MP0492.

Proxecto de desenrolo de aplicacións multiplataforma.

Autor: Xan Cortés Arnaiz

Título: Raccoon Quest

Fecha: 25/08/2025

Sumario

1.Motivación/Xustificación do proxecto

Inclúese o nome, a **motivación**/xutificación do proxecto e a imaxe corporativa (logotipo do proxecto, cores a empregar, etc).

Desarrollar un videojuego me parecía más motivador que hacer otro tipo de aplicación, en parte porque los videojuegos son uno de mis principales pasatiempos, no solo jugarlos si no aprender de diseño de niveles mecánicas etc y por otro lado porque ya tengo algo de experiencia haciendo aplicaciones web mientras que los conocimientos que tengo sobre unity son muy limitados, por eso el reto de intentar hacer este proyecto resultaba mas interesante.

Logo:

2.Obxectivos

Desarrollar una primera versión funcional de un videojuego de plataformas y puzzles, con niveles sencillos y rápidos que pueda ser escalado a un proyecto más ambicioso en un futuro.

Objetivos especificos:

Diseñar y programar mecánicas básicas pero agradables de control de movimiento y interacción con objetos del entorno.

3.Estudo de mercado

Estado actual del mercado:

La industria del videojuego a nivel global está en un estado inestable, los estudios más grandes están en una fase de recesión debido a las expectativas irreales de crecimiento que tuvieron tras la crisis del COVID y al no alcanzar estás está habiendo una serie de despidos masivos a la vez que una subida de precio de los videojuegos triple A, alcanzando algunos los 90$, esto ha causado un cambio en el comportamiento de los consumidores, muchos jugadores, con salarios más estancados y reducidos y con menor tiempo para dedicar a los juegos mientras los grandes títulos siguen aumentando en duración, optan por juegos más pequeños, cortos, accesibles o incluso gratuitos, lo que ha creado un nicho de mercado perfecto para juegos indie similares a este.

Competidores principales:

Hablar de competidores en este proyecto es complicado ya que la mayoría de juegos conocidos con publico o mecánicas similares al mio son relativamente antiguos, aunque siguen siendo populares, juegos como Portal que, salvando las distancias, comparte estructura con el mío, o hollow knight y celeste que comparten público objetivo salieron hace años por lo que, a pesar de que siguen disponibles y son un competidor la mayoría de público que los haya querido jugar ya los habrá jugado, pero no por ello se pueden ignorar.

Por otro lado competidores más actuales podrían ser:

CATO: buttered cat: Juego de plataformeo con una estética y mecánicas similares.

Hollow Knight: Silksong: Exploración y puzzles en un mundo interconectado.

Valor diferenciador del producto:

Precio reducido: El juego será accesible para un público amplio aprovechando el nicho de mercado de juegos indie frente a título AAA de alto coste

Niveles rápidos y satisfactorios: las partidas ofrecen recompensas inmediatas permitiendo que el jugador disfrute de una experiencia completa sin necesidad de largas sesiones.

Jugabilidad simple: controles intuitivos y mecánicas fáciles de aprender.

Mapaches: a todo el mundo le gustan los mapaches

Oportunidades y amenazas:

Oportunidades:

Crecimiento del mercado inide, con plataformas como steam, gog o itch.io ofreciendo visibilidad.

Demanda de experiencias de juego únicas y económicas, especialmente frente al aumento de precios de la industria

Amenazas:

Alta competencia indie.

Limitación de presupuesto que puede afectar a la calidad y sobretodo al marketing,

4.Orzamento do proxecto

Duración del proyecto: 26 horas.

Coste de desarrollo:

26horas x 10€/hora = 260€

Gastos de software:

Unity: Motor de desarrollo Versión gratuita 0€

Libresprite: Aplicación que he usado para todos los recursos gráficos del juego, versión gratuita 0€

Total: 260€

5.Documentación técnica

Introducción

Esta es la documentación técnica del proyecto Raccoon Quest, hablaremos de la estructura de carpetas y assets, las funciones y métodos de las clases, los diagramas de casos de uso y de clases y algunos errores conocidos y marcados para resolver en un futuro

Estructura del proyecto

```
Assets/
├── MyAssets/
│   ├── Animation/ Las animaciones y los controladores de las mismas
│   ├── Character/ Los sprites relacionados con el personaje
│   ├── Materials/ Los materiales usados para los distintos elementos
│   ├── Prefabs/ Las plantillas para los objetos complejos
│   ├── Signs/ Sprites para los carteles
│   ├── Timeline/ Los controladores de las cinemáticas del juego
│   └── Titles/
│       ├── Imagenes/ PNGs usados para los fondos y decoraciones
│       └── Titlemaps/ Plantillas creadas a partir de los sprites que se
            usan en los tilemaps
├── Scenes/ Los niveles del juego
├── Scripts/ Archivos de códgio
├── Settings/
└── TextMesh Pro/
```

La carpeta de scripts debería tener mas subdivisiones, carpetas especificas para jugador, enemigos, plataformas...pero como el proyecto aún es pequeño y hay pocos scripts tener que ir moviendose entre las carpetas era más contraproducente, a futuro habría que dividirla

Diagrama de clases

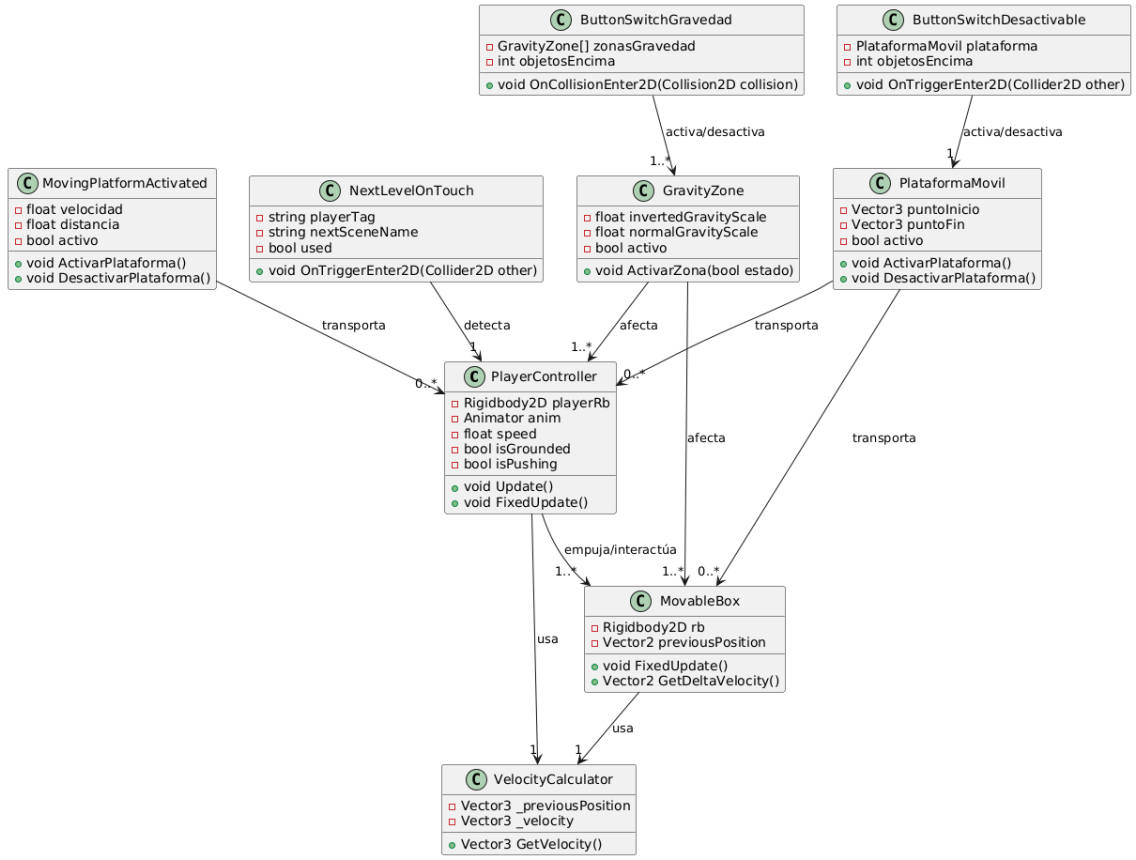
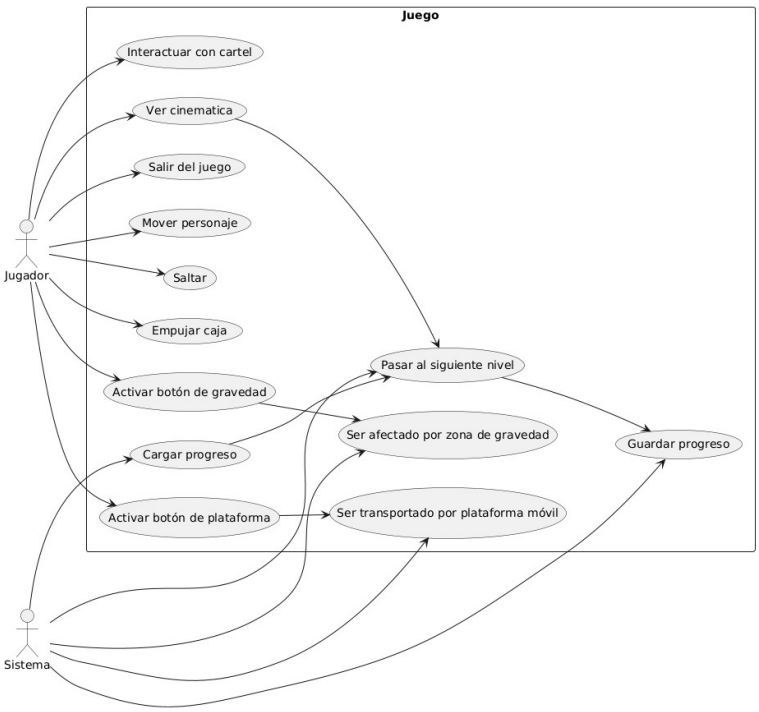


Diagrama de casos de uso



Clases principales

- ### BasuraCinemática
  - **OnTriggerEnter2D (Collider2D other)**
    - **Verifica si el objeto que entra en el trigger tiene el tag "Player"**
    - **Si es así, desactiva el seguimiento de cámara (cameraFollow.enabled = false) y reproduce la cinemática con director.Play()**
    - **Se suscribe al evento stopped del PlayableDirector para ejecutar OnCinematicaTerminada**
  - **OnCinematicaTerminada(PlayableDirector obj)**
    - **Reactiva el seguimiento de cámara**
    - **Carga la escena especificada en nivelSiguiente usando SceneManager.LoadScene**

- ### ButtonSwitchDesactivable
  - **OnTriggerEnter2D (Collider2D other)**
    - **Si el objeto tiene el tag "Player" o "Caja", incrementa objetosEncima**
    - **Llama a ActualizarEstado() para verificar si se debe activar la plataforma**
  - **OnTriggerExit2D (Collider2D other)**
    - **Si el objeto tiene el tag "Player" o "Caja", decrementa objetosEncima**
    - **Vuelve a llamar a ActualizarEstado()**
  - **ActualizarEstado()**
    - **Compara objetosEncima con objetosNecesarios**
    - **Si se cumple la condición, activa la plataforma (plataforma.SerActive(true)) y cambia el sprite del botón**
    - **Si no, desactiva la plataforma y restaura el sprite original**

- **ButtonSwitchGravedad**
  - **OnCollisionEnter2D(Collision2D collision)**
    - **Si el objeto tiene el tag "Player" o "Caja", incrementa objetosEncima**
    - **Llama a ActivarBoton() si hay suficientes objetos**
  - **ActivarBoton()**
    - **Itera sobre zonasGravedad**
    - **Si la zona es temporal (zona.temporal == true), llama a ActivarZona() y luego DesactivarTrasTiempo()**
    - **Si es permanente, solo llama ActivarZona()**
  - **OnCollisionExit2D(Collision2D collision)**
    - **Decrementa objetosEncima si el objeto se retira**

- **CameraFollow**
  - **LateUpdate()**
    - **Calcula la posición deseada del jugador**
    - **Si el jugador se sale de la zona muerta (deadZone), interpola la posición de la cámara con Vector3.SmoothDamp**
  - **OnDrawGizmos()**
    - **Dibuja un rectángulo en el editor para visualizar la zona muerta**

- **GravityZone**
  - **ActivarZona()**
    - **Cambia el estado de la zona a activa**
    - **Modifica la gravedad del jugador (playerRb.gravityScale = gravedadInvertida)**
    - **Cambia el color del sprite para indicar activación**
  - **DesactivarTrasTiempo()**
    - **Usa Invoke() para llamar a DesactivarZona() tras tiempoActivo segundos**
  - **OnTriggerStay2D(Collider2D other)**
    - **Si la zona está activa y el objeto es el jugador, aplica la gravedad invertida**
  - **OnTriggerExit2D(Collider2D other)**
    - **Restaura la gravedad original si el jugador sale de la zona**

- **PlayerController**
  - **Update()**
    - **Captura entrada horizontal (Input.GetAxisRaw("Horizontal")) y salto (Input.GetButtonDown("Jump"))**
    - **Cambia animaciones según el estado del jugador (corriendo, saltando, cayendo)**
  - **FixedUpdate()**
    - **Aplica movimiento horizontal con playerRb.velocity**
    - **Si se detecta salto y el jugador está en el suelo, aplica fuerza vertical**
  - **OnCollisionStayr2D() / OnCollisionExit2D()**
    - **Detecta si el jugador está en contacto con el suelo para permitir saltar**

- **MovableBox**
  - **FixedUpdate()**
    - **Calcula la velocidad heredada de la caja (inheritedVelocity) como la diferencia entre la posición actual y la anterior**
    - **Guarda la posición actual para el siguiente frame**
  - **OnCollisionStay2D(Collision2D collision)**
    - **Si el objeto encima tiene Rigidbody2D, aplica la velocidad heredada con ApplyDelta()**
  - **ApplyDelta(Rigidbody2D rb)**
    - **Suma la velocidad heredada a la velocidad del objeto encima**

- **NextLevelOnTouch**
  - **OnTriggerEnter2D(Collider2D other)**
    - **Si el objeto tiene el tag "Player" y used == false, llama a Invoke("LoadNext", delay)**
  - **LoadNext()**
    - **Carga la escena especificada en nextSceneName**

Errores comunes:

Problemas con las físicas de las cajas, por la transmisión de movimiento a veces empujan al jugador demasiado en situaciones en las que no tendrían que hacerlo

Problemas con la colisión del jugador, la hitbox del mapache no ocupa todo el sprite, la cola queda sin colisión, esto causa que en situaciones concretas al girarse el jugador pueda colarse entre algunas partes del mapa

6.Librerías

Librerias:

UnityEngine

UnityEngine.SeneceManagement

UnityEngine.Playables

Unityengine.UI

System.Collections

Herramientas:

UnityHub

LibreSprite: Aplicación de dibujo para pixelArt

7.Probas e validacións

Pruebas realizadas mediante el editor de unity, añadidos debugs en las acciones para poder observar comportamiento y Gizmos para ayudas visuales durante el desarrollo.

8.Problemas atopados e solucións

Problemas con las físicas, tanto del personaje como de los objetos empujables, principalmente para conseguir que al quedarse encima de una plataforma móvil el movimiento se les transmitiese de forma correcta.

Finalmente se consiguió gracias a la clase auxiliar VelocityCalculator.

Problemas con la colisión del personaje principal contra el fondo, el fondo tilemap al estar hecho mediante “bloques” cada uno tenia su propia colisión y el jugador se quedaba atascado entre ellos, añadiendo un tilemap collider y un collision collider los unificó pero el personaje se seguía quedando clavado si saltaba directamente contra el, añadiendole al personaje un material que no tuviese rozamiento arreglaba el problema y se deslizaba correctamente.

Al deslizarse por una superficie hasta el suelo no detectaba una nueva colisión lo que hacia que se desactivase el salto, se arregló añadiendo un objeto hijo para el jugador que servia para detectar colisiones de forma constante debajo de él.

Cambios en las animaciones, conseguir que estas se activasen en el momento correcto de forma fluida.

9.Conclusións

Ha sido una buena experiencia a nivel técnico, trabajar con Unity me ha permitido adentrarme en un ámbito de la programación que había tocado poco y resulta interesante ademas de permitirme también realizar algunas tareas de edición y diseño.

Salirme de los programas y los lenguajes a los que estoy acostumbrado me ha ayudado a aprender nuevas formas de estructurar y ordenar el código y coordinar funciones asíncronas como las animaciones.

Los principales impedimentos fueron sobretodo pelear con el nuevo lenguaje de programación ya que no había usado prácticamente C#.

10.Propostas de mellora/Ampliacións do proxecto

Seguir con el desarrollo y añadir más niveles

Añadir más mecánicas: Portales, power-apps, selector de niveles, varios espacios para guardar la partida, cosméticos para el mapache

11.Bibliografía/Webgrafía

Manual de unity

Vídeo para realizar cutscenes

Vídeo sobre como dejar de atascarse en las paredes

Vídeo sobre plataformas móviles

Post sobre velocityCalculator para plataformas móviles

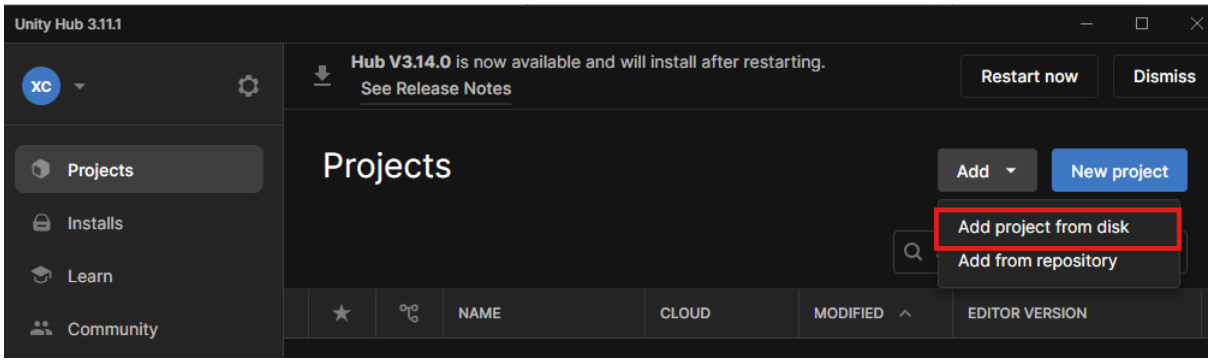
Vídeo sobre la gravedad invertida

Copilot para arreglos y dudas concretas

12.Manual técnico

Al ser un proyecto de unity que no es muy complejo no es difícil de desplegar.

Descargar el proyecto y descomprimirlo, ir al hub de unity y pulsar add y despues add from disk

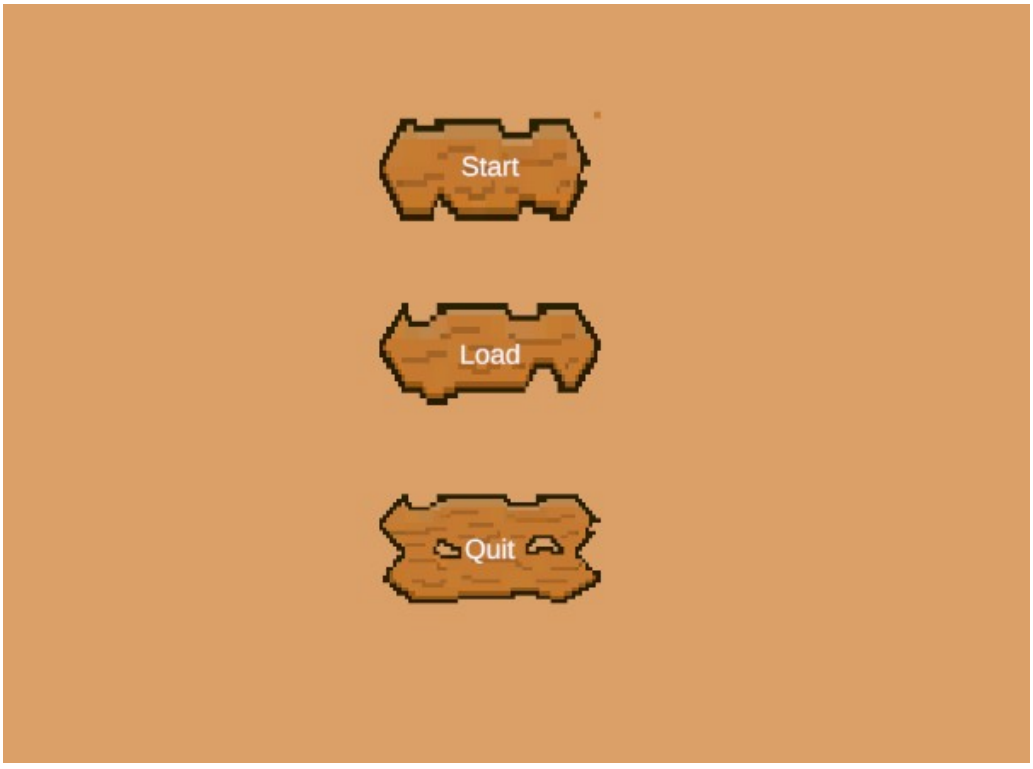


Buscar el archivo que acabas de descargar y añadirlo, aparecerá en la lista y podrás ejecutarlo

13.Manual de usuario

Ejecutar el RaccoonQuest.exe

En el menú principal pulsar Start para comenzar una nueva partida, load para cargar o quit para salir del juego



Puedes mover al mapache con las flechas de adelante y atrás o con los botones A y D

El mapache puede saltar pulsando espacio

El mapache puede empujar cosas andando contra ellas y los rayos de gravedad hacen que se pegue al techo

Puedes reiniciar el nivel en el que estás pulsando el botón R

Puedes leer carteles con la tecla E y salir de ellos con el botón Escape

Con el botón Esc abrirás el menú lo que te permite continuar con el juego o cerrarlo