
        particulasParent.transform.eulerAngles = carModel.transform.eulerAngles;
        for (int i = 0; i < particulas.Length; i++)
        {
            particulas[i].Stop();
        }
    }


    if (Input.GetButtonUp("Jump")) {
        drifting = false;
        int boost = 0;
        if (tiempoDrift > 1 && tiempoDrift <= 2) {
            boost = 7000;
        } else if (tiempoDrift > 2 && tiempoDrift <= 4) {
            boost = 9000;
        } else if (tiempoDrift > 4) {
            boost = 11000;
        }
        if (boost > 0) {
            RecibirBoost(boost);
        }

        Debug.Log("Boost recibido: " + boost);
        tiempoDrift = 0;
        fuerzaGiro = fuerzaGiroOriginal;
    }
}


public void AplicarVelocidad()
{
    if (acabado || spawn.i > 0 || stuneado)
        return;

    tocandoSuelo = false;
    RaycastHit hit;

    //COMPROBACION TOCANDO SUELO
    if (Physics.Raycast(puntoRayoSuelo.position, -transform.up, out hit, longitudRayoSuelo, carretera))
    {
        tocandoSuelo = true;

        transform.rotation = Quaternion.FromToRotation(transform.up, hit.normal) * transform.rotation;
    }
    else if (Physics.Raycast(puntoRayoSuelo.position, -transform.up, out hit, longitudRayoSuelo, suelo))
    {

        tocandoSuelo = true;
        transform.rotation = Quaternion.FromToRotation(transform.up, hit.normal) * transform.rotation;
        if (!invencible) {
            velocidadInput /= 2;
        }
    }

    if (ralentizado)
        velocidadInput /= 3;

    if (drifting)
        velocidadInput /= 1.3f;

    if (tocandoSuelo)
    {
        esfera.drag = dragEnSuelo;
        if (Mathf.Abs(velocidadInput) > 0)
        {
            //SUMAR VELOCIDAD
```

```
            esfera.AddForce(transform.forward * velocidadInput);
        }
    }
    else
    {
        //ESTA EN EL AIRE
        esfera.drag = 0.1f;
        esfera.AddForce(Vector3.up * -gravedad * 100f);
    }
}
#endregion


#region ACTUALIZACION HUD
public void SumaVuelta()
{
    if (vista.IsMine) {
        propiedadesJugador = PhotonNetwork.LocalPlayer.CustomProperties;
        vuelta++;
        propiedadesJugador["jugadorVuelta"] = vuelta;
        //propiedadesJugador["jugadorVuelta"] = 0;
        //PhotonNetwork.LocalPlayer.CustomProperties = propiedadesJugador;
        PhotonNetwork.SetPlayerCustomProperties(propiedadesJugador);
        ActualizarRanking();
        Debug.Log("Vuelta: "+vuelta);
        if (vuelta <= 3)
        {
            contadorVueltas.text = vuelta + "/3";
        }
        else
        {
            //ESTE JUGADOR HA ACABADO
            propiedadesJugador = PhotonNetwork.LocalPlayer.CustomProperties;
            acabado = true;
            int puestoFinal = rank.GetComponent<Ranking>().MiPuestoFinal();
            propiedadesJugador["jugadorPFinal"] = puestoFinal;
            if (propiedadesJugador.ContainsKey("jugadorPunFinal"))
            {
                propiedadesJugador["jugadorPunFinal"] = (int)propiedadesJugador["jugadorPunFinal"] +
(PhotonNetwork.CurrentRoom.PlayerCount+1 - puestoFinal) * 5;
                Debug.Log(propiedadesJugador["jugadorPunFinal"]);
            }
            else {
                propiedadesJugador["jugadorPunFinal"] = (PhotonNetwork.CurrentRoom.PlayerCount+1 - puestoFinal) * 5;
            }
            //PhotonNetwork.LocalPlayer.CustomProperties = propiedadesJugador;
            PhotonNetwork.SetPlayerCustomProperties(propiedadesJugador);
            GuardarTiempo();
        }
    }
}

public void ActualizaControl(int n)
{
    propiedadesJugador = PhotonNetwork.LocalPlayer.CustomProperties;
    numPuntoControl =n;
    propiedadesJugador["jugadorPuntoControl"] = numPuntoControl;
    //PhotonNetwork.LocalPlayer.CustomProperties = propiedadesJugador;
    PhotonNetwork.SetPlayerCustomProperties(propiedadesJugador);
    ActualizarRanking();

}

public void ActualizarRanking() {
    //ESTA FUNCION ACTUALIZA EL HUD PARA VER EL RANKING IN GAME
    if (vista.IsMine)
```

```
    {
        rank.GetComponent<Ranking>().ActualizarPosiciones();
    }
}

public void CargarCooldowns(int cdh1,int cdh2,int cdh3,int cdh4) {
    if (vista.IsMine)
    {
        coolDowns[0] = cdh1;
        coolDowns[1] = cdh2;
        coolDowns[2] = cdh3;
        coolDowns[3] = cdh4;
        for (int i = 0; i < timerCD.Length; i++)
        {
            //timerCD[i] = coolDowns[i];
            timerCD[i] = 0;
            isCD[i] = true;
        }
    }
}

#endregion

#region EFECTOS SOBRE JUGADOR (HABILIDADES)
public void RecibirBoost(int cantidad) {
    boosted = true;
    cantidadBoost = cantidad;
    StartCoroutine(DesactivarBoost());
}
public void RecibirResbalar(int min,int max)
{
    if (invencible)
        return;
    if (protegido)
    {
        protegido = false;
    }
    else
    {
        int i = Random.Range(0, 1);
        int o = Random.Range(0, 1);
        if (i == 0)
        {
            i = -1;
        }
        if (o == 0)
        {
            o = -1;
        }
        int x, z;
        x = Random.Range(min, max);
        z = Random.Range(min, max);
        x *= i;
        z *= o;
        esfera.AddForce(new Vector3(x, 0, z), ForceMode.Impulse);
    }
}

public void RecibirResbalar() {
    if (invencible)
        return;
    if (protegido)
    {
        protegido = false;
    }
    else
```

```
    {
        int i = Random.Range(0, 1);
        int o = Random.Range(0, 1);
        if (i == 0)
        {
            i = -1;
        }
        if (o == 0)
        {
            o = -1;
        }
        int x, z;
        x = Random.Range(500, 1500);
        z = Random.Range(500, 1500);
        x *= i;
        z *= o;
        esfera.AddForce(new Vector3(x, 0, z),ForceMode.Impulse);
    }
}

public void RecibirRalentizar(int tiempo)
{
    if (invencible)
        return;

    if (protegido)
    {
        protegido = false;
    }
    else
    {
        ralentizado = true;
        StartCoroutine(DesactivarRalentizado(tiempo));
    }
}
public void RecibirStun(int tiempo)
{
    if (invencible)
        return;

    if (protegido)
    {
        protegido = false;
    }
    else
    {
        stuneado = true;
        StartCoroutine(DesactivarStuneado(tiempo));
    }
}

public IEnumerator AplicarCeguera() {
    if (vista.IsMine)
    {
        if (!invencible)
        {
            if (protegido)
            {
                protegido = false;
            }
            else
            {
                panelGas.SetActive(true);
                yield return new WaitForSeconds(5);
                panelGas.SetActive(false);
```

```
            }
        }
    }
}

public GameObject SoltarPrefab(GameObject prefab)
{
    GameObject go = PhotonNetwork.Instantiate(prefab.name, puntoPrefabs.position, transform.rotation);
    return go;
}

public GameObject SoltarPrefab(GameObject prefab,Vector3 offset) {
    GameObject go = PhotonNetwork.Instantiate(prefab.name, puntoPrefabs.position+offset, transform.rotation);
    return go;
}

protected void ReducirCoolDown(int i)
{
    if (vista.IsMine)
    {
        if (acabado || spawn.i > 0)
            return;
        if (timerCD[i] < coolDowns[i])
        {
            imagenes[i].color = Color.red;
            timerCD[i] += Time.deltaTime;
            imagenes[i].fillAmount = timerCD[i] / coolDowns[i];
        }
        else if (timerCD[i] >= coolDowns[i])
        {
            imagenes[i].color = Color.cyan;
            timerCD[i] = coolDowns[i];
            imagenes[i].fillAmount = 1;
            isCD[i] = false;
        }

    }
}

public int RecogerInputHabilidades()
{
    if (acabado || spawn.i > 0 || stuneado)
        return 5;
    if (Input.GetKeyDown(KeyCode.U))
    {
        return 0;
    }
    else if (Input.GetKeyDown(KeyCode.I))
    {
        return 1;
    }
    else if (Input.GetKeyDown(KeyCode.O))
    {
        return 2;
    }
    else if (Input.GetKeyDown(KeyCode.P))
    {
        return 3;
    }

    return 5;
}

public void ModificarSize(float factor,int tiempo,bool y) {
    Vector3 escala;
    Vector3 escalaEsfera;
```

```
        if (y)
        {
            escala = new Vector3(transform.localScale.x * factor, transform.localScale.y * factor, transform.localScale.z * factor);
            escalaEsfera = escala;
        }
        else {
            escala = new Vector3(transform.localScale.x * factor, 0.5f, transform.localScale.z * factor);
            escalaEsfera = new Vector3(transform.localScale.x * factor, transform.localScale.y * factor, transform.localScale.z *
factor);
        }

        transform.localScale = escala;
        esfera.transform.localScale = escalaEsfera;
        StartCoroutine(DesactivarCambioSize(factor,tiempo,y));
    }

    public IEnumerator DesactivarCambioSize(float factor,int tiempo,bool y) {
        yield return new WaitForSeconds(tiempo);
        Vector3 escala;
        Vector3 escalaEsfera;
        if (y)
        {
            escala = new Vector3(transform.localScale.x / factor, transform.localScale.y / factor, transform.localScale.z / factor);
            escalaEsfera = escala;
        }
        else
        {
            escala = new Vector3(transform.localScale.x / factor, 0.5f, transform.localScale.z / factor);
            escalaEsfera = new Vector3(transform.localScale.x / factor, transform.localScale.y / factor, transform.localScale.z /
factor);
        }
        transform.localScale = escala;
        esfera.transform.localScale = escalaEsfera;
    }

    public IEnumerator DesactivarBoost()
    {
        yield return new WaitForSeconds(2f);
        boosted = false;
    }

    public IEnumerator DesactivarResbalado()
    {
        yield return new WaitForSeconds(1f);
        resbalado = false;
    }

    public IEnumerator DesactivarInvencible(int tiempo)
    {
        yield return new WaitForSeconds(tiempo);
        invencible = false;
    }

    public IEnumerator DesactivarRalentizado(int tiempo)
    {
        yield return new WaitForSeconds(tiempo);
        ralentizado = false;
    }

    public IEnumerator DesactivarStuneado(int tiempo) {
        yield return new WaitForSeconds(tiempo);
        stuneado = false;
    }

    #endregion
```

```
    #region ONLINE
    public void CargarDatos() {

        spawn = GameObject.Find("Spawner").GetComponent<Spawn>();
        GameManager.Instancia.AddCoche(this);
        rank = GameObject.Find("RANKING");
        puntosControl = GameObject.Find("PuntosControl");

        vista = GetComponent<PhotonView>();
        if (vista.IsMine) {
            barraVida.transform.parent.gameObject.SetActive(true);
            habilidades.SetActive(true);
            textoVida.gameObject.SetActive(true);
            contadorVueltas.gameObject.SetActive(true);
        }
        propiedadesJugador = PhotonNetwork.LocalPlayer.CustomProperties;
        propiedadesJugador["jugadorNickName"] = PhotonNetwork.LocalPlayer.NickName;
        propiedadesJugador["jugadorVuelta"] = 0;
        propiedadesJugador["jugadorDistancia"] = Vector3.Distance(transform.position,
puntosControl.transform.GetChild(numPuntoControl).transform.position);
        propiedadesJugador["jugadorPuntoControl"] = numPuntoControl;
        propiedadesJugador["jugadorPosicion"] = 0;
        propiedadesJugador["jugadorPFinal"] = null;
        propiedadesJugador["jugadorPuntos"] = 0;
        camaras = GameObject.Find("Camaras");
        carModel = transform.GetChild(0).transform.GetChild(4).gameObject;
        //CARGAMOS LAS PARTICULAS Y COLORES
        particulas[0] = particulasParent.transform.GetChild(0).transform.GetChild(0).GetComponent<ParticleSystem>();
        particulas[1] = particulasParent.transform.GetChild(0).transform.GetChild(1).GetComponent<ParticleSystem>();
        particulas[2] = particulasParent.transform.GetChild(1).transform.GetChild(0).GetComponent<ParticleSystem>();
        particulas[3] = particulasParent.transform.GetChild(1).transform.GetChild(1).GetComponent<ParticleSystem>();
        colores[0] = Color.yellow;
        colores[1] = Color.cyan;
        colores[2] = Color.magenta;
        //PhotonNetwork.LocalPlayer.CustomProperties = propiedadesJugador;
        PhotonNetwork.SetPlayerCustomProperties(propiedadesJugador);
        ActualizarHP(maxHP);
        ActualizarRanking();
    }

    public void IniciarTemporizador() {
        tiempoCircuito = Time.deltaTime;
    }

    public void ActualizarTemporizador() {
        tiempoCircuito += Time.deltaTime;
    }

    public void GuardarTiempo() {
        //CARGAR TIEMPOS ACTUALES
        string nombre = "";
        switch (SceneManager.GetActiveScene().name)
        {
            case "Ovalo":
                nombre = "Leaderboard Ovalo";
                break;
            case "Karting":
                nombre = "Leaderboard Karting";
                break;
            case "Castillo":
                nombre = "Leaderboard Castillo";
                break;
        }
        var request = new UpdatePlayerStatisticsRequest
        {
```

```
            Statistics = new List<StatisticUpdate> {
                new StatisticUpdate {
                    StatisticName = nombre,
                    Value = (int)tiempoCircuito-5
                }
            }
        };
        PlayFabClientAPI.UpdatePlayerStatistics(request, null, null);
        //PlayFabClientAPI.GetUserData(new GetUserDataRequest(),TiemposRecibidos,null);
    }

    public void TiemposErroneos(PlayFabError error) {
        Debug.Log("ERROR: " + error.ToString());
    }
    #endregion

    #region INFORMACION JUGADOR
    public void CargarPuntosControl() {
        TrackCheckpoints tracker = GameObject.Find("Tracking").GetComponent<TrackCheckpoints>();
        tracker.AddCoche(this);
        tracker.AddCocheTransform(esfera.transform);
    }
    #endregion



    public void SetNickJugador(string nick) {
        nickJugador = nick;
    }

    public IEnumerator ActualizarDistanciaPuntoControl() {
        while (true)
        {
            propiedadesJugador = PhotonNetwork.LocalPlayer.CustomProperties;
            distanciaSiguientePunto = Vector3.Distance(transform.position,
puntosControl.transform.GetChild(numPuntoControl).transform.position);
            propiedadesJugador["jugadorDistancia"] = distanciaSiguientePunto;
            //PhotonNetwork.LocalPlayer.CustomProperties = propiedadesJugador;
            PhotonNetwork.SetPlayerCustomProperties(propiedadesJugador);
            yield return new WaitForSeconds(2);
            ActualizarRanking();
        }
    }

    public void ActualizarHP(int cantidad) {
        if (vista.IsMine)
        {
            if (invencible)
                return;
            hp += cantidad;
            if (hp > maxHP)
            {
                hp = maxHP;
            }
            else if (hp <= 0)
            {
                hp = 0;
                Respawn();
            }
            textoVida.text = "" + hp;
            barraVida.fillAmount = (float)hp / (float)maxHP;
        }
    }

    public void Respawn() {
        ActualizarHP(maxHP);
```

```
        esfera.position = puntosControl.transform.GetChild(numPuntoControl).transform.position+new Vector3(0,1,0);
    }
}
```