UNIVERSITAT POLITÈCNICA DE VALÈNCIA

CAMPUS D'ALCOI

Desarrollo de un Videojuego enfocado a Realidad Virtual

Grado en Ingeniería Informática

Trabajo final de Grado

Alumno: **Axel Domínguez Aránega**

Tutor: **Dr. Jordi Joan Linares Pellicer**

Curso Académico: 2016/2017

Indice

1. Introducción

1.1. Motivación

Mi curiosidad de saber cómo funcionan las cosas, de saber que hay detrás de todo. Esto me atrajo al mundo de la programación, decidí que era algo a lo que me gustaría dedicarme en cuarto de la eso, cuando tuve que hacer una página en HTML, el ver que cambiando una etiqueta podía cambiar una página web, me llamó mucho la atención. Me decidí a estudiar informática.

Los videojuegos siempre han sido parte de mi vida, desde pequeño recuerdo invitar a mis amigos a jugar a mi casa, recuerdo que el tiempo volaba mientras jugábamos. Más adelante, cuando al fin tuve internet en casa, todavía me atrajeron más y más, hasta hoy, sigo jugando, sigo conociendo a gente y sigo disfrutando de la experiencia.

Gracias a una asignatura optativa he podido acercarme un poco más al mundo de los videojuegos, existía la posibilidad de desarrollar un videojuego como trabajo final de grado así que no la he rechazado. El tema de realidad virtual, al nivel que está hoy, es algo nuevo, algo que llama la atención y con lo que poder disfrutar. Este es el motivo por el que he decidido hacer el videojuego en realidad virtual.

1.2. Objetivos

El objetivo de este proyecto es realizar un videojuego funcional en realidad virtual, será un juego que mezclará elementos en 2D y 3D.

El jugador controlará al personaje principal del juego y tendrá que avanzar a lo largo de los niveles completando una serie de desafíos. El jugador tendrá que interactuar con los diferentes elementos del entorno para avanzar al siguiente nivel.

2. Realidad Virtual

La realidad virtual no es algo nuevo, en 1962, Morton Heilig construyó el prototipo de Sensorama, una especie de televisor que mostraba películas estereoscópicas en 3D, sonido estéreo y se podían activar olores mientras se reproducía la película.

Sensorama

Más adelante (1961) Phillco Corp. desarrolló un proyecto para utilizarse en entrenamientos militares. Era un casco que incorporaba una pantalla y tenía un control de posición de la cabeza.


Ultimate Display

En 1965 un equipo del MIT desarrolló un dispositivo (llamado Ultimate Display) que consistía en un casco acoplado a un ordenador, debido a la tecnología de la época el casco y los ordenadore eran tan grandes y pesados que el casco tenía que estar colgado del techo.

En 1982, la empresa de Jaron Lanier (VPL Research), desarrolló Data Glove, unos guantes con sensores capaces de reconocer el movimiento y la posición de los dedos. Como dato curioso, estos guantes consistían en 6502 microcontroladores. Más adelante también desarrollaron un traje completo, llamado DataSuit, que medían el movimiento de los brazos, piernas y del torso, también desarollaron EyePhone, unas gafas de realidad virtual que podía detectar el movimiento de la cabeza.

EyePhone
DataGlove
Data Suit

A principios de los 90 aparecían los primeros videojuegos que implementaban la realidad virtual. Los primeros cascos de realidad virtual que aparecieron fueron Sega VR (1993) de Sega y Virtual Boy (1995) de Nintendo.

Sega VR

Virtual Boy

En 2010, Palmer Luckey desarrolló el primer prototipo de Oculus Rift. Consigue financiarlo gracias a una campaña en Kickstarter donde trataba de conseguir 250.000$. La campaña consiguió recaudar 2.5 millones de dólares. Más tarde Facebook compró todo el proyecto y la compañía Oculus por la cantidad de 2.000 millones de dólares.

Oculus Rift

2.1. ¿En qué consiste?

La realidad virtual consiste en un mundo virtual generado por ordenador en el que el usuario tiene la sensación de estar en el interior de este mundo y dependiendo del nivel de inmersión, el usuario puede interactuar con este mundo y los objetos del mismo en un grado u otro. El mundo virtual puede estar basado en entornos reales o no y para percibirlo (a día de hoy) se necesitan unas gafas de realidad virtual. El objetivo de esta tecnología es crear un mundo del que puedes formar parte e incluso ser el protagonista. Los usos son muy diversos, puedes estar visitando una casa que todavía no ha sido construida, pilotando un avión, montándote en una montaña rusa, etc.

2.2. Mecanismos que intervienen en la realidad virtual

- Gráficos 3D: gráficos tridimensionales que permiten una percepción real de lo que vemos a través de las gafas de realidad virtual.
- Técnicas estereoscopia: da a los gráficos 3D profundidad y realismo, es un efecto que se consigue con dos imágenes paralelas, estas imágenes se superponen y crean una sensación de profundidad.
- Simulación del comportamiento: los movimientos del personaje son improvisados y tienen múltiples variables, por lo que están en constante evolución.
- Facilidad a la hora de navegar: a la hora de controlar nuestro personaje disponemos de nuestra visión, que se fusiona con la aplicación, lo que hace que el desarrollarse por el mundo sea más intuitivo.
- Técnica de inmersión total: las gafas de realidad virtual nos aíslan del mundo real, con ello se consigue que la inmersión sea lo más completa posible. El sonido y la vista son los sentidos que más estímulos reciben.

2.3. Opciones para introducirse en el mundo de Realidad Virtual

FSe van a detallar las diferentes opciones a la hora de introducirse en el mundo de Realidad Virtual. Como veremos, hay diferentes modelos, diferentes precios, diferentes plataformas donde se ejecutan, más y menos inmersión, entre otras características.

2.3.1. Google Cardboard

Esta es la opción más sencilla y económica para introducirse en este mundo. Las características más llamativas de esta opción es que las gafas están hechas de cartón por lo que su precio es realmente económico. A partir de esta opción surgieron otra más cuidadas, hechas de plástico y otros elementos para hacerlas más cómodas, también de

un precio superior.

Google Cardboard

Para hacer uso de Google Cardboard, lo único que necesario es un smartphone que pueda ejecutar aplicaciones o reproducir videos de realidad virtual. También hay que tener en cuenta el tamaño del móvil, ya que suelen encajar casi todo tipo de tamaños, pero conviene comprobar si el móvil que se va a usar junto a Google Cardboard es del tamaño correcto.

Como toma de contacto con el mundo de la Realidad Virtual, este tipo de técnica está muy bien, pero la inmersión que tenemos no tiene ni punto de comparación con otras opciones más cuidadas. Dependemos de que nuestro smartphone para todo, como que tenga giroscopio (para controlar el movimiento de la cabeza), tendríamos que hacer uso de un mando u otro dispositivo para para interactuar con el smartphone, ya que estará en las gafas y no podremos interactuar con él usando la pantalla del smartphone.

2.3.2. Samsung Gear VR

Samsung Gear VR

Esta opción va un paso más allá que las Google Cardboard, el ángulo de visión es mayor, hay sensores en las gafas que nos ayudan a seguir el movimiento de la cabeza, en un lado de las gafas existen una serie de controles que nos permiten interactuar con los menús de las diferentes aplicaciones y juegos con mayor facilidad.

El problema principal es que para hacer uso de las Samsung Gear VR, nos hace falta un modelo de smartphone de Samsung (Galaxy Note 5, Galaxy S6/S6 Edge/S6 Edge+, Galaxy S7/S7 Edge, or Galaxy S8/S8+).

2.3.3. PlayStation VR

A partir de esta opción, ya no nos hace falta un smartphone para sumergirse en la realidad virtual.

PS4, Dualshock, PS Camera, PSVR y PS Move

Las PlayStation VR son de Sony, nos hará falta una PlayStation 4 y una cámara (PS Camera) para exprimir al máximo la experiencia de realidad virtual. El diseño de las gafas hacen posible trabajar conjuntamente con la cámara, lo que hace el seguimiento de nuestra cabeza mucho más preciso que cualquiera de las opciones anteriores.

Uno de los “problemas” es que necesitas estar conectado permanentemente con la consola, un cable relativamente largo saldrá de las gafas para ir a la consola. Otro de ellos es que el hardware de la PS4 es inferior al que podríamos ver en un PC. Una de las ventajas es que el coste total (gafas + camara + PS4) será inferior al de un ordenador junto a otras gafas. Otra ventaja es que Sony dispone de diferentes controles (PS Move y Dualshock) que pueden más atractiva esta opción a la hora de jugar.

2.3.4. Oculus Rift

Oculus Rift

Este proyecto nacido bajo el amparo de Kickstarter se convirtió desde su origen en el detonante de una tendencia que hemos tardado cinco años en ver cristalizar, pero el resultado ha sido el de contar no ya con la prometedora propuesta de los chicos de Oculus, sino con otras muchas que tratan de aportar su granito de arena en este mercado.

Son las gafas que dejaron claro hacia dónde deberían dirigirse aquellos que buscaran experiencias de la máxima calidad en este ámbito.

Cuenta con accesorios muy útiles como Oculus Rift Sensors, se encargan de hacer el seguimiento de las constelaciones de LED infrarrojos para convertir tus movimientos en realidad virtual. O los Oculus Touch Controllers que harán que podamos “tener” manos en el mundo virtual.

Uno de los problemas principales es que el coste de las gafas es algo elevado (sobre los 700€, incluyendo los Oculus Touch) y la inversión no solo afecta a las gafas, sino que también tenemos que tener un ordenador potente para sacarle provecho a la realidad virtual.

2.3.5. HTC Vive

HTC Vive

Unas gafas de realidad que surgían con el apoyo de Valve y su plataforma de distribución de juegos. De hecho aquí la apuesta es importante porque ya se han sacado de la manga SteamVR, una sección especial en la que no solo encontraremos juegos para disfrutar de experiencias nativas de realidad virtual, sino que también "adaptará" contenidos tradicionales -juegos de toda la vida- para hacer que podamos disfrutarlos con estas gafas de realidad virtual.

Que Valve y Steam estén detrás de este proyecto hace que estas gafas sean una muy buena opción a la hora de plantearse comprarlas. Pero esto no significa que el resto de proyectos no vayan a tener los mismos contenidos o más.

El problema es el mismo que en el caso de las Oculus Rift, la inversión no afecta solo a las gafas (sobre los 900€) sino que también afecta al ordenador, que tendrá que ser potente para sacar rendimiento a la realidad virtual.

2.4. Aplicaciones de la Realidad Virtual

Se van a nombrar las aplicaciones más interesantes de la realidad virtual, muchas de las aplicaciones ya están siendo desarrolladas pero en una fase temprana del desarrollo, aún les queda mucho camino por delante, pero el futuro para la realidad virtual es muy prometedor.

- Medicina: Desde apoyar a los cirujanos a realizar operaciones hasta tratar depresiones o fobias.
- Uso militar: Entrenar para operaciones o anticiparse a problemas que pudieran surgir en una operación.
- Arquitectura: Puedes diseñar tu propia casa o un arquitecto puede mostrarte su diseño, realizar un tour virtual por un edificio que todavía no ha sido construido.
- Arte: Podemos dibujar en 3D haciendo uso de las “manos virtuales”.
- Redes sociales: Reunirte con tus amigos que no están en tú casa, ciudad o país podrá ser posible gracias a la realidad virtual.
- Educación: Los alumnos no necesitarán imaginar cómo era la prehistoria o cómo eran los dinosaurios, gracias a la realidad virtual podrán desplazarse ahí, podrán recorrer el cuerpo humano como (un poco así como protagonistas de “Érase una vez: El cuerpo humano”).
- Información: Desde noticias donde podremos movernos en ese lugar, hasta pasearnos por un desfile de moda o vivir “en primera persona” un debate parlamentario.
- Deporte: Colocarte dentro del campo para ver un gol o seguir a tu jugador favorito por el campo como si fueses él o ella.
- Viajes: Visitar una ciudad que ya no existe o viajar a otro planeta sin salir de tu casa.

3. Herramientas utilizadas

3.1. Hardware

El desarrollo del videojuego se ha implementado desde el día uno en un móvil de la marca Xiaomi, modelo Redmi Note 4. Es un móvil de gama media, la característica más importante es que dispone de giroscopio, necesario para poder hacer uso de la realidad virtual. Tiene una resolución 1080p, algo que no es lo ideal para la realidad virtual, ya que se verá todo algo pixelado.

Como headset, elemento que se usa para poner el móvil dentro y poder disfrutar de la realidad virtual, se ha usado el Xiaomi VR Headset, económico, ligero y en el que se puede usar prácticamente cualquier móvil.

Se ha utilizado también un mando para que el juego fuese más divertido. El modelo del mando usado ha sido MOCUTE Mini Bluetooth V3.0, un mando pequeñito, que se comunica con el móvil por bluetooth y que es recargable. Tiene las suficientes teclas para poder sacar el máximo rendimiento al juego.

3.2. Software

3.2.1. Unity 3D

Es una herramienta que nos ayuda a desarrollar videojuegos para diversas plataformas mediante un editor y scripting para crear videojuegos con un acabado profesional. Esta herramienta está accesible al público en diferentes versiones, gratuita y profesional, cada cual con sus ventajas y limitaciones, evidentemente la más completa es la profesional pero es necesario hacer un desembolso que no todo el mundo puede permitirse. Usaremos la versión gratuita para desarrollar este videojuego.

Unity 3D nos provee de un editor visual muy útil y completo donde mediante unos pocos clicks podremos importar nuestros modelos 3D, texturas, sonidos, etc. para después ir trabajando con ellos.

Los lenguajes de programación soportados por Unity3D son Javascript, C# y Boo (una extensión de Python). El videojuego se creará usando C#.

Unity 3D nos permite exportar nuestro juego a diversas plataformas como Web, PC (Windows, Linux, OS X), dispositivos móviles (Android, iOS, Windows Phone, Tizen), Smart TV, consolas y dispositivos de realidad virtual, sin necesidad de desarrollar el juego desde cero para cada plataforma.

3.2.2. Visual Studio

Visual Studio es un IDE (siglas en inglés de entorno de desarrollo interactivo) que soporta diferentes lenguajes de programación como C++, C#, Visual Basic .NET, F#, Java, Python, Ruby y PHP.

Lo usamos para crear los scripts necesarios para el juego.

3.2.3. Blender

Blender es un programa informático multiplataforma, dedicado especialmente al modelado, iluminación, renderizado, animación y creación de gráficos tridimensionales.

Lo usaremos para crear los modelos básicos del juego.

4. Diseño y desarrollo del proyecto

4.1. Estructura del proyecto

- Animations: donde están las animaciones del juego.
    - Enemy: animaciones del NPC enemigo.
    - Player: animaciones del jugador.
- Resources: donde se guardan todos los recursos del juego.
    - Materials: materiales creados para que el juego sea más vistoso.
    - Prefabs: objetos ya creados y listos para su reutilización.
    - Sounds: sonidos del juego.
    - UI: donde se guarda todo lo usado para crear la interfaz del juego.
        - Fonts: fuentes de la interfaz.
        - Prefabs: objetos ya creados y listos para su reutilización de la interfaz.
        - Sprites: imágenes que se usan para dar forma a la interfaz.
- Scenes: donde se guardan los niveles y los menús del juego.
- Scripts: donde se guarda el código del juego.
    - Camera: scripts destinados al control de la cámara.
    - Controllers: scripts de movimiento, tanto del jugador como del enemigo.
    - Level: scripts para manejar el juego.
    - Objects: scripts que se encargan de los objetos.
    - UI: scripts destinados a la interfaz.
- Sprites: donde se guardan las imágenes usadas que no tengan que ver con la interfaz.

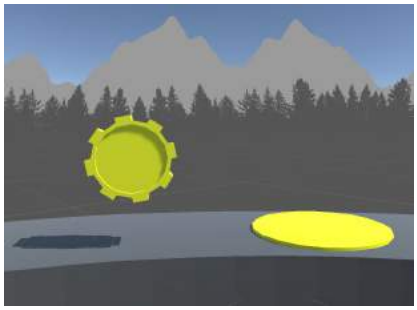
4.2. Mecánica del juego

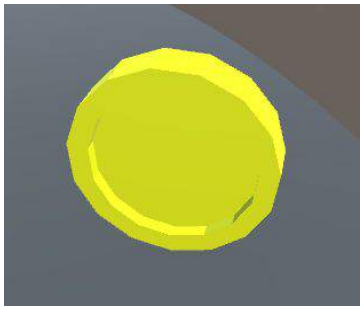
- Es un juego de un solo jugador.
- La partida no finaliza hasta que el jugador no completa el nivel o pulsa la tecla designada para salir al menú.
- El jugador se podrá desplazar lateral y verticalmente.
- Para completar la partida tendrá que cumplir los diferentes objetivos.
- Los objetivos son: recoger las monedas que encontrará esparcidas por el nivel y tendrá que depositar sobre la plataforma correspondiente los engranajes del mismo color.
- En los distintos niveles, podrá existir algún enemigo, si el enemigo toca al jugador, el nivel se reiniciará.
- El objetivo del juego es completar el nivel en el mínimo tiempo posible.
- Si el jugador cae de la plataforma, volverá al punto donde apareció al principio del nivel.
- Si un objeto cae de la plataforma, volverá al punto donde apareció al principio del nivel.
- Habrán plataformas que serán necesarias activar en un determinado orden para conseguir avanzar por el nivel desbloqueando alguna barrera que encontrarás por el camino.

4.3. Objetivos del juego

Los objetivos del juego son muy sencillos, se trata de coger todas las monedas que encontremos en el nivel y de colocar todos los engranajes sobre las plataformas del mismo color. En los niveles podremos encontrar diferentes obstáculos, como caídas al vacío o enemigos.

La imagen de la izquierda es un ejemplo de engranaje y de la plataforma dónde tenemos que colocarlo. La imagen de la derecha es una de las monedas que tendremos que recoger.



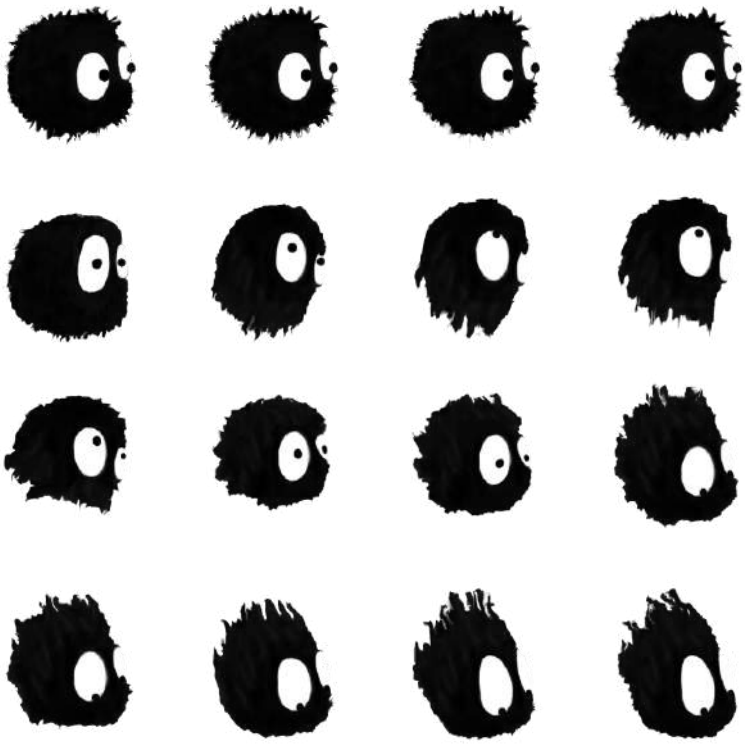


4.4. Personajes y animaciones

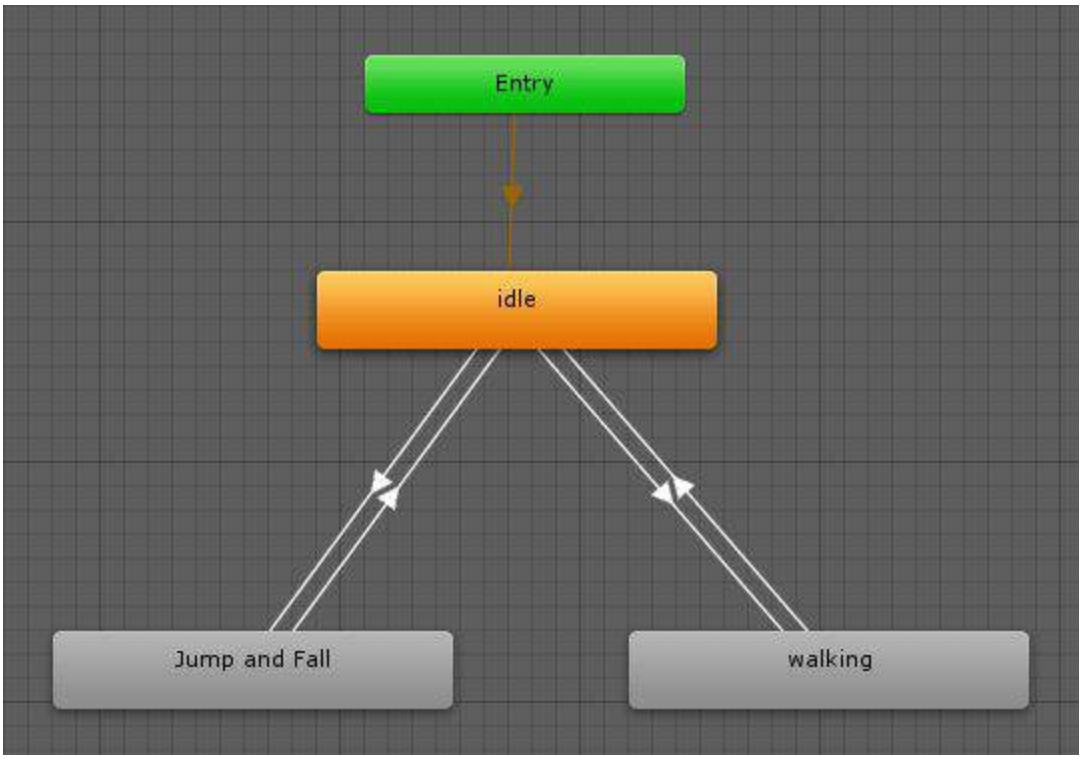
4.4.1. Jugador

Este es el jugador, es un personaje en 2D, las diferentes animaciones del jugador se controlan mediante el uso de variables (si está o no tocando el suelo, la velocidad vertical y la velocidad en horizontal) dependiendo de estas variables, la animación será una u otra.

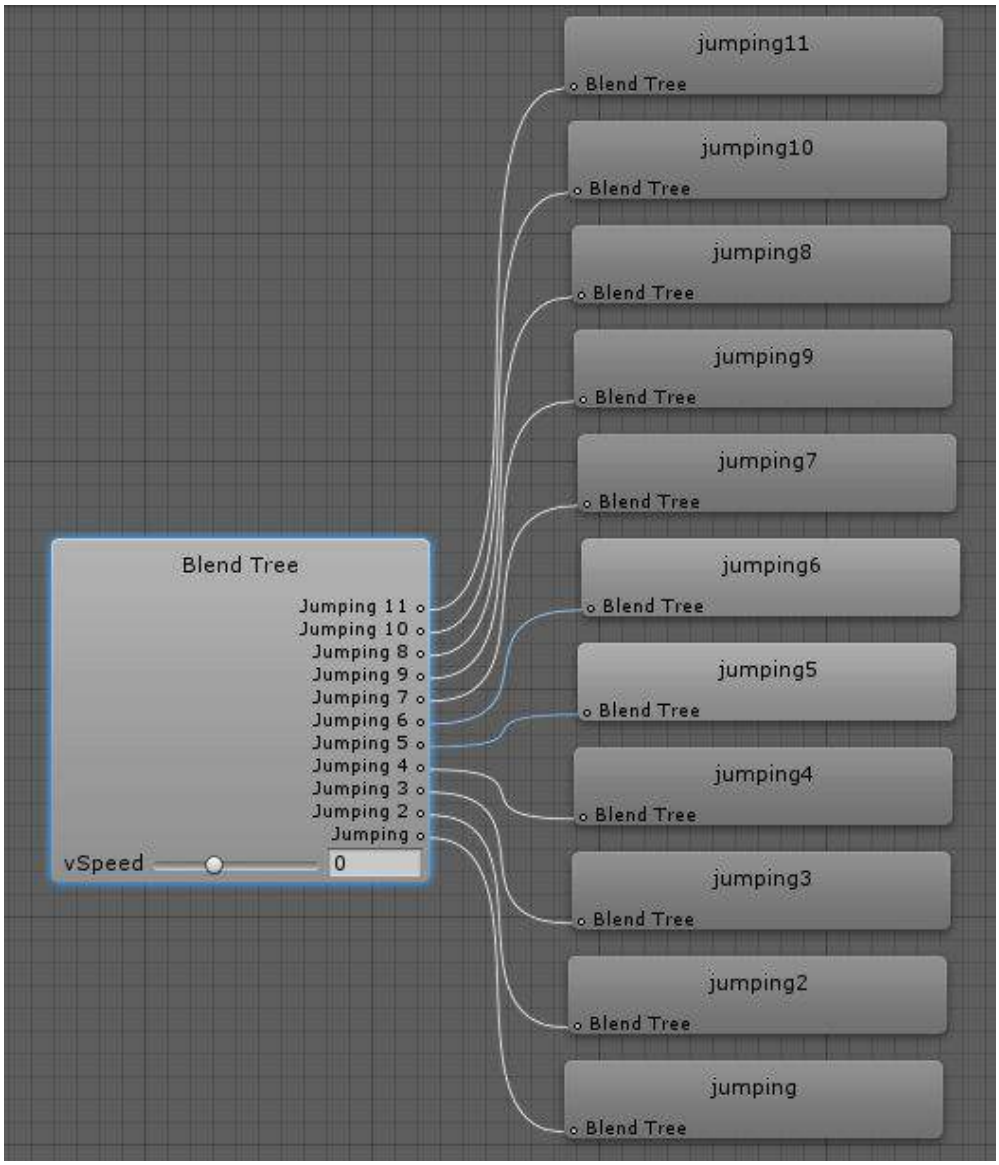
El jugador se controla haciendo uso del mando, puedes saltar y moverte hacia los lados.



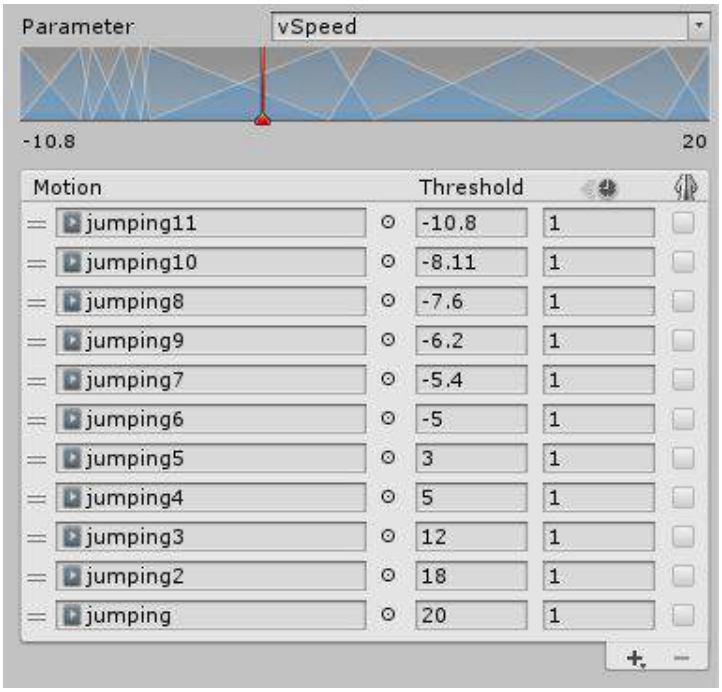
Personaje con las diferentes imágenes (o sprites) que se usan para hacer las animaciones.



Estas son las diferentes animaciones, cuando el personaje aparece en el nivel (*Entry*) pasa directamente al estado *idle* (en reposo o parado), dependiendo del jugador, podrá pasar al estado *walking* (andando) o al estado *Jump and Fall* (saltar y caer). De los estados *Jump and Fall* y *walking* volverá al estado *idle* (si el jugador no salta y deja de moverse).



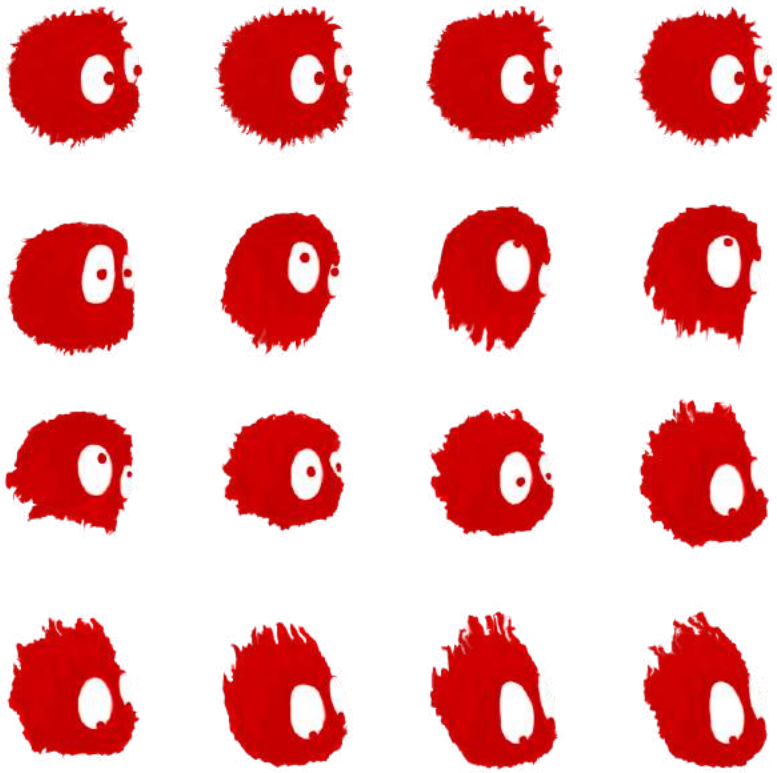
Como podemos apreciar, hay diferentes etapas de salto, dependiendo de la velocidad vertical (*vSpeed*) la animación (la imagen o sprite) será diferente.



En esta imagen, podemos ver los diferentes valores de la velocidad vertical (*vSpeed*) siendo positivo la velocidad hacia arriba (fuerza de salto) y negativo la velocidad hacia abajo (de caída)

4.4.2. Enemigo

Actualmente solo hay un enemigo, es igual que el jugador, usa las mismas animaciones, la única diferencia es que el enemigo se mueve solo, de un lado a otro, si encuentra un obstáculo dará la vuelta.



4.5. Efectos de sonido y música

En la versión inicial no existen muchos sonidos.

Está la música del juego, que se empieza a reproducir cuando iniciamos el juego y se escucha en bucle hasta que cerramos la aplicación.

Y el efecto de sonido del salto, que se escucha cuando saltamos con el jugador.

4.6. Componentes en los elementos principales

4.6.1. Jugador

Player	
Transform	Posición, rotación y escala en el nivel.
Animator	Animación del elemento.
Player Controller (Script)	Script para controlar al jugador.
Sphere Collider	Se encarga de ver si el jugador colisiona con algo.
Rigidbody	Para que le afecte la física del juego.
Grab Object (Script)	Script para controlar si se interacciona con algún objeto.
Audio Source	Efecto de sonido.
Hijos de Player	
Textures	Se muestra una imagen del jugador.
GroundDetection	Sirve para detectar si el jugador está tocando el suelo.
Particle System	Muestra partículas cuando el jugador se mueve.
Hand	Donde se coloca cualquier objeto que el jugador recoge.

4.6.2. Enemigo

Enemy	
Transform	Posición, rotación y escala en el nivel.
Animator	Animación del elemento.
Enemy Controller (Script)	Script para controlar al enemigo.
Sphere Collider	Se encarga de ver si el enemigo colisiona con algo.
Rigidbody	Para que le afecte la física del juego.
Hijos de Enemy	
Textures	Se muestra una imagen del enemigo.
GroundDetection	Sirve para detectar si el enemigo está tocando el suelo.
Particle System	Muestra partículas cuando el enemigo se mueve.

4.6.3. Barrera

Barrier	
Transform	Posición, rotación y escala en el nivel.
Box Collider	Es un Box Collider con un tag específico (Wall) que sirve para que el jugador no pueda atravesarlo.
Barrier Controller (Script)	Script para controlar la barrera, controla si se ha cumplido el objetivo para desbloquear la barrera.

4.6.4. Cámara

Camera	
Transform	Posición, rotación y escala en el nivel.
Camera	Opciones acerca de la cámara y el fondo.
Follow Target (Script)	Script para mostrar la posición del jugador si no está a la vista.
Hijos de Camera	
Raycast	Controla donde apuntas con la cabeza.
CanvasLevel	Interfaz con información acerca del nivel.

4.6.5. CanvasLevel

CanvasLevel	
Rect Transform	Posición, rotación y escala de la interfaz.
Canvas	Opciones acerca del canvas.
Canvas Scaler (Script)	Opciones a la hora de escalar en los diferentes dispositivos.
Graphic Raycaster (Script)	Opciones a la hora de interactuar con objetos que se cruzan en el raycast.
Hijos de CanvasLevel	
Platforms	Imágenes y texto de la interfaz de plataformas.
Coins	Imágenes y texto de la interfaz de monedas.
Clock	Imágenes y texto de la interfaz del tiempo del nivel.
Arrows	Imágenes que muestran las flechas cuando el jugador no está a la vista.
EndWindow	Resumen una vez completado el nivel.

4.6.6. GrabObject

GrabObject	
Transform	Posición, rotación y escala en el nivel.
Rigidbody	Para que le afecte la física del juego.
Box Collider (Trigger)	Detecta si el jugador entra en contacto para permitir o no coger el objeto.
Platform Object (Script)	Script para controlar si puede posicionarse en una plataforma o no, también cambia el color.
Levitate Object (Script)	Controla la levitación del objeto.
Grabbing Object (Script)	Controla si el objeto está siendo sujetado por el jugador o la plataforma objetivo.
Hijos de GrabObject	
GearCoin	La parte visual del GrabObject

4.6.7. ObjectivePlatform

ObjectivePlatform	
Transform	Posición, rotación y escala en el nivel.
Cylinder (Mesh Filter)	La parte visual de la plataforma.
Capsule Collider	Para detectar si el jugador deposita el objeto que se requiere.
Mesh Renderer	Opciones a la hora de renderizar (mostrar) la plataforma.
Platform (Script)	Controla si se ha depositado el objeto que toca.
Hijos de ObjectivePlatform	
LevitatePoint	Punto donde el objeto quedará levitando.

4.6.8. LevelManager

LevelManager	
Transform	Posición, rotación y escala en el nivel.
LevelManager (Script)	Información acerca del nivel (Plataformas a completar, monedas a recoger, nombre y número del nivel y cuál será el siguiente nivel.

4.6.9. CoinObjective

CoinObjective	
Transform	Posición, rotación y escala en el nivel.
Cylinder (Mesh Filter)	La parte visual de la plataforma.
Sphere Collider	Para detectar si el jugador recoge el objeto.
	Opciones a la hora de renderizar (mostrar) la plataforma.
Levitate Object (Script)	Controla la levitación del objeto.
Sphere Objective (Script)	Controla si se ha recogido la moneda para avisar al LevelManager.
Material	Pintar del color que toque la moneda.

4.6.10. FallOffPlatformm

FallOffPlatformm	
Transform	Posición, rotación y escala en el nivel.
Box Collider (Trigger)	Para detectar si algún objeto toca esta caja invisible.

4.6.11. Fan

Fan	
Transform	Posición, rotación y escala en el nivel.
Hijos de Fan	
AirForce	Es una Capsule Collider (Trigger) que tiene un script que ejerce una fuerza a quien entra.
Particle	Dibuja partículas para ver dónde está el ventilador (Fan)

4.6.12. WalkingWall

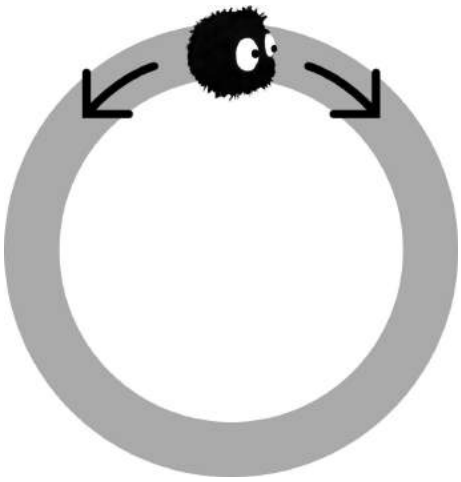
WalkingWall	
Transform	Posición, rotación y escala en el nivel.
Hijos de WalkingWall	
Walking	Es un Box Collider que es invisible que sirve para que el jugador pueda andar sobre este.
Wall	Es un Box Collider con un tag específico (Wall) que sirve para que el jugador no pueda atravesarlo.

4.7. Scripts principales

En esta sección se describirán los fragmentos de código más importantes de cada componente.

4.7.1. Player

Para que el jugador pueda desplazarse en círculo, ha sido necesario implementar una función de movimiento que lo haga. Esto se realiza en el script PlayerController.cs.



La función getHor() se usa al principio para colocar en una posición debida al jugador y devuelve el arcocoseno entre el jugador y el radio.

```
private float getHor() {
	if( transform.position.x > Radius ) {
		transform.position =
			new Vector3( Radius,
				transform.position.y,
				transform.position.z );
	}

	return Mathf.Acos( transform.position.x / Radius );
}
```

Con esto, lo que conseguimos es que nuestra variable de desplazamiento (**hor**) sea la adecuada para la posición del jugador.

Ahora extraemos la parte más importante de la función Update().

```
void Update() {
	mov = Input.GetAxis( "Horizontal" ) * Time.deltaTime * horizontalSpeed;
	isGrounded = Physics.CheckSphere( groundCheck.position, 0.15f, whatIsGround );
	rb.velocity = new Vector3( mov * maxSpeed, rb.velocity.y, 0 );
}
```

En la primera línea de la función detectamos si el jugador está desplazándose horizontalmente una vez por segundo y lo multiplicamos por un valor que elegimos desde el editor de unity.

En la segunda línea, detectamos si el jugador está tocando el suelo, esto se usa para la animación del jugador, si está tocando el suelo, la animación será de andar, si no está tocando el suelo, el jugador estará cayendo, así que la animación será la de caer. Usamos la función ya implementada de CheckSphere y le pasamos los datos de la posición del hijo que estará debajo de player (groundCheck), el radio de la "bola" que queremos detectar si está en contacto con el suelo (0.15F) y le decimos que es el suelo (whatIsGround), que se asigna mediante el editor de Unity.

En la última línea asignamos al Rigidbody del jugador la velocidad de movimiento (en el eje de las X).

Si dejamos esto así, el jugador podría moverse hacia la derecha y la izquierda (eje de las X), no se movería en círculos que es lo que buscamos. Para ello en la función de LateUpdate() colocamos al jugador sobre el círculo que queremos que esté.

```
private void LateUpdate() {
	if( ( mov > 0 && atravesarDer ) || ( mov < 0 && atravesarIzq ) ) {
		hor -= mov;
		transform.position = new Vector3(
			Mathf.Cos( hor ) * Radius,
			transform.position.y,
			Mathf.Sin( hor ) * Radius );
	}
}
```

En esta función comprobamos si el jugador se está desplazando a la izquierda o la derecha y si puede atravesar el objeto que hay delante de él. Si fuera posible atravesar el objeto, modificamos nuestra variable **hor** y colocamos al jugador en la posición que corresponde.

Para calcular esta posición hacemos uso de los cosenos y los senos.

Antes hemos nombrado atravesarDer y atravesarIzq, para comprobar si es posible atravesar el objeto que tiene a un lado y al otro hacemos uso de la función OnCollisionEnter().

```
void OnCollisionEnter( Collision collision ) {
	if( collision.gameObject.tag == "Wall" ) {
		if( mov < 0 ) {
			atravesarIzq = false;
		} else if( mov > 0 ) {
			atravesarDer = false;
		}
		doubleJump = false;
	}
}
```

En esta función comprobamos si el tag del objeto con el que hemos colisionado es el tag “Wall”, significando que si es esto cierto, estamos chocando con una pared. Si la variable mov es negativa, significa que estamos moviéndonos a la izquierda, así que tenemos un muro a la izquierda y no dejamos atravesarlo, cambiando la variable atravesarIzq a falso. Si movimiento fuera positivo, pasaría lo mismo solo que a la derecha. También impedimos que el jugador pueda usar el doble salto cambiando la variable doubleJump a falso.

Coger y dejar objetos. Estas funciones se usan para que podamos completar el nivel, para ello entran en juego tres funciones importantes.

```
private bool isGrabbing() {
	if( Hand.transform.childCount > 0 )
		return true;
	return false;
}
```

```
private void grab( GameObject go ) {
	GrabbingObject s = go.GetComponent<GrabbingObject>();
	if( !s.BeingGrabbed && !isGrabbing() ) {
		s.BeingGrabbed = true;
		go.transform.parent = Hand.transform;
		go.transform.position = Hand.transform.position;
	}
}

private void release( GameObject go ) {
	GrabbingObject s = go.GetComponent<GrabbingObject>();
	if( isGrabbing() ) {
		s.BeingGrabbed = false;
		go.transform.parent = null;
	}
}
```

En la primera función comprobamos si nuestra mano tiene hijos, si los tuviera significa que ya está sujetando un objeto.

En la función grab le pasamos un GameObject, que será el objeto que tendría que sujetar si fuera posible sujetarlo. Accederemos al script GrabbingObject y cambiaríamos la variable pública BeingGrabbed a true para indicar que el objeto está siendo sujetado, cambiaríamos el padre del objeto a sujetar y le indicaremos que nuestro Hand es su padre, y lo pondremos en la posición del objeto Hand.

La función release también le pasamos un GameObject, accedemos al script GrabbingObject y si nuestro objeto Hand está sujetando un objeto, le indicamos que deje de sujetarlo y le cambiamos el padre para que no tenga ninguno.

4.7.2. Camera

En la cámara tenemos un script que hace que veamos unas flechas a la derecha o la izquierda de la pantalla para que sepamos donde se encuentra el jugador, también mueve la cámara verticalmente. Para ello usamos la función Update().

```
void Update() {
	if( !Target.GetComponent<Renderer>().isVisible ) {
		relPoint = transform.InverseTransformPoint( Target.transform.position );
		if( relPoint.x < 0.0f ) {
			LeftArrow.GetComponent<Image>().enabled = true;
			RightArrow.GetComponent<Image>().enabled = false;
		} else {
			LeftArrow.GetComponent<Image>().enabled = false;
			RightArrow.GetComponent<Image>().enabled = true;
		}
	} else {
		RightArrow.GetComponent<Image>().enabled = false;
		LeftArrow.GetComponent<Image>().enabled = false;
	}
	destination = new Vector3( transform.position.x,
				Target.transform.position.y + 5,
				transform.position.z );
	transform.position = Vector3.SmoothDamp( transform.position,
					destination,
					ref velocity,
					0.15f );
}
```

Target será el jugador, se asigna desde el editor de Unity. Comprobamos si está visible, si no lo está calculamos usando la función InverseTransformPoint entre nuestra posición (la cámara, nuestros ojos) y la posición del Target (el jugador). Si el resultado es negativo, mostramos la flecha izquierda que habremos asignado desde el editor de Unity, sino, mostramos la fecha derecha. Si es visible desde nuestra cámara, ocultamos ambas flechas.

Para mover arriba y abajo dependiendo de la posición de nuestro jugador hacemos uso de SmoothDamp, una función que nos mueve desde una posición (transform.position, la posición de la cámara) a la posición destination (la posición Y del jugador + 5, para verlo desde un poco más arriba) a una velocidad (velocity) y en cierto tiempo (0.15f).

4.7.3. GrabObject

El script más importante de este componente es el script GrabbingObject, este script se encarga de permitir que el objeto pueda ser cogido.

```
void Update () {
	if( BeingGrabbed ) {
		rb.isKinematic = true;
		colliderChildren.isTrigger = true;
	} else {
		rb.isKinematic = false;
		colliderChildren.isTrigger = false;
	}
}
```

Cuando convertimos un Rigidbody (rb) en Kinematic, la física deja de afectarle, con esto conseguimos que el objeto no caiga al suelo, ya no le afecta la gravedad.

Cuando le decimos colliderChildren que sea Trigger, le permitimos atravesar objetos.

4.7.4. Funciones comunes

En este apartado se explicarán las funciones comunes de los diferentes objetos.

```
private void OnTriggerEnter( Collider other ) {
	if( other.tag == "FallOffPlatform" ) {
		rb.velocity = new Vector3( 0, 0, 0 );
		gameObject.transform.position = posInicial;
	}
}
```

Esta función estará en todos los objetos que se puedan mover (GrabObject y Player). Con esta función detectamos si el objeto cae al “vacío” (no será el vacío hay una plataforma invisible que impedirá que caiga) y en ese caso, lo devuelve a su posición inicial (cada objeto tiene la suya, se inicializa en la función Start() del objeto).

```
void Start ()
{
	rend = GetComponentInChildren<Renderer>();
	if( ObjectiveTarget == Enums.Objective.RED )
		mat = Resources.Load( "Materials/Platforms/RedPlatform" ) as Material;
	else if( ObjectiveTarget == Enums.Objective.GREEN )
		mat = Resources.Load( "Materials/Platforms/GreenPlatform" ) as Material;
	else if( ObjectiveTarget == Enums.Objective.YELLOW )
		mat =Resources.Load( "Materials/Platforms/YellowPlatform" ) as Material;
	else if( ObjectiveTarget == Enums.Objective.BLUE )
		mat = Resources.Load( "Materials/Platforms/BluePlatform" ) as Material;
	rend.material = mat;
}
```

Esta función está presente en ObjectivePlatform y en GrabObject, se usa para que desde el editor podamos cambiar el color del objeto para que no tengamos que hacerlo a mano.

5. Futuras versiones

Este juego da para mucho *juego*, me gustaría añadir muchas funcionalidades y modos de juego en el futuro.

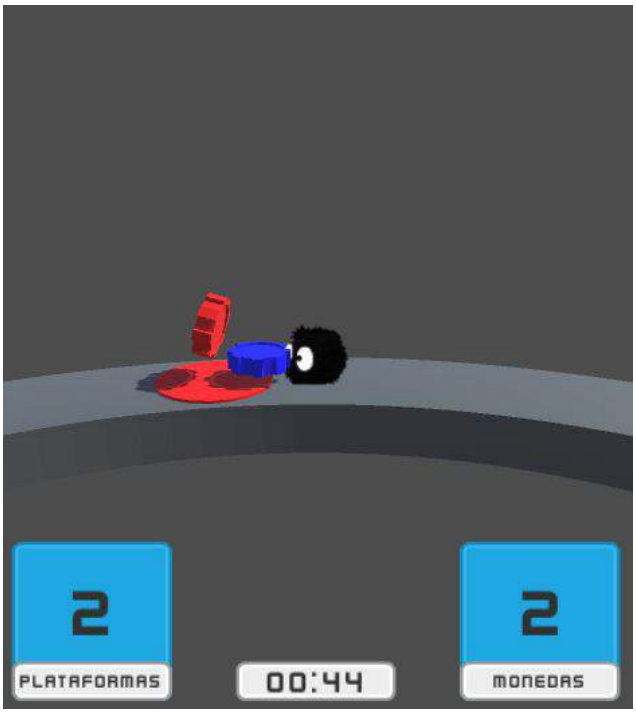
- Multijugador: podrás jugar con otra persona en equipo para lograr completar los niveles cooperando.
- Nuevos elementos: para hacer más entretenido el juego.
- Nuevos niveles: usando los nuevos elementos y hacer más largo el juego.
- Efectos de sonido y música: implementar más sonidos y nueva música al juego.
- Más plataformas: lanzar el juego también en una versión web y consolas.
- Nuevos modos de juego: competir a contrarreloj o tener que eliminar a todos los enemigos.
- Nuevos enemigos: agregar nuevos enemigos más inteligentes y más peligrosos.

6. Resultados

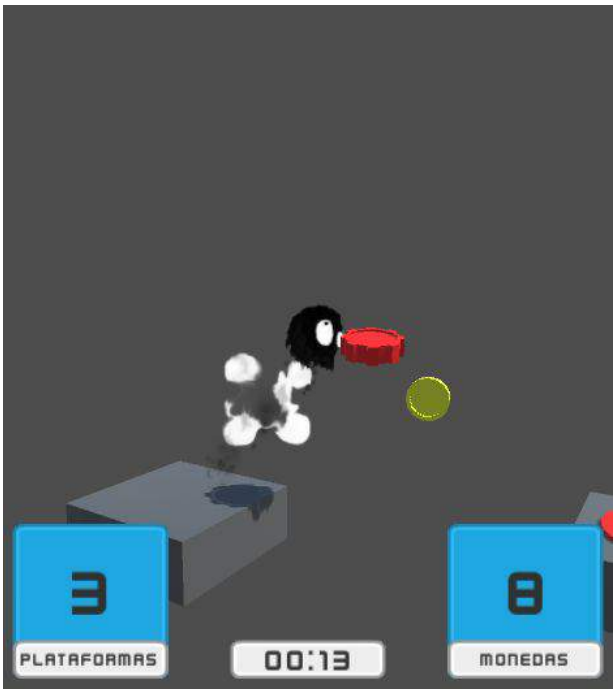
A lo largo del desarrollo me he encontrado con problemas sencillos, cómo hacer que los modelos en dos dimensiones no parezcan que son de dos dimensiones, hasta más complejos, cómo sobre qué hacer para que el jugador se mueva en círculos y pueda chocar con otros objetos. Me ha servido para ver que no todo es fácil a la hora de desarrollar un videojuego.

Desde el punto de vista de la realidad virtual, es complejo, muy complejo desarrollar un videojuego enfocado a realidad virtual, desde los controles hasta los menús se hacen complicados.

Menú principal

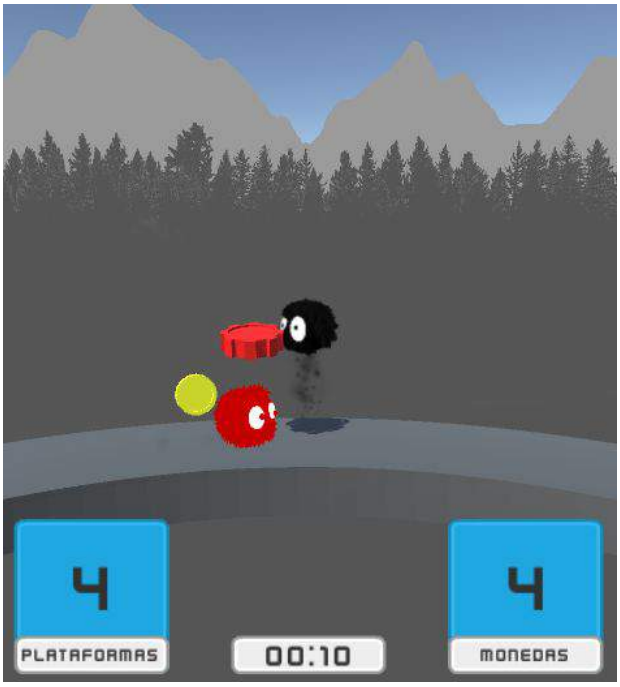


Aquí podemos ver al jugador sujetando un engranaje (azul) y un engranaje ya colocado en su plataforma (rojo).



En esta imagen aparece el jugador saltando mientras sujeta un engranaje y a su derecha hay una moneda.

Aquí vemos al jugador saltando y viéndose afectado por uno de los Fan.



Aquí apreciamos al jugador tratando de saltar por encima al enemigo, sujetamos un engranaje y buscamos recoger la moneda.

7. Bibliografía

https://www.xataka.com/realidad-virtual-aumentada/la-guerra-de-la-realidad-virtual-2016-ya-esta-aqui-comparativa-a-fondo-de-todas-las-opciones

https://vr.google.com/cardboard/

http://www.samsung.com/global/galaxy/gear-vr/

https://www.playstation.com/es-es/explore/playstation-vr/

https://www.oculus.com/

https://www.vive.com/

http://www.androidpit.es/10-posibles-usos-de-la-realidad-virtual-vr

https://www.edx.org/course/introduccion-al-desarrollo-de-upvalenciax-uny201-x-1

https://docs.unity3d.com/

https://www.assetstore.unity3d.com/en/

http://mundo-virtual.com

https://stackoverflow.com/