DOCUMENTO DE JUSTIFICACIÓN – RÚBRICA RODEO PANDA

Joc endless relativament equilibrat

Velocidad progresiva de 10 a 30 con sistema de 3 carriles. Dificultad creciente sin frustrar al jugador.

Lògica adaptada a format “endless”
Pista generada proceduralmente a partir de 4 tracks preconfigurados. GameOver solo por colisión. No hay fin definido.

Lògica de joc arcade funcional
Controles por swipes, respuesta inmediata, power-ups (escudo, imán, velocidad), obstáculos variados y puntuación por monedas + metros.

Interfície amb un cert treball gràfic (prohibit assets default de Unity)
UI totalmente personalizada: fuentes custom (Ubuntu-Title SDF), iconos propios, menús únicos (inicio, tienda, settings, GameOver). Ningún asset por defecto de Unity usado.

Interfície adaptable a tamany de la pantalla funcionals
Canvas Scaler ajusta UI automáticamente a cualquier resolución. Elementos con anchors y script AdaptiveUIElement evitan desbordamientos.

Interfície funcional i inputs d’usuari funcionals
Todos los botones funcionan: Play, Settings, Shop, Quit, Retry. Swipes detectados correctamente. Sliders de volumen y compras en tienda operativos.

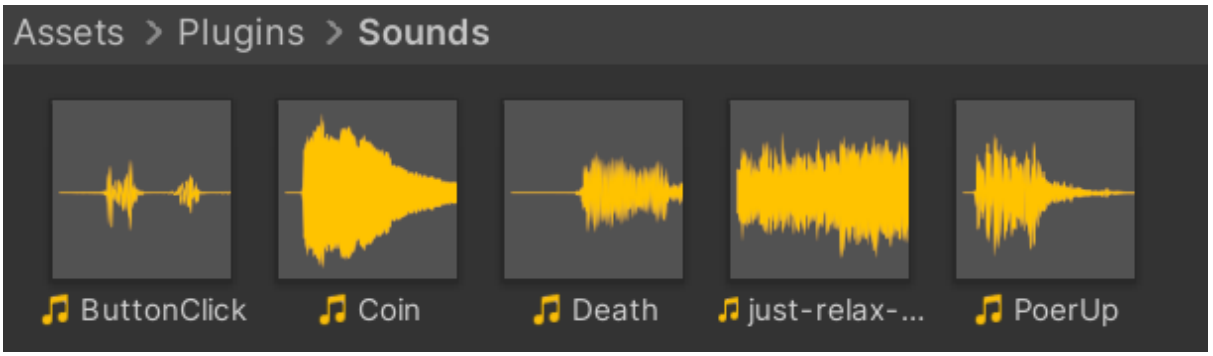
Ús de dotween en llocs localitzats

DOTween usado para: cambios suaves entre carriles, efecto de shake al morir (DOShakeScale). Código visible y demostrado en video/captura.

```
2 references
void StartBackflipLoop()
{
    // Matar cualquier animación anterior
    backflipSequence?.Kill();

    // Crear secuencia de animación
    backflipSequence = DOTween.Sequence();

    // 1. AGACHARSE (encogerse)
    backflipSequence.Append(
        transform.DOScale(originalScale * crouchScale, crouchDuration)
            .SetEase(crouchEase)
    );

    // 2. IMPULSO + BACKFLIP (todo junto)
    backflipSequence.Append(
        transform.DOJump(originalPosition, jumpHeight, 1, jumpDuration)
            .SetEase(jumpEase)
    );

    // Rotación del backflip (al mismo tiempo que el salto)
    backflipSequence.Join(
        transform.DORotate(
            new Vector3(backflipRotation, originalRotation.eulerAngles.y, originalRotation.eul
            jumpDuration,
            RotateMode.FastBeyond360
        ).SetEase(Ease.InOutQuad)
    );

    // Volver a escala normal mientras cae
    backflipSequence.Join(
        transform.DOScale(originalScale, jumpDuration * 0.5f)
```

Events sonors a totes les interaccions

Sonidos en: recoger monedas, activar power-ups, colisionar, morir, pulsar botones. Volumen ajustable por separado.



Efectes de partícules a totes les interaccions

Partículas activadas en: correr, saltar, deslizarse y morir. Efectos visuales mejoran feedback.

Joc adaptable a mode portrait i landscape

Funciona en ambas orientaciones gracias a SwipeDetector.cs y Canvas Scaler. Rotación habilitada en Player Settings.

Ús visible de sensors (2 sensors)

1. Vibración en 500m (Handheld.Vibrate()).
2. Flash de cámara al morir (API Android).

```
if (totalMeters >= 500f && !has500MetersVibrated)
{
    has500MetersVibrated = true;
    if (vibrationCoroutine != null) StopCoroutine(vibrationCoroutine);
    vibrationCoroutine = StartCoroutine(VibrateFor500Meters());
}

// Método del sensor
private System.Collections.IEnumerator VibrateFor500Meters()
{
    Debug.Log("[PlayerController] ¡500 metros alcanzados! Vibrando durante 3 segundos");

    float elapsedTime = 0f;
    while (elapsedTime < 3f)
    {
        #if UNITY_ANDROID || UNITY_IOS
        Handheld.Vibrate();
        #endif

        yield return new WaitForSeconds(0.5f);
        elapsedTime += 0.5f;
    }

    Debug.Log("[PlayerController] Vibración de 500 metros completada");
}
```

```
isDead = true;
StartCoroutine(FlashOnDeath());

// Método del sensor
private System.Collections.IEnumerator FlashOnDeath()
{
    Debug.Log("[PlayerController] Activando flash de cámara por muerte");

    #if UNITY_ANDROID
    AndroidJavaClass cameraClass = new AndroidJavaClass("android.hardware.Camera");
    AndroidJavaObject camera = null;
    AndroidJavaObject parameters = null;
    bool flashActivated = false;

    try
    {
        camera = cameraClass.CallStatic<AndroidJavaObject>("open");
        parameters = camera.Call<AndroidJavaObject>("getParameters");

        parameters.Call("setFlashMode", "torch");
        camera.Call("setParameters", parameters);
        camera.Call("startPreview");

        flashActivated = true;
    }
    catch (System.Exception e)
    {
        Debug.LogWarning($"No se pudo activar el flash: {e.Message}");
```

Treball general de cohesió del projecte (prohibit assets genèrics)
Assets 100% personalizados. Arquitectura limpia (Singleton, Event System), persistencia de datos, carpetas organizadas, código comentado. Sin assets default visibles.

fin