UNIVERSITAT POLITÈCNICA DE VALÈNCIA

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Politécnica Superior de Gandia

Prueba de concepto de un videojuego de puzles basados en la gravedad

Trabajo Fin de Grado

Grado en Tecnologías Interactivas

AUTOR/A: Candel Sampedro, Yeray

Tutor/a: Palacio Samitier, Daniel

CURSO ACADÉMICO: 2022/2023

UNIVERSITAT POLITÉCNICA DE VALÉNCIA

CAMPUS DE GANDÍA

Grado en tecnologías interactivas

TRABAJO FIN DE GRADO

Prueba de concepto de un videojuego de puzles basados en la gravedad.

Yeray Candel Sampedro

Tutor: Daniel Palacio Samiter

Septiembre 2023

Resumen (castellano)

El objetivo del presente trabajo es el desarrollo de una prueba de concepto de un videojuego en el que se use la gravedad para resolver puzles. El juego consistirá en un cubo dentro del cual se disponen varios objetos y el avatar del jugador, y en una de las paredes una puerta que se abre solo si el puzle se resuelve y por la que ha de salir el avatar del jugador. El jugador puede girar el cubo, sobre su centro y en cualquiera de los tres ejes, de forma que, al caer, los objetos y el avatar adquieran una nueva disposición. Se pueden desarrollar también algunas mecánicas secundarias como plataformas, bloqueo de la gravedad en ciertos objetos, movimiento de ciertos objetos por el jugador, etc.

Los pasos para seguir serán:

a) determinar la mejor forma de implementar la mecánica de gravedad para este juego. Por ejemplo, usando el motor de físicas de Unity, o bien desarrollando un sistema personalizado, una mezcla de un motor personalizado y el de unity, etc.

b) crear un nivel de pruebas en el que se pruebe la jugabilidad de la mecánica de gravedad con los distintos objetos que puede haber y sus interacciones.

c) crear varios niveles (uno por mecánica secundaria) en los que se prueben la jugabilidad que puedan ofrecer las distintas mecánicas secundarias.

d) crear una demo con varios niveles que permita demostrar la jugabilidad y el progreso de dificultad del juego.

Abstract (English)

The objective of this work is the development of a proof of concept of a video game in which gravity is used to solve puzzles. The game will consist of a cube inside which various objects and the player's avatar are arranged, and on one of the walls a door that opens only if the puzzle is solved and through which the player's avatar has to exit. The player can rotate the cube, about its center and in any of the three axes, so that when it falls, the objects and the avatar acquire a new arrangement. Some secondary mechanics can also be developed such as platforms, gravity lock on certain objects, movement of certain objects by the player, etc.

The steps to follow will be:

a) determine the best way to implement the gravity mechanic for this game. For example, using the Unity physics engine, or developing a custom system, a mix of a custom and unity engine, etc.

b) create a test level in which the gameplay of the gravity mechanics is tested with the different objects that may exist and their interactions.

c) create several levels (one per secondary mechanic) in which the gameplay that the different secondary mechanics can offer are tested.

d) create a demo with several levels that allows to demonstrate the gameplay and the difficulty progress of the game.

Palabras clave (Castellano)

Videojuego, demo, gravedad, mecánica, puzle, laberito, 3d, rotación, juego, plataformas, difícil, pensativo, creativo, cubo, resolución, encerrado, portal, físicas, unity, blender, desarrollo, código

Keywords (Inglés)

Video game, demo, gravity, mechanics, puzzle, maze, 3d, rotation, game, platform, difficult, thoughtful, creative, cube, resolution, locked, portal, physics, unity, blender, develop, code

Agradecimientos

Gracias a este TFG he podido desarrollar uno de los videojuegos que he tenido en mente desde que empecé a desarrollar-los. Siempre me pareció muy interesante la generación de figuras imposibles y resolución de puzles mediante la implementación de estas, obligando al cerebro a imaginar espacios y solucionar problemas que no serían posibles de plantear si no existiesen estos videojuegos, por eso decidí desarrollar una serie de puzles relacionados con la gravedad y generación de espacios euclídeos para potenciar la mente en resolución de problemas. Durante el desarrollo de este TFG he recibido apoyo de mis familiares y amigos, los cuales han visto todo el proceso y problemas que han ido surgiendo y que por ende me gustaría agradecer-les todo el apoyo recibido y por haber ido probando poco a poco este proyecto. También agradecer a mi tutor por que trabajar con alguien por tanta pasión por los videojuegos hace que el camino sea mucho más ameno y ser un gran líder para este desarrollo.

INDICE DE CONTENIDOS

1. Introducción

1.1 Presentación

El presente proyecto se desarrollará usando la plataforma gratuita de desarrollo de videojuegos 3D llamada Unity. Elección de esta plataforma porque es la que se ha utilizado para desarrollar videojuegos durante el grado de Tecnologías Interactivas, aplicando los distintos conocimientos adquiridos durante el curso para desarrollar las distintas mecánicas y desafíos que implican el consecuente proyecto como desarrollar un nuevo sistema de gravedad mezclando técnicas que tiene el propio motor de Unity y desarrollando nuevas técnicas para el correcto desarrollo del videojuego.

1.2 Motivación

Desde que empecé en el mundo de los videojuegos como consumidor, sentí una gran pasión por ellos. De todos los videojuegos que he jugado hay uno que me marco la vida por completo el cual trataba acerca de una persona sin nombre que poseía una pistola de portales la cual era utilizada para resolver puzles mediante el uso de los portales y objetos que habían dentro de la sala. El videojuego es llamado “Portal” y fue un boom entre los consumidores de videojuegos pues este era uno de los primeros videojuegos que creo una forma de resolver puzles en forma de visión espacial. Había que ser muy habilidoso para entender este juego a la primera pues todos los que lo jugaron por primera vez eran incapaces de interpretar la resolución del puzle incluso aun después de haberlo resuelto.

Es desde ese día, que desarrolle una gran pasión por puzles de ese tipo y siempre he esperado el nuevo lanzamiento de algún videojuego similar. Hubo uno que intentó convencerme el cual era “The Entropy Center” pero al final todos sus puzles resultaban ser muy sencillos de resolver y se hacían repetitivos.

Pues entonces fue cuando decidí atreverme a crear mi propio videojuego de puzles en el que el cerebro humano fuese incapaz de entender como en ciertos momentos el personaje principal puede estar en distintas salas a vez y tener que resolver puzles dentro de estas salas con la posibilidad de rotar-se en todos sus ejes por elección del jugador y resolver puzles con los objetos que están siendo afectados por un sistema de gravedad propio con el que se puedan resolver los puzles correctamente.

Pienso que este proyecto aparte de ser un reto personal para mí creo que podría aportar bastante a la industria ya que hay muy pocos juegos de resolución de puzles que aporten algo de valor adquirido a esta. Pues este videojuego se plantea desde un jugador para jugadores y no como un proyecto para sacar dinero, y con el objetivo de romper la cabeza de los propios jugadores y poder mandarles un reto con sentido, complejo, pero con enganche a la resolución de problemas.

En fin, mi motivación por este proyecto se basa en la pasión por el desarrollo de videojuegos y en especial mi interés por la física y la resolución de puzles que requieran de habilidades más allá de las que se puedan ver a simple vista, a parte me brinda la oportunidad de superación de mí mismo por desarrollar puzles con figuras euclídeas con un sistema de gravedad propio.

Se trata de un videojuego de puzles que requerirá del usuario crear espacios en su mente para resolver los problemas planteados en el interior de un cubo el cual puede rotar en cualquier dirección sobre su eje usando elementos que hay en su interior que reaccionan en la gravedad. De esta manera también podrá desarrollar un aprendizaje sobre el área espacial y ver soluciones más allá de lo que simplemente tiene delante.

1.3 Objetivos

El objetivo de este TFG es la creación de un videojuego que contenga los siguientes puntos:

1. Diseñar y desarrollar un sistema de rotación de salas sobre su propio eje en el que el jugador pueda rotar en cualquier dirección y sentido.

2. Diseñar y desarrollar un motor de físicas para ciertos elementos del escenario reaccionen correctamente a la rotación del cubo, incluyendo el jugador, el cual seguirá la dirección de la gravedad de los objetos.

3. Desarrollar un sistema de colisiones para que los objetos que sean movidos por el sistema de motor de físicas propio sean capaces de detectarse entre ellos y permitir un correcto funcionamiento de estos cuando el cubo esté en rotación.

4. Implementar un sistema de control e interfaz intuitiva y fácil de utilizar que permita al jugador rotar el cubo y moverse de el sin complicaciones. La idea es que el usuario rote el cubo en función de donde esté mirando y así tener una referencia visual para rotar el cubo en los ejes horizontal/vertical.

5. Crear la mecánica principal de resolución de puzles la cual predominara en la mayor parte de los niveles. Concretamente desarrollar la mecánica de figuras euclídeas/imposibles mediante la utilización de portales y copia del jugador en distintas salas para simular la estancia del jugador en todas estas a la vez.

6. Crear una serie de mecánicas extras que añadan cierta dificultad al videojuego y ser más entretenida la experiencia.

7. Crear una serie de niveles de prueba para probar cada una de las mecánicas que se van a ir mostrando durante el proceso del jugador en la partida.

En resumen, el objetivo principal del desarrollo del videojuego es explorar todas las opciones para desarrollar un motor de físicas propio para la reacción de los objetos en función de la rotación del cubo en el cual va a estar el jugador y además explorar la mecánica principal de desarrollar figuras imposibles.

1.4 Estructura de la memoria

La memoria consta de 5 capítulos:

1. Introducción: Presentación de la herramienta utilizada para desarrollar el videojuego y una breve introducción y objetivos del juego a desarrollar en cuestión.
2. Entorno del Arte: Análisis de los fundamentos teóricos acerca de la física en el mundo real, teorías y exploración de cómo funcionan las figuras imposibles y la física de los portales y exploración de rotación del escenario y diseño de los niveles para la presentación de las distintas mecánicas.
3. Diseño y desarrollo: Breve explicación de cómo se han implementado las distintas mecánicas como la rotación del escenario sobre sus propios ejes, el motor de física de los objetos y sus colisiones entre ellos y el sistema de portales para simular un espacio euclídeo / de figuras imposibles.
4. Resultados: Exposición de los resultados tras la aplicación de todos los puntos anteriores y muestra de los niveles de prueba.
5. Conclusiones y propuesta de trabajo a futuro: Breve conclusión acerca del desarrollo del videojuego y visión a futuro de este desarrollo.

1.5 Relación con ODS

Este proyecto tiene como objetivo desarrollar un prototipo de un videojuego basado en un nuevo sistema de físicas y figuras no euclídeas. Aplicando este proyecto a ODS, no podría entrar en ninguno de sus apartados a menos relacionarse con el ODS 4, ya que este videojuego se podría utilizar para desarrollar la capacidad de entender los espacios no euclídeos y desarrollar una visión espacial y capacidad de resolver puzles poco comunes.

Ver Tabla de ODS en Anexo 1.

2. Entorno del arte

2.1 Recorrido juegos en sistemas de físicas y gravedad y puzles

Haciendo un recorrido por los principales juegos con un sistema de físicas propia y con puzles de figuras imposibles (no euclídeas) podemos destacar dos grandes títulos por los que se ha inspirado el desarrollo de este videojuego.

El primer videojuego para destacar se trata de "Portal" lanzado por Valve en 2011. Este juego fue un gran avance para la industria debida a las mecánicas que incluía el título. Consistía en tener una pistola la cual era utilizada para crear 2 portales. Estos 2 portales estaban comunicados entre sí, de esta manera si entrabas por uno podías salir por el otro y viceversa.

El primer contacto con el videojuego es bastante confuso ya que hay que entender la teoría de los portales, para posteriormente resolver los puzles que se plantean.

El segundo videojuego para destacar por el cual sentí curiosidad por crear algo similar se trata de "SUPERLIMINAL" el cual trata de coger objetos y moverlos por el escenario para que estos cambien de tamaño según en que perspectiva estes mirando al objeto.

Hay más juegos que tiene juegan con mecánicas de figuras imposibles como "Monument Valley" o antichamber.

2.2 Diferenciación de mi juego en respecto a los demás

La gran diferencia del desarrollo de este videojuego respecto de los demás es que hasta el momento es el único que intenta crear figuras imposibles y que a su vez se necesite de resolver de unos puzles planteados en estas figuras, y que para resolverse se necesite de objetos que son movidos independientemente del jugador, es decir, que el jugador no necesite coger ningún objeto para "cambiar" de escenario.

Esto es la gran dificultad pues en la misma sala, el objeto utilizado para resolver el puzzle, ha de ser capaz de atravesar el "portal" que limita la figura imposible y llegar al otro portal.

El usuario ha de interpretar en todo momento que el objeto esté en la misma sala en todo momento, y esto es muy difícil de simular ya que hay objetos que se están moviendo por los distintos escenarios en todo momento.

2.3 Como implementar un sistema de físicas propio y los distintos sistemas de Unity

Antes de desarrollar un videojuego en general, hemos de analizar los sistemas de físicas propios que llevan cada uno de los motores para elegir correctamente el motor de videojuegos sobre el que vamos a desarrollar.

En nuestro caso hemos elegido Unity porque gracias a su sistema de físicas podemos utilizar muchos de los métodos públicos que este utiliza para simular cualquier tipo de físicas.

En Unity podemos observar que incluye 2 sistemas de físicas distintos, 1 para 3D llamado PhysX desarrollado por Nvidia que incluye las siguientes características principales:

- Componentes de físicas 3D: En el motor de físicas 3D, se utilizan componentes como RigidBody, Collider y Joint para controlar las físicas de los objetos 3D.
- Simulación tridimensional completa: El motor de físicas 3D proporciona simulaciones detalladas en un espacio 3D, considerando aspectos como colisiones, gravedad, fuerzas, torque y fricción.
- Gravedad 3D: Se aplica la gravedad 3D para simular la caída natural de los objetos y la interacción con la gravedad en todas las direcciones.
- Movimiento y rotación 3D: Los objetos pueden moverse y rotarse en un espacio tridimensional, lo que permite implementar movimientos complejos y realistas para personajes y objetos del juego.

3. Diseño y desarrollo

3.1 Rotación del escenario

Uno de los desarrollos más complejos de este proyecto, puesto que había que desarrollar una rotación del escenario mientras el jugador se situaba dentro teniendo en cuenta donde estaba este y hacía que pared estaba observando este.

3.1.1 Planteamiento rotación local

El cubo solamente se puede rotar en cualquiera de los 3 ejes estrictamente, esto quiere decir qué, el cubo ha de rotar unos 90 grados en cualquiera de los 3 sentidos siempre que el usuario haya pulsado una acción para que esto ocurra. La velocidad de rotación del cubo dependerá de la opción más interactiva con el jugador.

En este caso se ha decidido una rotación de 1 segundo para que los objetos que haya en el interior reaccionen totalmente homogéneos con el entorno del escenario.

Para rotar el cubo se decidió una técnica para que este rotase sobre su propio eje. Para que el cubo rote sobre su centro, en Unity no hay una opción incluida para esto por lo que la única solución es crear un objeto vacío, ponerlo en el centro exacto del cubo y asignar el escenario como hijo a este objeto, y rotar este objeto padre del cubo para que el hijo reciba la misma rotación.

Para elegir la dirección de rotación del cubo debemos conocer sobre qué cara está situado el jugador y a que cara está observando. Como sabemos un cubo tiene 6 caras por lo que asignando un identificador a cada una de esas caras podríamos saber dónde está situado el jugador y hacía que pared está este observando.

Las paredes fueron etiquetadas con los nombres: Front, Back, Right , Left, Bottom, Top

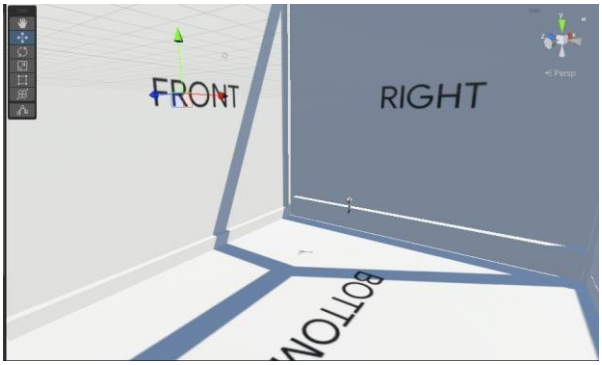

Imagen 1. Etiquetado paredes

El principal problema de esta idea fue que, al rotar el cubo, el jugador seguía en la misma pared “técnicamente” pero el cubo estaba rotado realmente. Por lo que había que añadir una segunda condición la cual era, aparte de comprobar en que pared estaba situada el jugador y hacia donde estaba mirando, comprobar en que grado de rotación estaba el cubo.

Esto conllevó una gran serie de problemas ya que había rotaciones que coincidían con otras debido al motor de Unity en sí para rotar objetos localmente sobre su propio eje, por ejemplo, si has rotado un objeto 360º este en el logger aparece como 180º y no sabes si se ha rotado por un lado o por el inverso.

El segundo gran problema fue el largo desarrollo que conlleva ya que hay que observar todas las posibles rotaciones y situaciones del jugador.

Sabiendo que el cubo puede rotar 90º hasta 360º teniendo un total de 4 posiciones disponibles (90º, 180º, 270º, 360º) y tenemos 3 ejes disponibles (X,Y,Z). Tenemos un total de 4 rotaciones ^3 ejes = 64 posibilidades, es decir, 64 condicionales para rotar el cubo. Además, añadiendo las condiciones de saber de anterioridad hacia donde mira el jugador y en que pared está situado. 4 direcciones de mira del jugador * 6 caras del cubo = 24 condiciones extra. 24 * 64 = 1536 posibilidades. Lo cual es demasiado extenso y sin sentido, aunque haciendo diversos bucles y condicionales iguales se podría evitar esta última multiplicación de posibilidades y solo quedarse en 64.

3.1.2 Planteamiento con rotación global

Así que se canceló esta idea y se planteó una nueva, la cual consistía en rotar el cubo de manera global.

La gran diferencia de la rotación global a la rotación local es que, en la rotación global, independientemente de cómo esté orientado actualmente el objeto, este cambiará su orientación con respecto al mundo.

Por ejemplo, si un objeto está orientado hacia el eje positivo de X y se aplica una rotación global de 90° alrededor del eje Y, el objeto se rotará 90° hacia arriba desde su posición inicial en relación con el mundo, independientemente de su orientación actual.

Mientras qué, en la rotación local, la rotación se realiza en relación con su propio sistema de coordenadas local. Esto quiere decir que la rotación afectará a la orientación del objeto en su propio espacio local.

Por ejemplo, si un objeto está orientado hacia el eje positivo de X en su propio sistema de coordenadas local y se aplica una rotación local de 90° alrededor del eje Y, el objeto se rotará 90° hacia arriba desde su posición inicial en relación con su sistema de coordenadas local. Esto puede resultar en una rotación diferente en el espacio global, dependiendo de cómo esté orientado el objeto en ese momento.

Ejemplo práctico: Si estamos en rotación del cubo X,Y,Z (90,180,0)

Si queremos rotar en el eje Y +90º, el resultado en la rotación global sería (90,270,0) mientras que el resultado en la rotación global podría ser (caso inventado) (0, 270, 90) por que rota de manera distinta y se ve reflejado de manera distinta en el inspector.

Conociendo el comportamiento de la rotación global, ahora existía un problema el cual consistía en que aún se debía saber en qué pared estaba el jugador y hacia cual estaba mirando para poder hacer la rotación desde el modo “jugar” correctamente.

Este problema se solucionó con crear un cubo auxiliar con las caras identificadas anteriormente y escalarlo a un 150% mayor al cubo original en el que se iban a producir los puzles. Ver imagen siguiente:

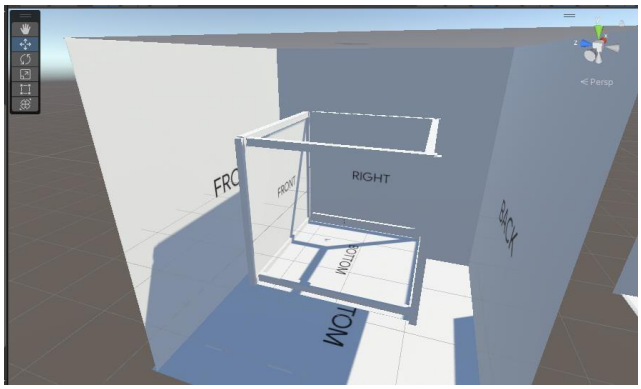

Imagen 2. Paredes extras en escenario

Con este nuevo cubo por el exterior del principal se recogían los datos de donde estaba situado encima el jugador y hacia donde estaba mirando y en función de eso rotar el cubo principal en su eje correspondiente mientras que el cubo auxiliar nunca rota.

Con esto ganamos que independientemente de que rotación este el cubo principal podemos rotar el eje que queramos en cualquier momento. Por ejemplo, si estamos situados sobre el eje Y (bottom) y estamos mirando hacia el eje Z (frente) podemos rotar en el eje Y de manera horizontal y en el eje X de manera vertical. Por lo que el número de condicionales solamente se reduce a 24, como hemos visto en casos anteriores 4 direcciones de mira del jugador * 6 caras del cubo = 24 condiciones.

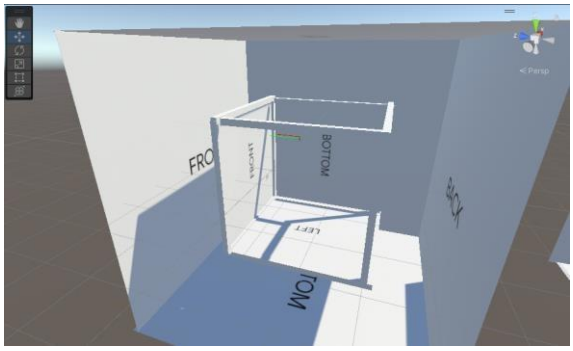

Imagen 3.

3.1.3 Solución rotación en todos los escenarios

Para hacer la mecánica de niveles euclídeos, se requiere de cubos extra, estos también requieren que roten con el cubo principal.

La solución planteada para este problema consta de, crear un objeto vacío en el nivel y cuando ejecutamos la orden de rotar el cubo principal, lo que hacemos realmente es rotar este objeto vacío y luego con el cubo principal lo que hacemos es copiar la rotación exacta de este objeto a sí mismo.

Esto aplica para todos los cubos u escenarios que queramos añadir, todos copian la rotación de este objeto vacío.

3.2 Interacción del jugador con la rotación del cubo

Para la implementación de la rotación del escenario se planteó una opción para que el jugador pudiese relacionarse fácilmente con los controles y rotar el escenario se haga de manera intuitiva.

La opción que se planteó desde el primer momento y la que se estableció consta de, 2 líneas invisibles creadas por "gyzmos", una funcionalidad de Unity, la cual nos permite realizar ciertas acciones dependiendo de nuestras necesidades. En este caso hemos decidido crear estas rectas para tener la información de hacia dónde está mirando el jugador y en que suelo está situado (estas rectas no se pueden ver en el juego).

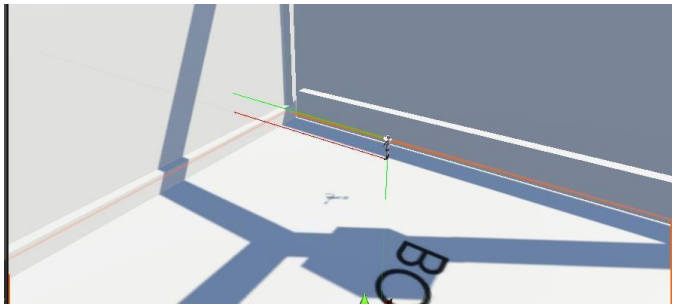

Imagen 4. Muestra gyzmos Jugador

Se ha decidido usar esta técnica porque con mirar a diferentes paredes podemos rotar el cubo el cualquiera de sus ejes, dependiendo del suelo en el que estemos situados. Aquí entra un concepto un poco ambiguo en el que el suelo puede ser cualquier pared y cualquier pared puede ser suelo.

La gran **ventaja** de esta técnica es que con solamente 4 interacciones podemos rotar en cualquiera de sus ejes (lo normal hubiese sido 6 interacciones una para cada dirección de su eje), la única **desventaja** es que no se puede rotar sobre 1 de los ejes dependiendo de donde se sitúe el jugador y a su vez dependiendo de hacía que dirección está observando.

Una recta horizontal que: se desplaza desde la cabeza del jugador hasta el frente unas 50 unidades (correspondientes a la unidad de medida de unity) para poder leer la pared a la que se está observando desde cualquier parte del escenario.

Esta recta nos proporciona una información para saber hacia qué dirección debemos rotar el escenario. En el caso de la imagen si decidimos rotar el escenario en el eje Z (color azul) debemos mirar hacia la pared en la que se está mirando en la imagen y pulsar una tecla que permita rotar horizontalmente el cubo en ese eje. De esta manera podemos rotar el cubo en esa dirección. Como se puede observar en la siguiente imagen.

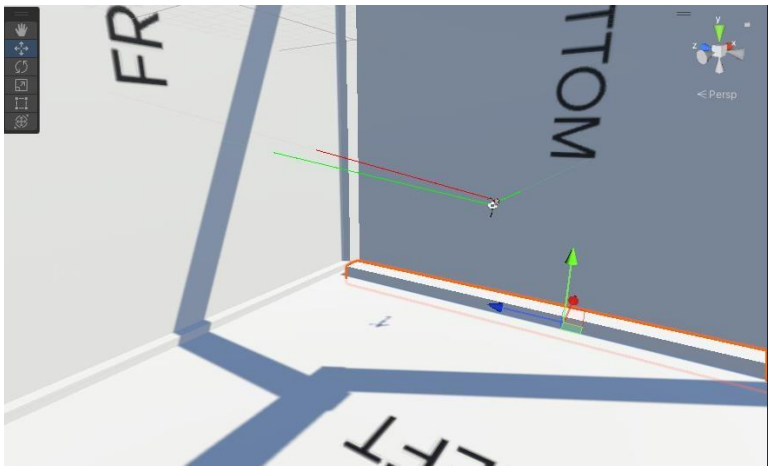

Imagen 5. Jugador rotado 90º a derecha

Después de producirse esta rotación el suelo ha pasado a ser la pared de la derecha, y la pared izquierda (que ahora se puede visualizar) ha pasado a ser el suelo.

Si ahora queremos rotar el cubo de en el eje Z para volver a su posición inicial, tenemos que, el jugador esta sobre el eje X por la recta que sale de sus pies hacia el suelo, y que además debe observar hacia el eje Y para poder producirse esta rotación. Como se observa en la siguiente imagen.

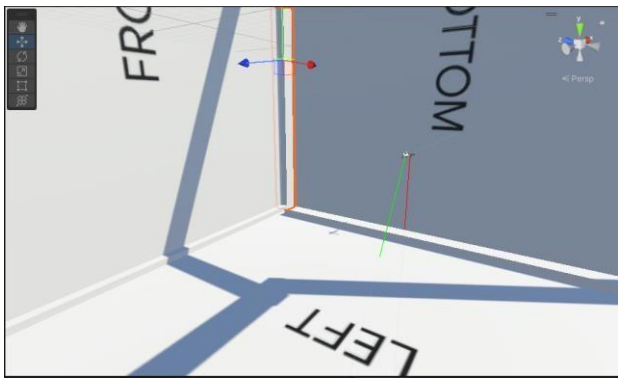

Imagen 6. Jugador mirando hacia left

Hechos todos los anteriores pasos debemos pulsar la tecla la cual nos permita rotar en el ejeZ y todas las paredes y suelos deberían recuperar su posición inicial tal y como se observa en la siguiente imagen:

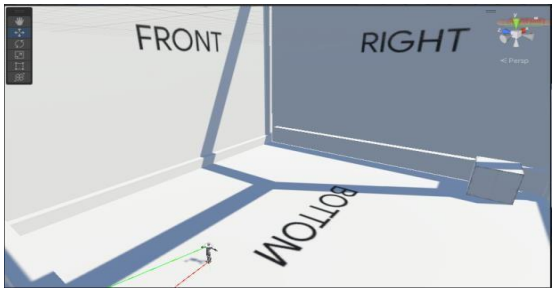

Imagen 7. Jugador rotado hacia adelante 90º El jugador se ha quedado adelantado por la propia rotación en el interior de las paredes.

3.3 Gravedad

La implementación del motor de físicas en los objetos ha sido creada mediante una mezcla del propio motor de físicas de Unity y unas combinaciones propias para un correcto funcionamiento de los objetos.

3.3.1 Gravedad de los objetos

El sistema de físicas ha sido diseñado para que los objetos siempre caigan verticalmente o tengan un sentido de la gravedad igual a la del jugador. Esto quiere decir que los objetos cambiarán su sentido de la gravedad siempre que el jugador lo decida.

Los objetos caen verticalmente con deslizamiento por el escenario desde el primer momento en el que este empieza a rotar, depende del grado de rotación en el que se encuentre el escenario el objeto caerá con más rapidez o no.

El problema principal de usar el motor de físicas de unity y no crear uno propio, es que, al rotar el escenario los propios objetos no se deslizaban por el escenario y lo que ocurre es que estos salen impulsados y rodando hacia el lado contrario en el que está situado el objeto.

Imagen de muestra antes de la rotación del escenario en los 2 objetos.

Amarillo con físicas de Unity.

Rojo con físicas propias.

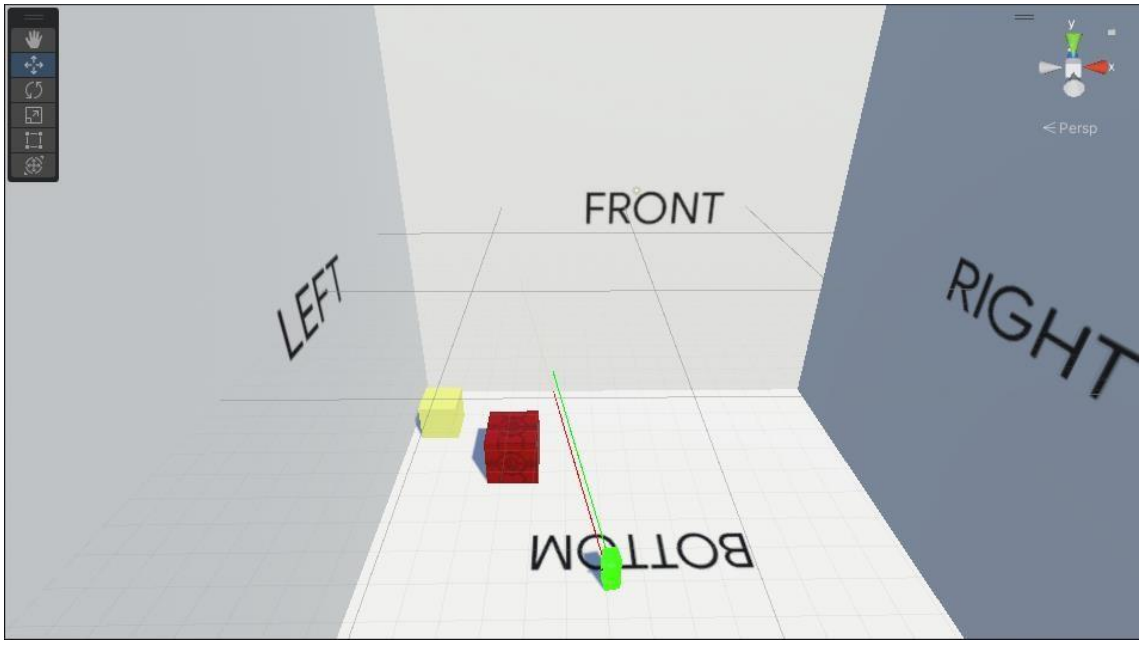

Imagen 8. Muestra objetos en escenario

Como se puede observar no hay ninguna rotación en el cubo rojo mientras que el cubo amarillo sale disparado y rodando. Para la finalidad de nuestro juego y poder resolver los puzles correctamente necesitamos el comportamiento con el motor de físicas propio.

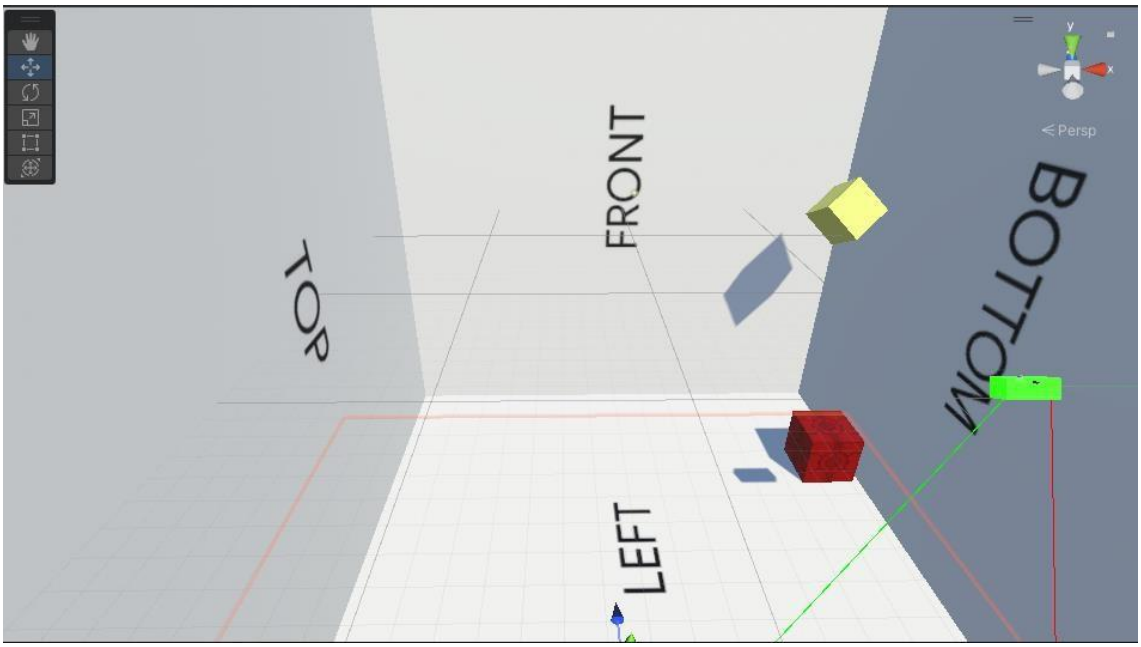

Imagen 9. Objetos reaccionando a la rotación

En la siguiente imagen podemos observar cómo hay 3 objetos. El amarillo con el motor de físicas de Unity, uno transparente para ver la recta roja y otro totalmente opaco. Los 3 están reaccionando a la rotación del cubo justo cuando este ha iniciado su marcha.

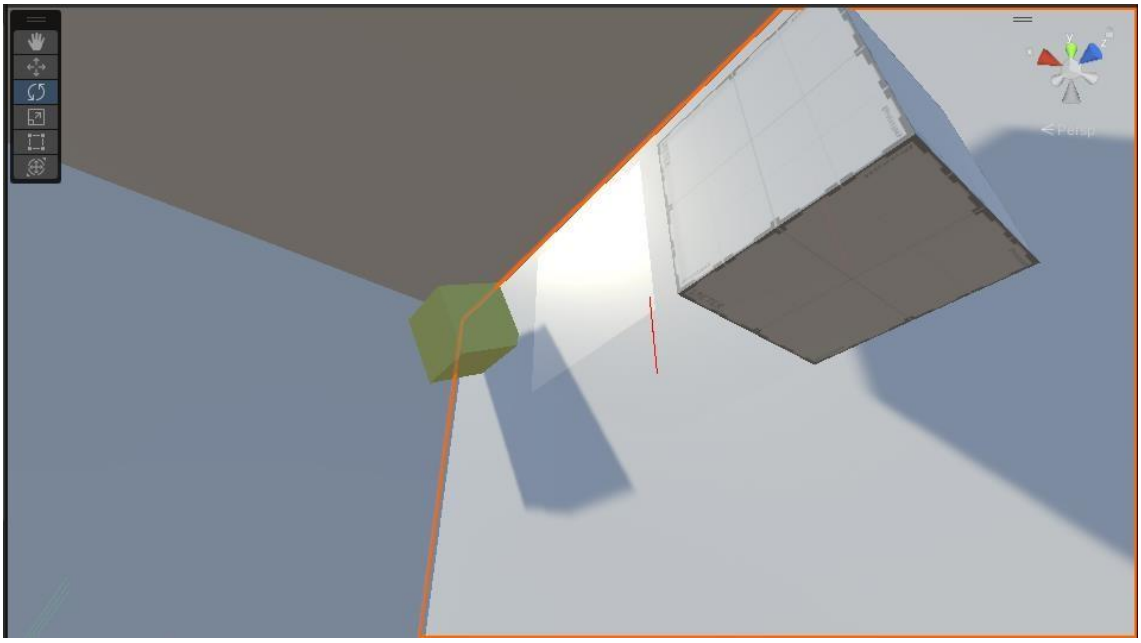

Imagen 10. Objetos reaccionados a la rotación

3.3.2 Interacción entre los objetos y entorno

La interacción entre los propios objetos es muy importante pues si disponemos de varios objetos con un sistema de motor propio, estos no deben atravesarse entre ellos y a su vez deben de tener un sistema de físicas lo más acercado a la realidad para simular cercanía al jugador, pero no muy excesivo para evitar los errores que ocurrían con el cubo amarillo.

Para desarrollar este sistema de colisiones lo que hemos decidido ha sido crear una recta que llega desde el centro del objeto hasta sus límites visibles.

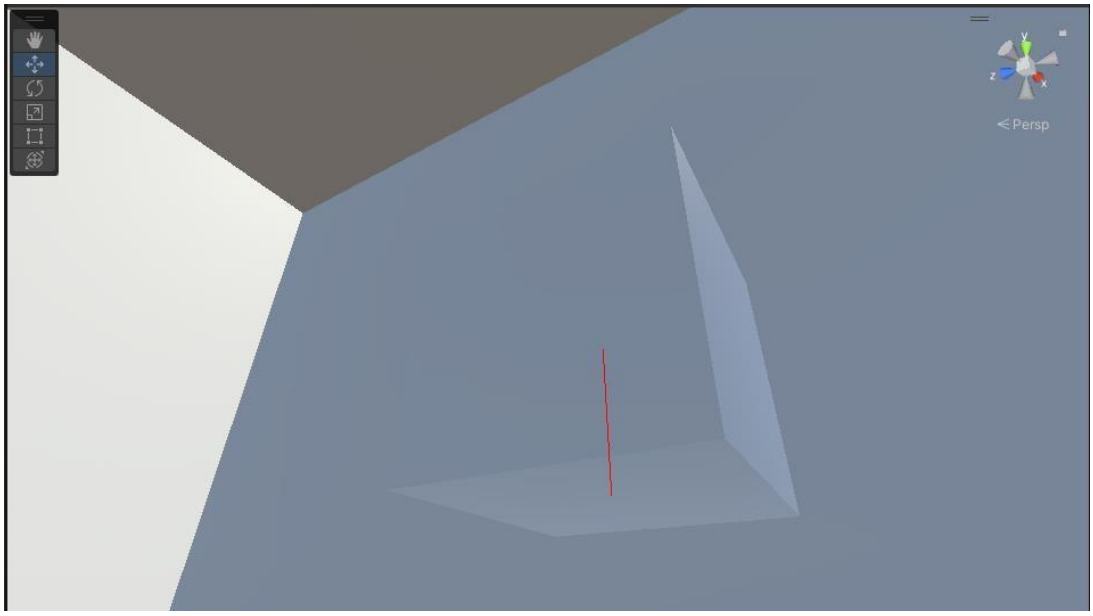

Imagen 11. Muestra gyzmos detección de colisiones

La recta siempre apunta desde el centro hacía la dirección de la gravedad que el jugador haya elegido. Por ejemplo, si la gravedad está en el eje Y esta recta tomara ese eje como referencia.

Con esta recta podemos saber cuándo esta esté objeto tocando el suelo del escenario o está encima de otro objeto. Si no está encima de ningún objeto o escenario se aplica una fuerza exponencial que representa el sistema de físicas del mundo real. Para que el objeto no rote, desde el momento en el que empieza a caer bloqueamos todas sus rotaciones.

En el momento en el que la recta este a 0.1 unidades e iba a más de 3 unidades/s cerca del suelo, desactivamos este sistema propio y activamos todos los otros sistemas como si tuviese interacción real, con fricción y rotación.

Si por casualidad el cubo cae con demasiada rapidez y de forma inclinada debido a que el jugador rote el escenario con demasiada rapidez, esté objeto intentara rotar sobre sí mismo debido a la fricción que genera con el suelo.

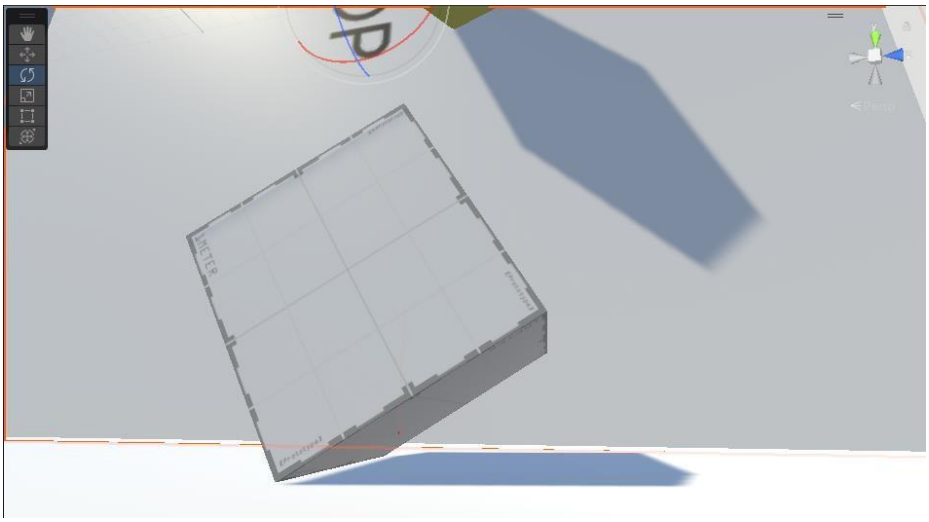

Imagen 12. Respuesta de objeto al sistema de gravedad propio

En la siguiente imagen del inspector del objeto se puede visualizar como se han activado todas las velocidades del propio sistema de unity, mezclado con el propio. Se han desactivado todas las tensiones para evitar errores.

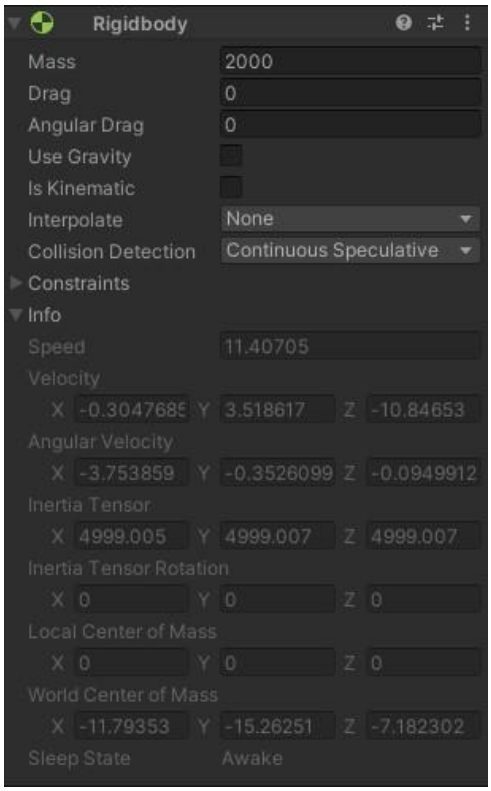

Imagen 13. Propiedades Rigidbody del objeto

La interacción de los objetos entre ellos mismos actúa de una forma parecida con el entorno, pero con una mínima diferencia.

Puede haber una posibilidad de que el objeto caiga encima del otro en una posición desplazada, y el cubo debe caer por su propio peso y rotar.

Imagen antes de la rotación

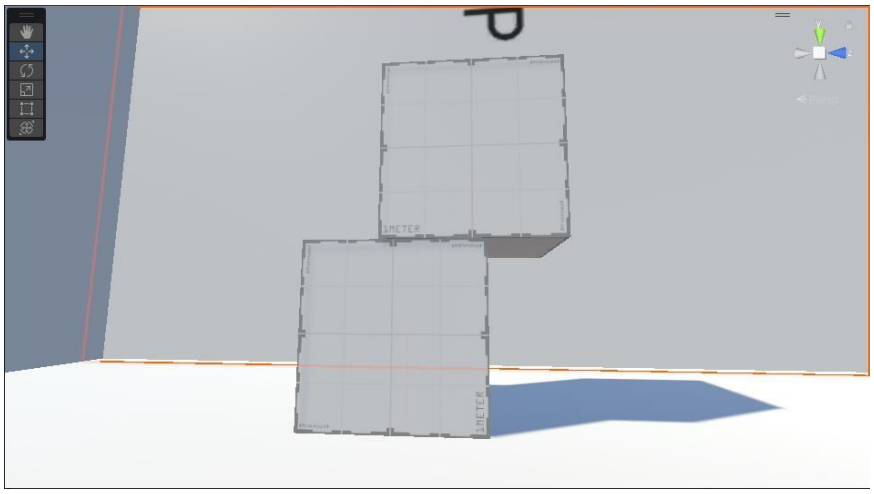

Imagen 14. Objetos antes de reaccionar a la gravedad

Imagen después de la rotación.

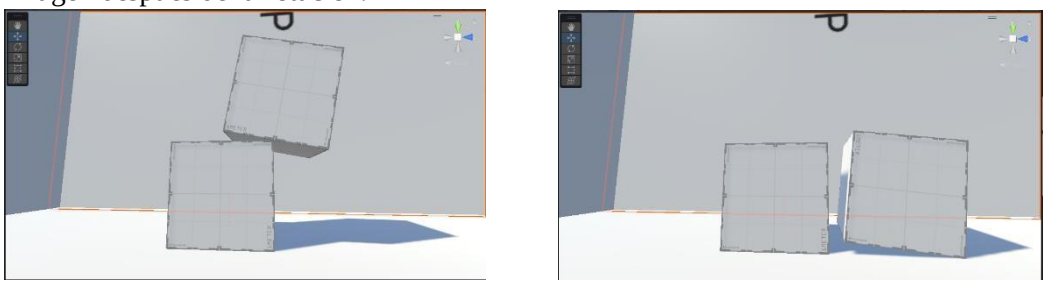

Imagen 15. Objetos después de reaccionar a la gravedad

De esta forma obtenemos un motor de físicas realista pero que se adapte a las necesidades para resolver los puzles.

3.3.3 Gravedad en el jugador

La gravedad utilizada por el jugador utiliza la misma técnica que usamos con los cubos. No podemos usar el sistema de gravedad para controlar el jugador "CharacterController" que nos proporciona Unity porque este nos causa un problema el cual hace que la gravedad siempre caiga en el eje Y, lo cual no queremos.

El jugador ya que está ligado al escenario como hijo, este siempre va a tener la misma rotación que tiene el escenario, de manera que, si el escenario rota el suelo de manera que el suelo quede en una pared, el jugador va a estar pegado a la pared, como hemos visto en imágenes anteriores.

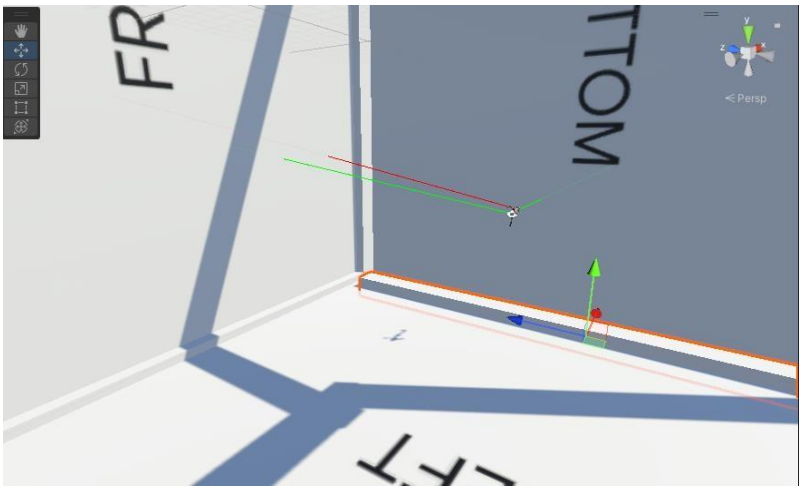

Imagen 16. Jugador pegado a la pared

3.4 Portales

Los portales son elementos esenciales para recrear espacios no euclídeos.

Para crear un portal es necesario que existan 2 planos y 2 cámaras. Por el que cualquier elemento pueda entrar y otro por el que los elementos puedan salir.

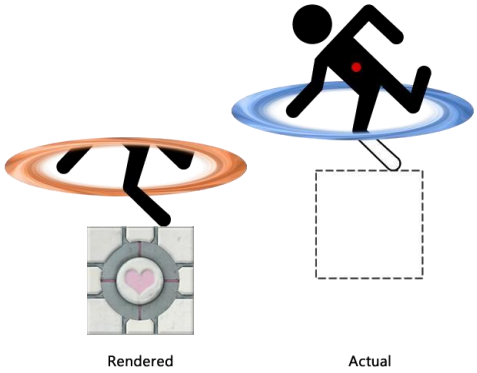

Imagen 17. Visualización de actuación de un portal

En nuestro caso necesitamos los portales para simular que el jugador está en 2 sitios a la vez.

3.4.1 Funcionalidad principal

Poniéndonos en situación, el jugador está dentro del escenario, y debe estar pudiendo ver 2 escenarios a la vez, es necesario que, tengamos un plano que represente lo que debería estar viendo el usuario si estuviese dentro del otro escenario.

Para esto creamos un plano en el escenario en el que queramos que el usuario lo vea.

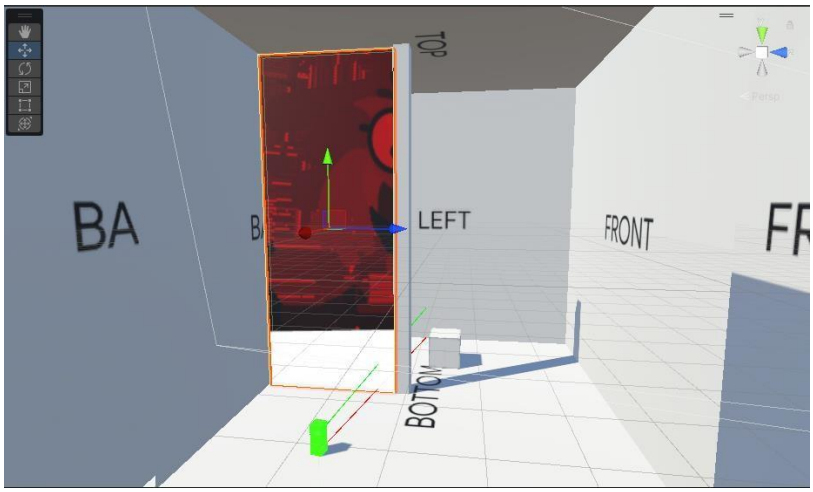

Imagen 18. Muestra plano portal

Luego creamos una cámara que copie exactamente todos los movimientos que está haciendo el jugador tanto de movimiento del cuerpo como la visión. Esta cámara debe tener la misma distancia de separación al jugador como los portales se distancian entre sí.

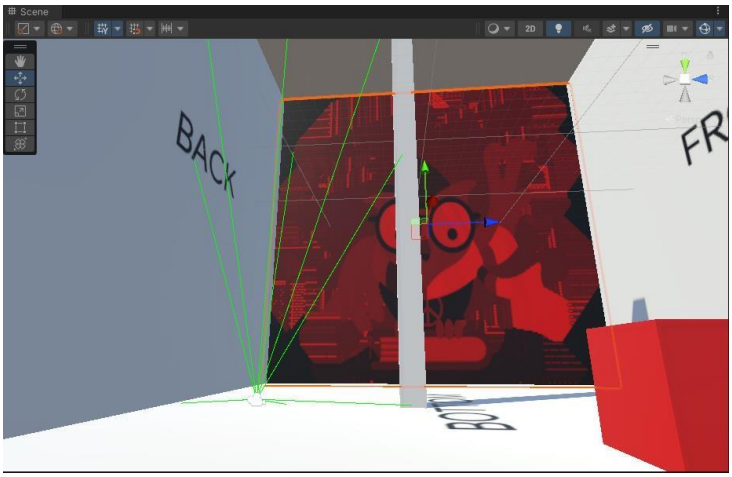

Imagen 19. Visualización escenario 2 sin Portal

Este sería el resultado de la técnica anterior:

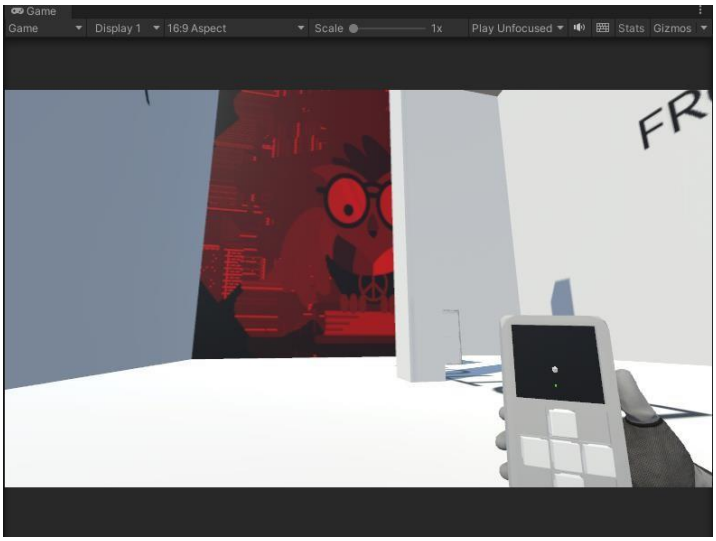

Imagen 20. Resultado mezcla de los 2 escenarios con 1 portal

Este efecto también se consigue gracias a un shader aplicado a este plano simulado como portal

Este Shader no trata la imagen como si fuese una textura, esto quiere decir que, si se estuviese tratando como una textura, estiraríamos el plano y la imagen que genera la cámara sobre el plano quedaría totalmente distorsionada. Entonces con este shader conseguimos que en el plano se muestren los pixeles que realmente está captando la cámara, para representar en dicho plano. Si el plano es más ancho o alto, se vería más cantidad de pixeles que está observando la cámara.

3.4.2 Interacción con el jugador

Para dar utilidad a estos portales es necesario que tanto el jugador como los objetos tenga una interacción con estos y sean capaces de teletransportar la entidad que este atravesando el portal en ese momento.

Para desarrollar la técnica de teletransportación a través de los portales en el jugador, se han creado 2 cámaras y se han superpuesto una encima de la otra para que aun que el jugador esté justo en medio de los 2 portales se puedan visualizar los 2 sitios a la vez sin que haya ningún tipo de corte.

Imagen del jugador justo en medio del portal.

Imagen 21. Teletransportación jugador

En el escenario de la izquierda está la copia del jugador y un objeto de color rojo, el cual no debe ver por qué el jugador está en la parte derecha del portal y debe estar viendo lo que está visualizando en el escenario de la derecha, el objeto de color gris.

En la siguiente imagen podemos ver el resultado de la técnica aplicada.

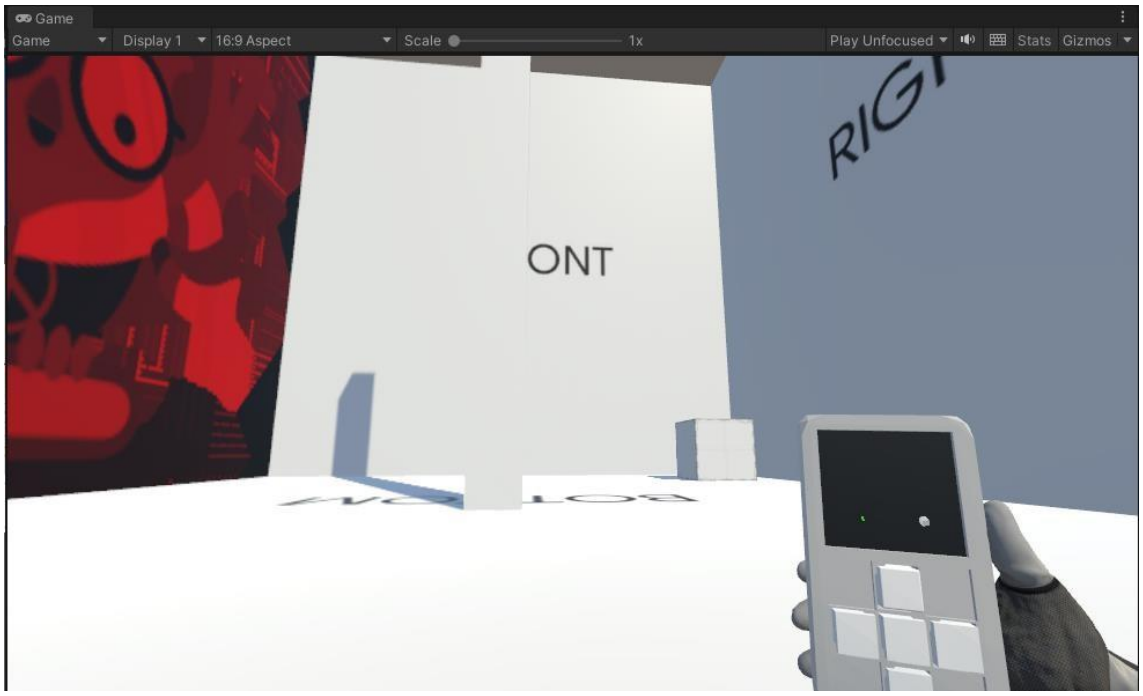

Imagen 22. Vista primera persona del jugador dentro del portal

Teniendo al jugador en este punto del portal hemos puesto una condición la cual es que si el jugador está por detrás del posar que va a travesar unas 0.02 unidades, esté será movido al otro escenario.

Aquí podemos ver como el usuario se ha teletransportado al escenario de la izquierda y ahora se ha duplicado el jugador en el otro escenario.

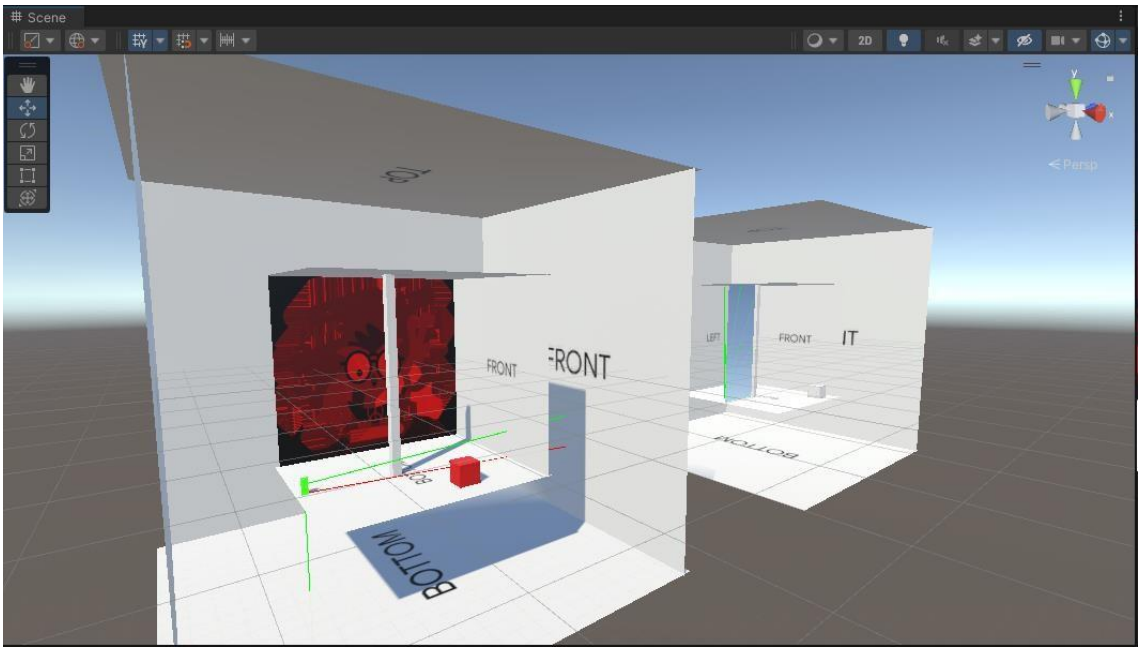

Imagen 23. Jugador teletransportado a escenario izquierdo

Pero todavía no debe estar viendo el objeto rojo, ya que está a la izquierda del portal, y debe estar viendo la copia del otro escenario. Como se puede observar en la siguiente imagen.

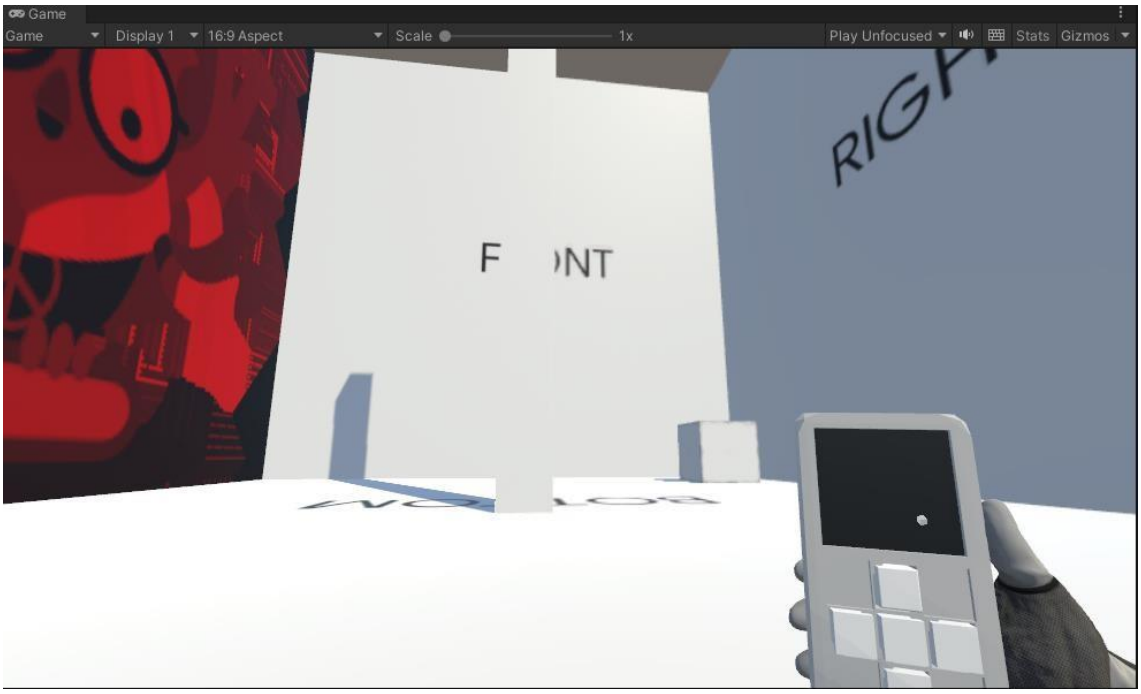

Imagen 24. Jugador en escenario 2

Para visualizar el cubo rojo debe voltear el pilar, de esta manera tiene el portal fuera de su vista (parcialmente) y puede observar el objeto rojo. También puede estar observando el objeto gris al mismo tiempo si tiene el portal dentro del rango de la cámara.

En las siguientes imágenes podemos visualizar como el jugador está en el escenario de la izquierda y hay una copia suya en el escenario de la derecha. El cubo rojo se presenta en el escenario de la izquierda y el cubo más grisáceo se presenta en el escenario de la derecha. Según la perspectiva del usuario y la copia de su cámara en el otro escenario debe ver los objetos a la vez.

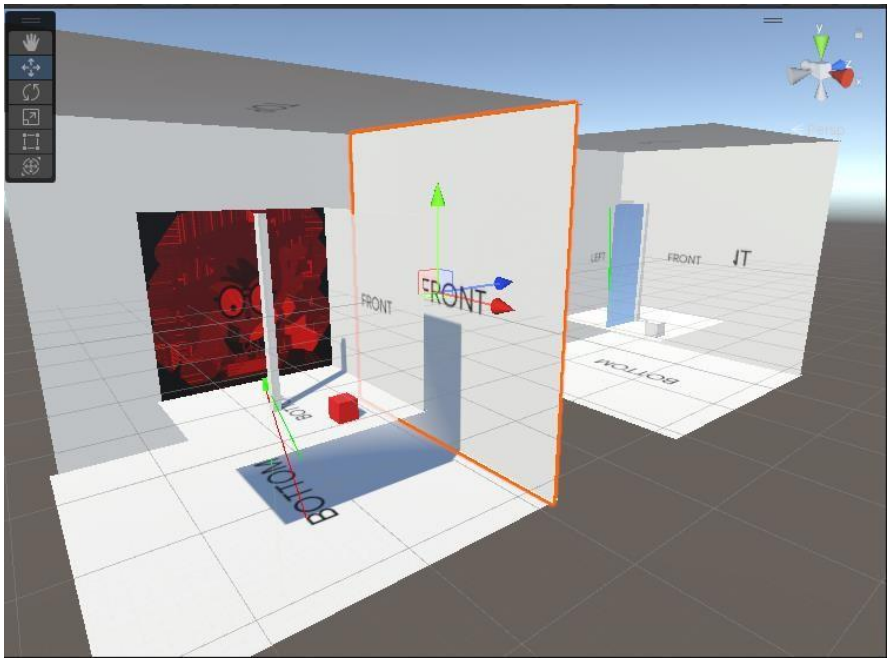

Imagen 25. Jugador volteando el escenario 2

Imagen del mismo caso anterior en distinta perspectiva. El jugador está viendo a través del portal el objeto de color gris.

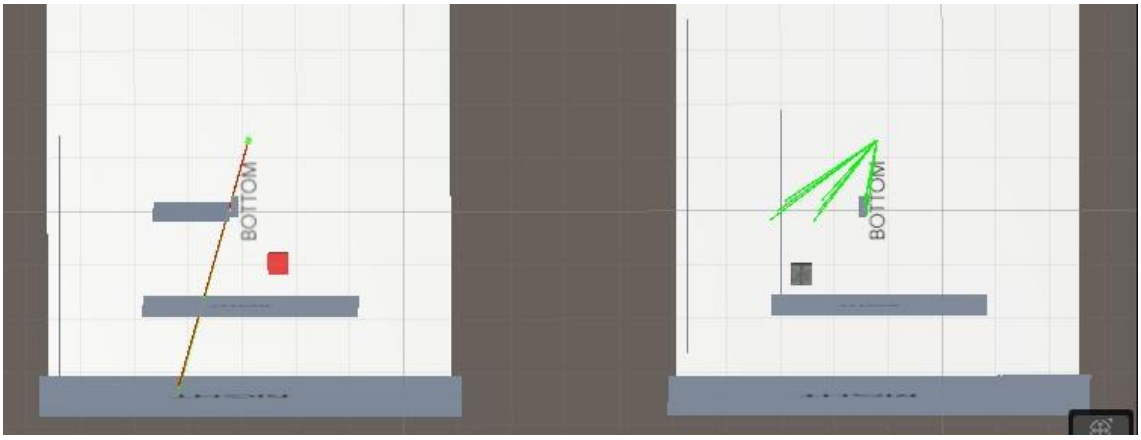

Imagen 26. Vista desde arriba de visualización de los 2 escenarios Imagen resultado de la técnica anterior

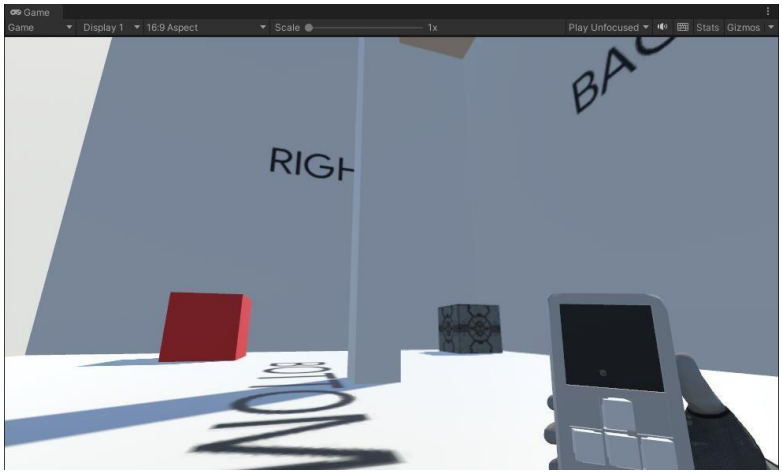

Imagen 27. Resultado del jugador viendo los 2 escenarios

Imagen extra de otra situación en la que los objetos están en la misma posición prácticamente, pero en escenarios distintos. Usando la técnica del pilar para ocultar el corte del portal y simular que hay un espacio no euclídeo en ese punto.

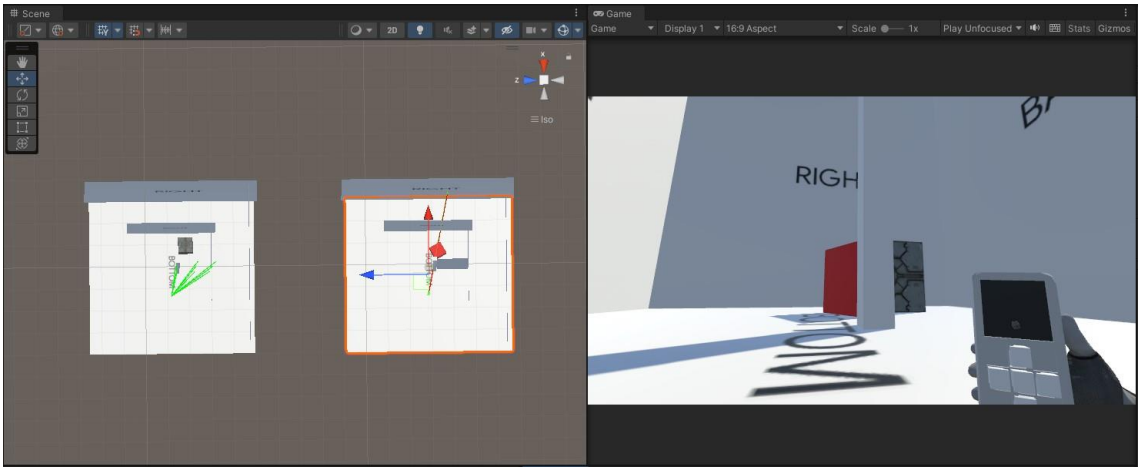

Imagen 28. Visualización de los objetos en distintos escenarios

3.4.3 Interacción con los objetos del escenario

La interacción de los objetos del escenario tiene la misma estrategia aplicada que el jugador, pero con una ligera diferencia. Teniendo en cuenta que el escenario podrá rotar, significa que los objetos van a atravesar estos portales con cierta velocidad y esta velocidad ha de ser conservada, debido a la teoría de portales la cual indica que si un objeto con cierta energía (sea el tipo que sea) este debe salir por el otro portal manteniendo esta energía.

Para aplicar esta estrategia con los objetos, se ha decidido que mientras el objeto este dentro de la zona de teletransporte, se crea una copia del objeto en el escenario al que se va a teletransportar.

En esta imagen podemos observar como el objeto de color gris está atravesando el portal, pero todavía no se ha decidido en qué lado del portal va a terminar.

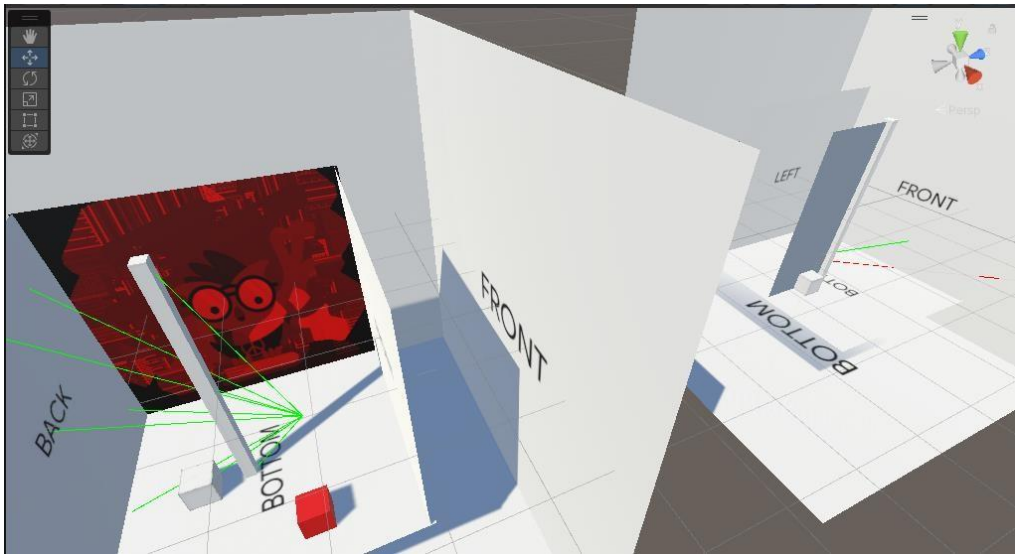

Imagen 29. Objeto teletransportándose de escenario derecho a escenario izquierdo

Se decidirá sobre que escenario se quede el objeto cuando este atraviese el condicional para teletransportarse. Poniendo en el caso que debe atravesar en el eje X y el inicio del eje está en el centro del portal 0. Si el objeto sale por $X > 0$ el objeto debe atravesar el portal de “derecha” a “izquierda”, y si sale por $X < 0$ el objeto debe atravesar el portal de “izquierda” a “derecha”.

Después de haber atravesado el portal la copia que se generó del objeto justo cuando este lo está atravesando, se eliminará. Como se está creando una copia del objeto y todos los movimientos que este está realizando, el objeto abandonara el portal que esté atravesando con la misma energía con la que este entro.

3.4.4 Problema y solución de la interacción de los objetos y jugador en los portales al rotar el escenario

Usando la estrategia de por cual eje estaba pasando el objeto para teletransportarse era buena hasta el punto de que el portal también rotaba cuando rotaba el escenario. Por lo que saber por qué punto ha salido el objeto era muy complicado ya que los ejes cambiaban con cada rotación y también podían hacerlo mientras el cubo estaba rotando.

Para solucionar este problema después de varias ideas inútiles, se decidió crear un eje propio para cada portal. ¿Qué significa esto? Que tenemos un eje "imaginario" que siempre sabremos por qué punto ha entrado el objeto y por cual punto ha salido.

Esta técnica la hemos creado dos puntos geométricos equidistantes en el portal en el escenario.

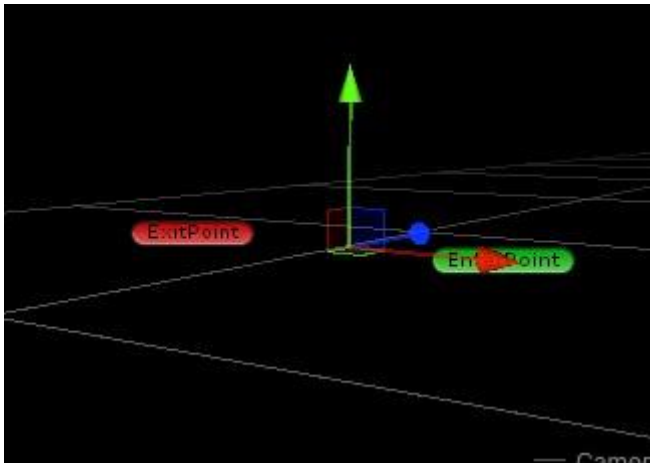

Imagen 30. Puntos de referencia de entrada salida portales

Con estos puntos calculamos la recta que hay entre el punto verde y el punto rojo. Y con esto sabemos cuál es el centro de la recta. Sabiendo el centro de la recta sabemos cuándo el objeto ha atravesado por un lado o por otro.

Por ejemplo, si el objeto entra por el punto verde y sale por el mismo, se queda en el mismo escenario y viceversa con el rojo. Pero si el objeto entra por el punto verde, y cruza el centro de la recta saliendo por el punto rojo, esto significa que el objeto se ha teletransportado, así que se mueve al otro escenario.

Es la mejor técnica de teletransporte para un sistema de portales que puede estar en constante movimiento incluso con objetos que están cruzando el portal al mismo tiempo.

En este enlace se puede observar el funcionamiento del teletransporte de los objetos mientras se rota el escenario: Gif Demostración

3.4.5 Problema y solución 1 de objetos en distintos escenarios

Durante el desarrollo del sistema de portales, hubo un problema que causaba que ciertos objetos que estaban en escenarios distintos se pudiesen ver con la cámara del jugador, cuando realmente esta cámara no debería renderizarlos.

Presentamos el error, teniendo en la siguiente imagen:

- 2 escenarios.
- 2 objetos y el jugador en el escenario numero 1 - Nada en el escenario 2.

Imagen 31. Jugador viendo los objetos en el escenario correcto

Aquí no hay nada extraño por que se ve lo que se debe ver.

Teniendo en la siguiente imagen, el jugador ha atravesado el portal:

- 2 escenarios.
- 2 objetos en el escenario numero 1 - El jugador está en el escenario 2.

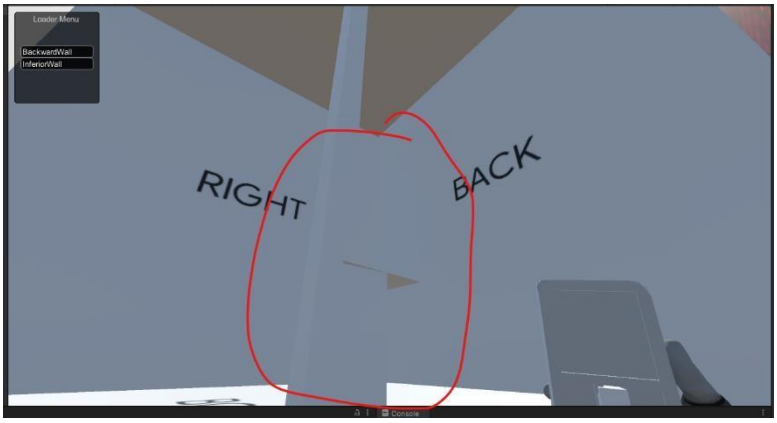

Imagen 32. Jugador viendo los objetos en el escenario incorrecto

El jugador puede ver los objetos que están en el escenario número 1, este error es causado por las 2 cámaras para evitar como el jugador atraviesa el portal. 1 cámara renderiza el escenario normal y otra cámara renderiza la visualización de los portales para que no haya cortes en el juego cuando los portales se atraviesan. Está metodología crea un bug que hace que todos los objetos por detrás del portal se puedan visualizar, aunque no estén físicamente delante.

Para solucionar esto fue un tanto complejo, por lo que tuve que tirar de papel y lápiz y dibujar el problema y empezar a ver soluciones.

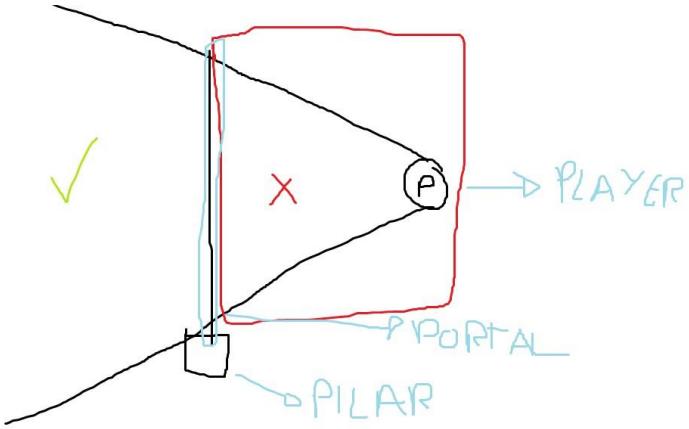

Imagen 33. Representación de lo que no debe ver el jugador estando en distintos escenarios

En el boceto de la imagen anterior observamos que el jugador está situado detrás del portal y dibujamos unas líneas desde el jugador (P) hasta el portal cubriendo todo lo que puede ver el jugador.

He dibujado con un cuadrado rojo, todo lo que podría estar en el otro escenario y lo que no se debe ver desde el escenario actual sobre el cual está el jugador.

Para resolver esto principalmente se planteó crear un shader para que la cámara del jugador solo renderice por detrás de donde está el portal, es decir, lo que ve el jugador en el otro escenario.

Según los problemas que observé durante el desarrollo del shader, vi que era prácticamente imposible o aun no tenía las capacidades suficientes para resolver el problema.

Así que opte por una solución menos compleja y un poco “campesina”, la cual consistía en trazar 8 puntos desde la COPIA del jugador hasta distintos puntos del portal para detectar los objetos que se cruzaban en el camino y una vez detectados poder hacerlos invisibles.

En esta imagen podemos ver como la copia del jugador tiene 8 puntos que salen del hacia el portal en distintas posiciones. Si alguno de estos rayos capta un objeto o cualquier cosa debe hacerse invisible.

Tiene que ser invisible y no desactivarse porque, aunque el objeto no se vea, este debe seguir con sus movimientos para resolver el puzle.

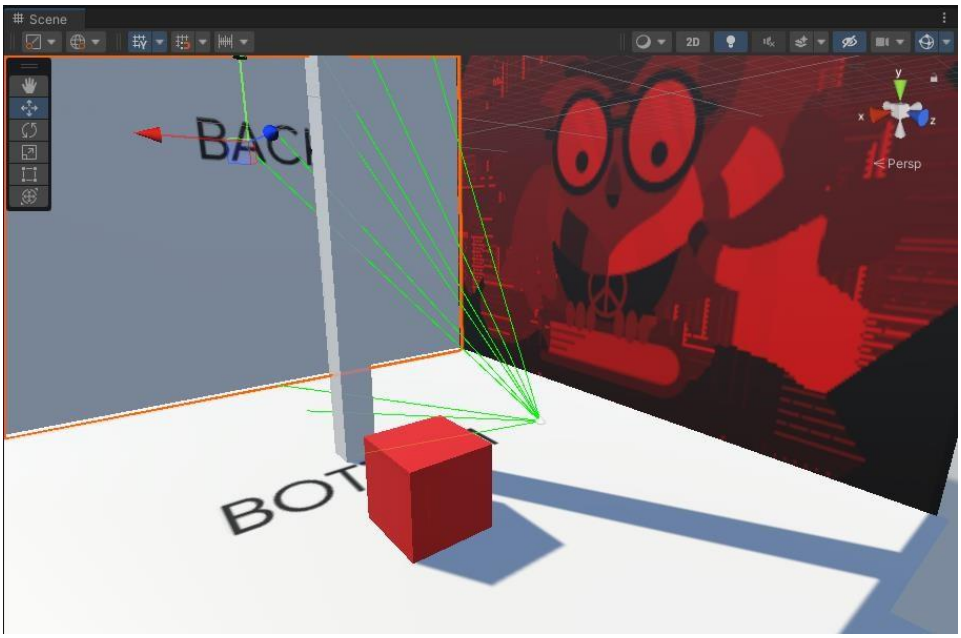

Imagen 34. Gyzmos apuntando a distintos lugares del portal

En esta imagen con el objeto seleccionado en el inspector podemos ver que se ha hecho invisible al entrar en la trayectoria de uno de los rayos.

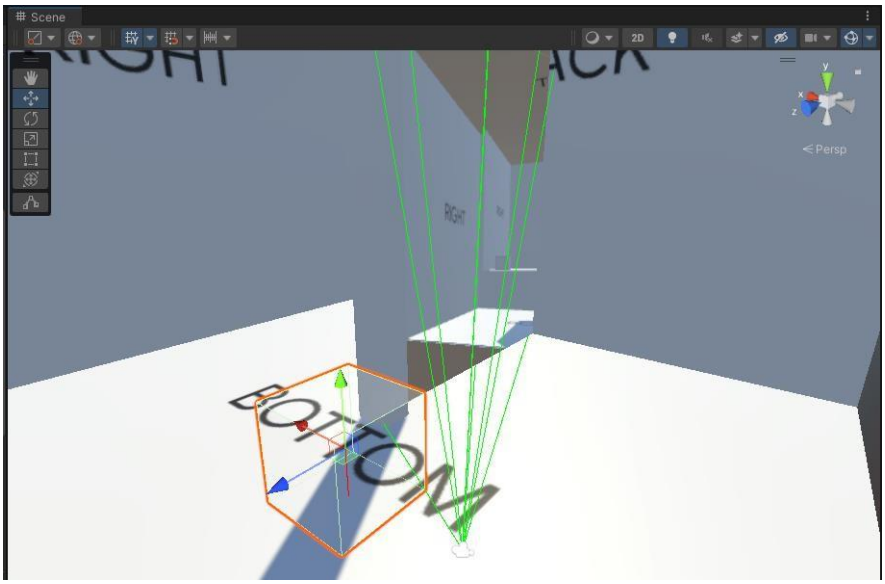

Imagen 35. Objeto invisible al entrar en contacto con gyzmos

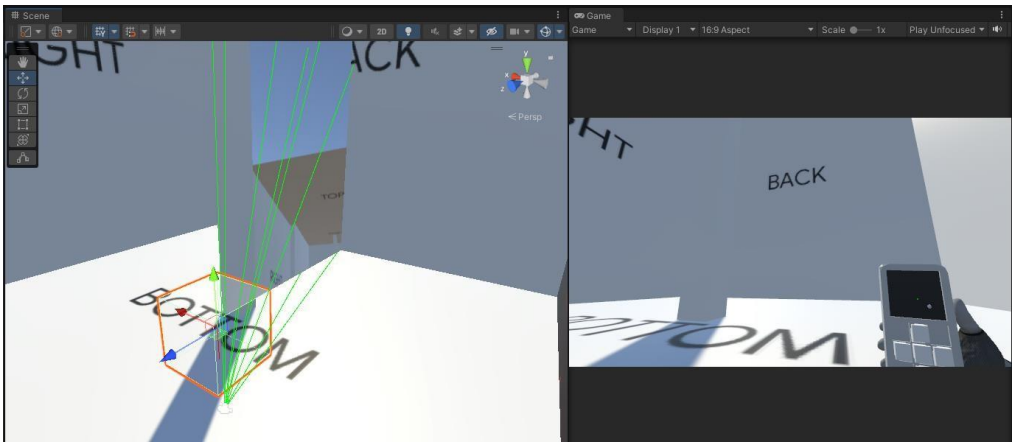

Imagen 36. Resultado en el videojuego

3.4.6 Solución 2 (Versión mejorada) de objetos en distintos escenarios

Durante el desarrollo de la técnica anterior se estuvo observando un par de fallos que podían resultar molestos para la jugabilidad del jugador.

El **primer fallo** detectado era la forma de aplicar la textura cuando la cámara del escenario representando al jugador en un cubo distinto, se superponía sobre un objeto que estaba sobre en ese mismo escenario. Lo que ocurría es que se mostraba una especie de sobra medio transparente, lo cual daba a entender que había un objeto invisible en el escenario y podía confundir al jugador.

La solución al problema anterior fue dejar de aplicar un cambio de textura cuando una cámara representada por el jugador en el otro escenario se ponía delante del cubo, y lo que se ha aplicado ahora es desactivar el "MeshRender" lo cual desactivaba la textura del objeto y aún se podía mover por el escenario.

No se puede desactivar el objeto en el escenario porque dejaría de reaccionar a los movimientos de rotación del cubo.

El **segundo fallo** fue que al usar los rayos para detectar los objetos que había delante es muy costoso para Unity y en general para un ordenador poco potente, también contenía el fallo de que había ciertos puntos ciegos en los que el objeto para ocultar podía quedar justo entre uno de los rayos y estos no tocarlo por lo que no se haría invisible.

La solución a estos dos problemas anteriores viene dada por usar "triggers" en distintas zonas del escenario. Como queremos que los objetos de otros escenarios no se visualicen en la cámara del jugador, aunque se vean en las cámaras que representan al jugador en los otros escenarios, solo cuando estas cámaras estén por detrás del portal en cuestión en el que se ha teletransportado, lo que debemos hacer es que cuando una de estas cámaras esté por detrás el portal en cuestión, los objetos que estén en esa zona se harán invisibles.

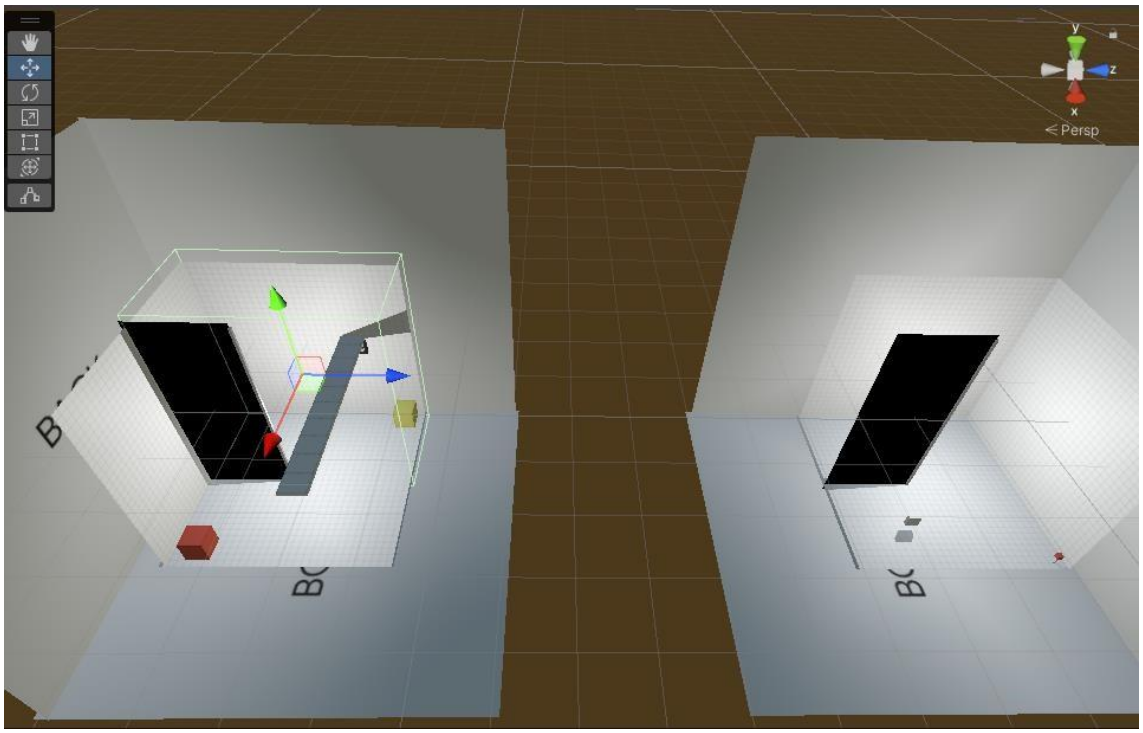

Imagen 37. Seleccionado trigger de ocultación de objetos

En la anterior imagen podemos ver 1 de los “triggers” de bordes de color verde, que lo que hace es detectar si hay alguna cámara copiando la posición del jugador en ese cubo, y si la detecta, ocultará todos los cubos que estén dentro de esta caja de bordes verdes.

En la siguiente imagen podemos ver al jugador en el cubo de la derecha y veremos cómo se oculta el cubo rojo, ya que hay una cámara copiando la posición del jugador en el cubo de la izquierda.

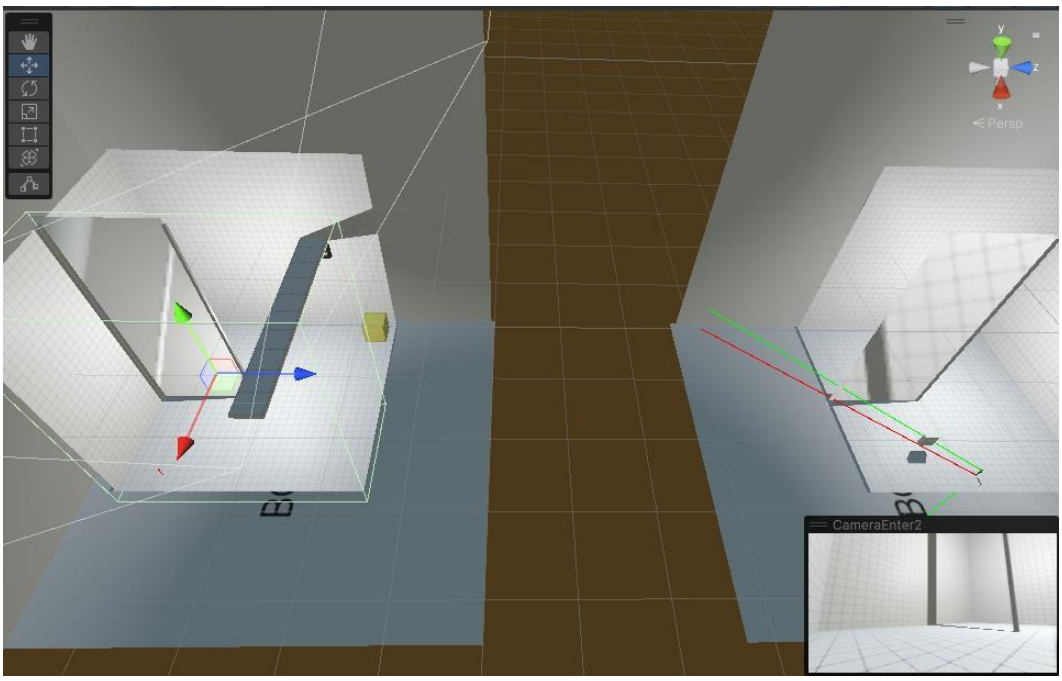

Imagen 38. Objeto invisible cuando una cámara contacta con el trigger

El objeto amarillo no se ocultará ya que si el jugador mira a través del jugador esté se debe ver.

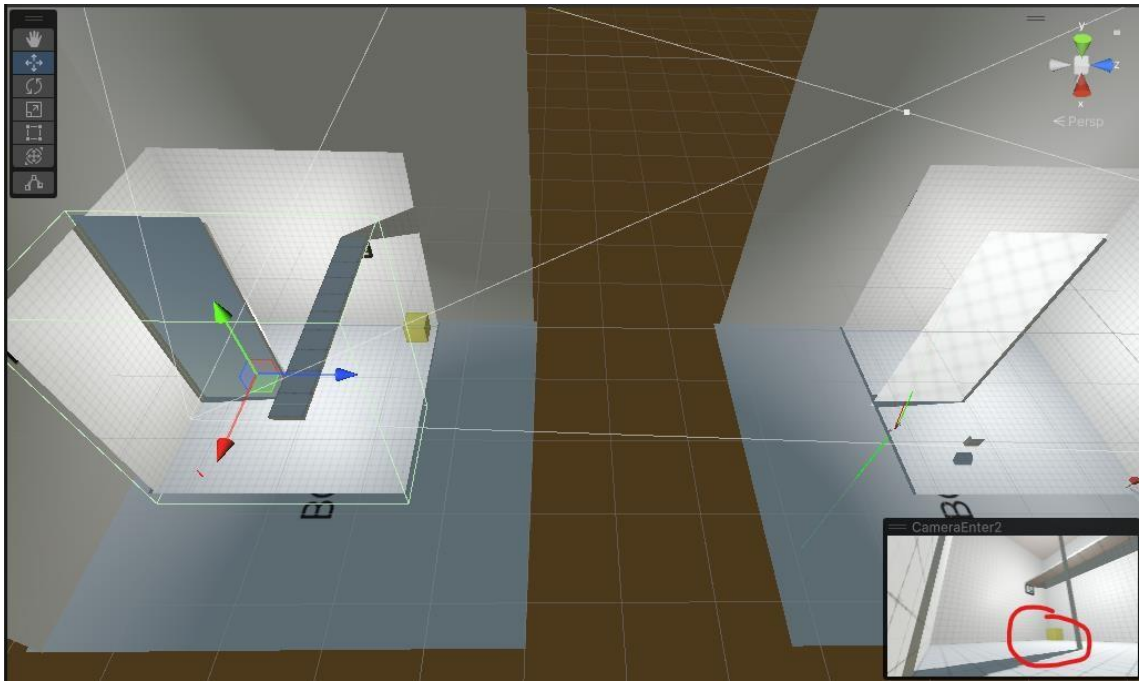

Imagen 39. Muestra de objeto amarillo no ocultado porque no está dentro del trigger

Si el jugador pasa al cubo con los 2 objetos, los 2 se volverán visibles ya que él se detecta que el jugador está en el cubo.

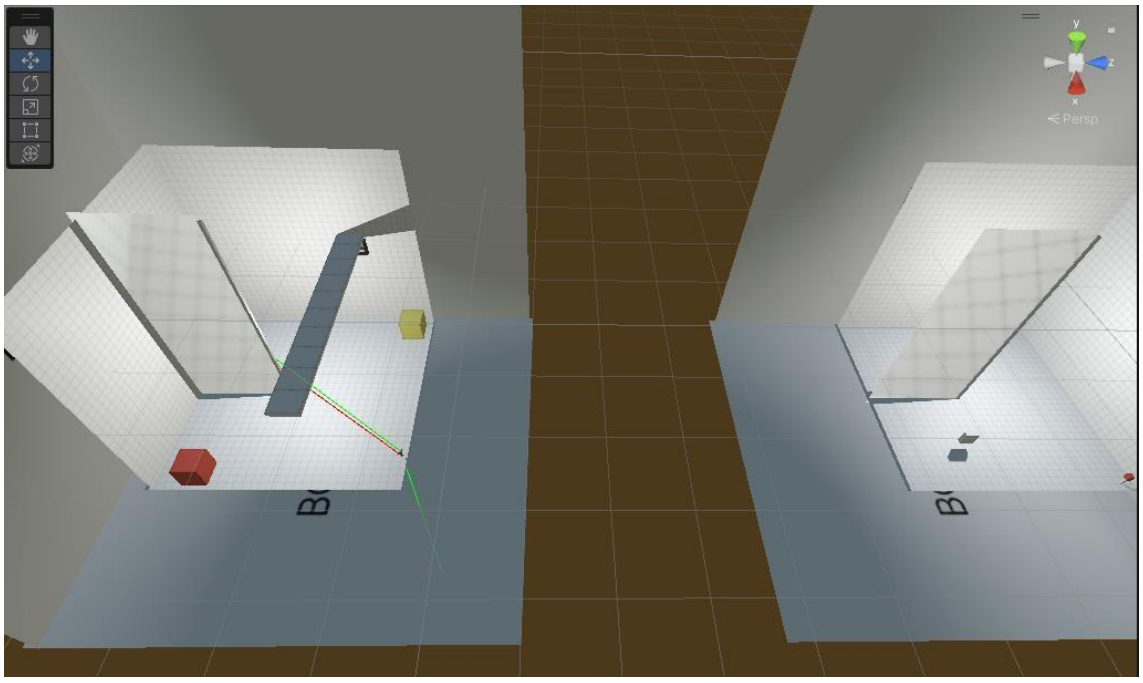

Imagen 40. Objetos no invisibles al entrar el jugador en el escenario

3.5 Figuras imposibles

Para la implementación de visualización de figuras imposibles desde la cámara del jugador para nuestro videojuego en concreto, solo se ha necesitado duplicar el escenario principal en todos los que sea necesario para dificultar el nivel.

El mínimo de escenarios duplicados es de 3.

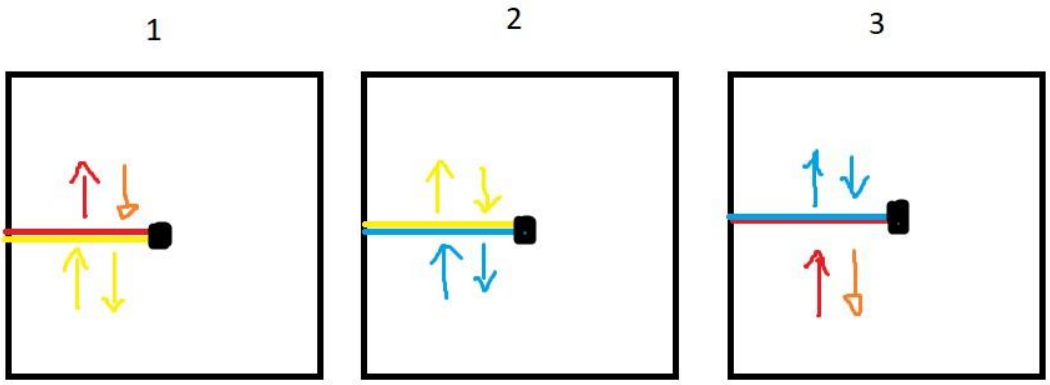

Imagen 41. Prototipo grafico de funcionamiento de portales con 3 escenarios

En la imagen anterior podemos ver una representación de los 3 escenarios unidos por portales. Para la creación de las figuras imposibles es esencial que los escenarios sean una copia exacta, no puede tener un tamaño distinto.

Para hacer un recorrido de la imagen anterior, si el jugador entra por el portal rojo en el escenario 3, debe salir por el escenario 1. Al dar la vuelta y entrar por el portal amarillo, debe aparecer en el escenario número 2 y salir por el portal amarillo. Y si sigue dando la vuelta y entra por el portal azul en el escenario 2, este debe terminar en el escenario por donde empezó saliendo por el portal azul. De esa manera creamos un bucle que crea la ilusión de que es infinito.

El punto negro de los escenarios representa el pilar. Un elemento esencial para delimitar el portal de una forma muy disimulada y que permite al usuario dar vueltas.

Está es la principal técnica de crear espacios no euclídeos en el videojuego, sin embargo, podemos situar los portales en diferentes sitios y diferentes posiciones para causar esté mismo efecto. Incluso podemos crear niveles en el que haya 2 vías diferentes de escape. Como crear 5 escenarios y usar 2 de ellos para enlazarlos a un bucle y otros 2 para otro bucle, y utilizar el último escenario para unir estos 2 bucles y crear un punto de inicio.

Ver Sección 3. Portales

3.5.2. Interacción con los objetos (Visual)

Para crear la simulación de que los objetos están atravesando el portal, desde un principio se ha decidido como solución la aplicación de un shader que corta los objetos por cualquiera de los ejes (solo visualmente).

Para crear este efecto hemos usado el mismo shader, pero creando 4 materiales diferentes para cada uno de los casos:

- Caso 1: Cuando el objeto entra por el portal (Se corta por detrás del portal)
- Caso 2: Cuando la copia del objeto sale por la parte inversa del portal (Se corta por delante del portal)
- Caso 3: Cuando el objeto entra por la parte inversa del portal (Se corta por delante del portal)
- Caso 4: Cuando la copia del objeto sale por el otro portal (Se corta por detrás del portal).

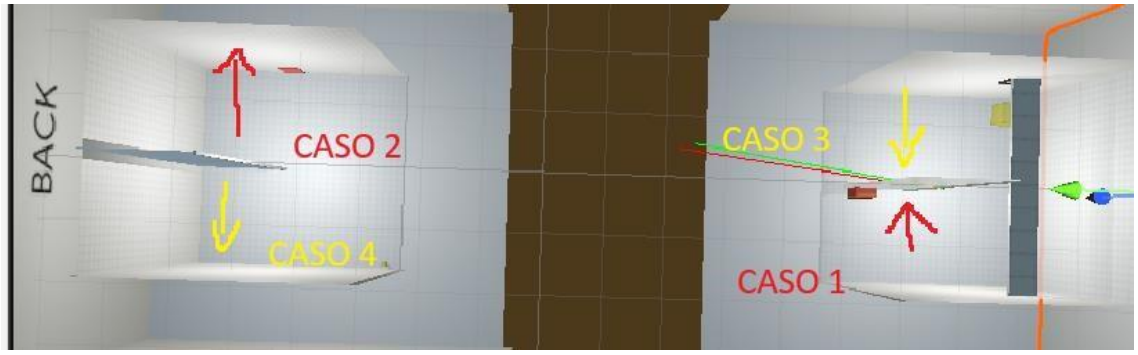

Imagen 42. Explicación corte objetos teletransportación desde portal

Para detectar cuando hacer el corte, se usan “triggers” en las entradas y salidas de los portales. Vista en la imagen siguiente.

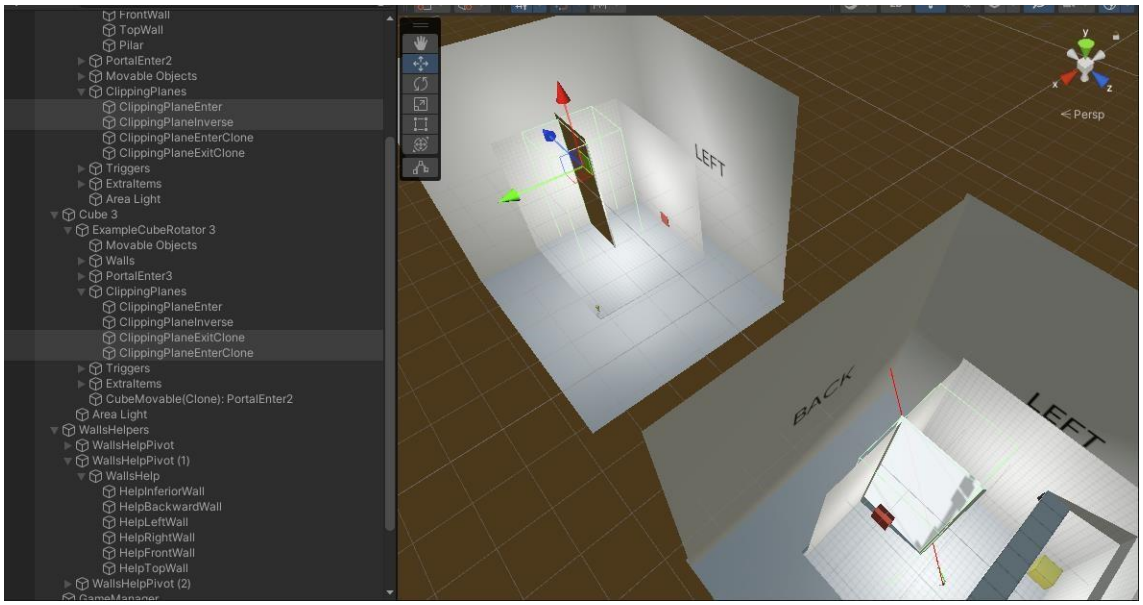

Imagen 43. Muestra triggers corte objetos

Para el caso 1: El jugador no va a ver el objeto desde la otra parte del portal.

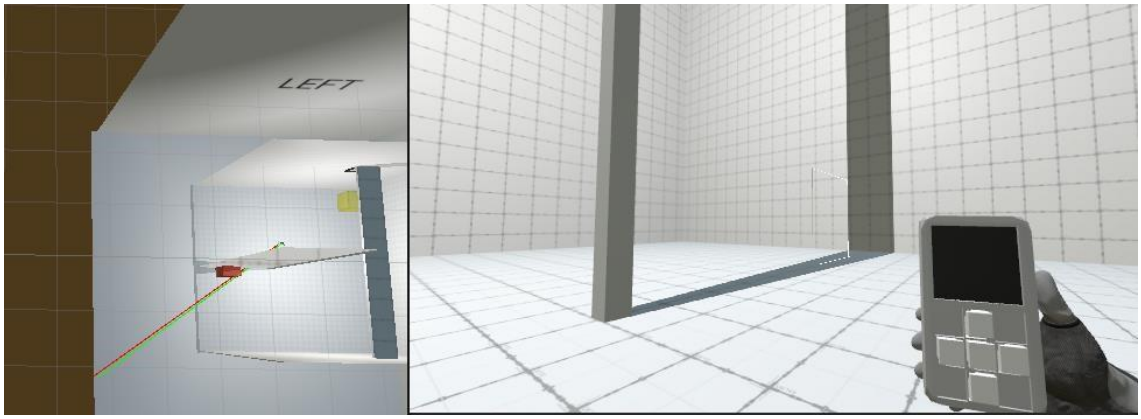

Imagen 44. Resultado corte objeto desde detrás del portal

Para el caso 2: El jugador debe ver el objeto original y la copia, si está en la parte del portal donde se puede ver la copia del objeto.

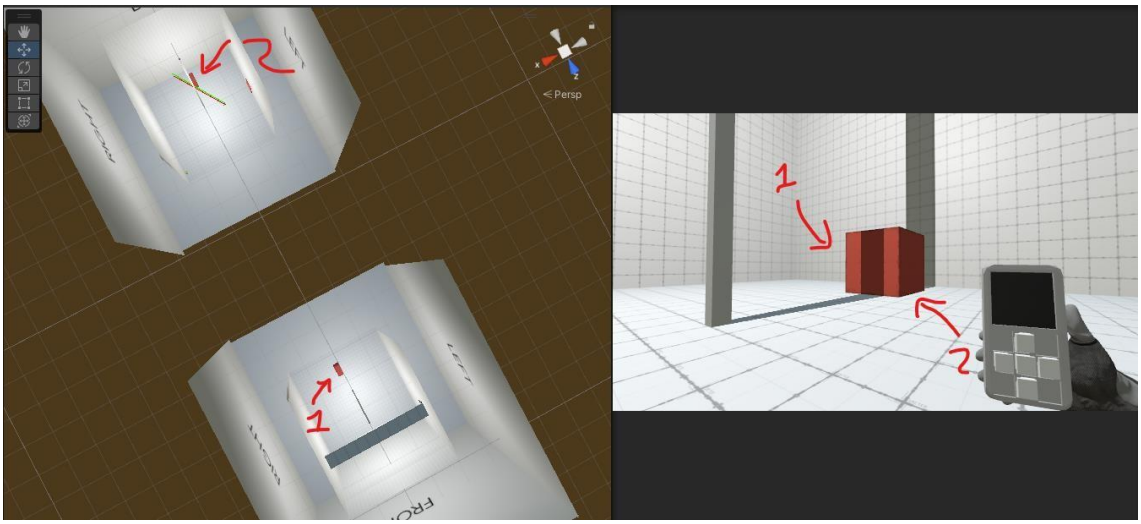

Imagen 45. Resultado de visualización del objeto teletransportándose

Si el jugador se sitúa detrás del portal donde se ha copiado el objeto, no debe ve ver ninguna de las partes del objeto.

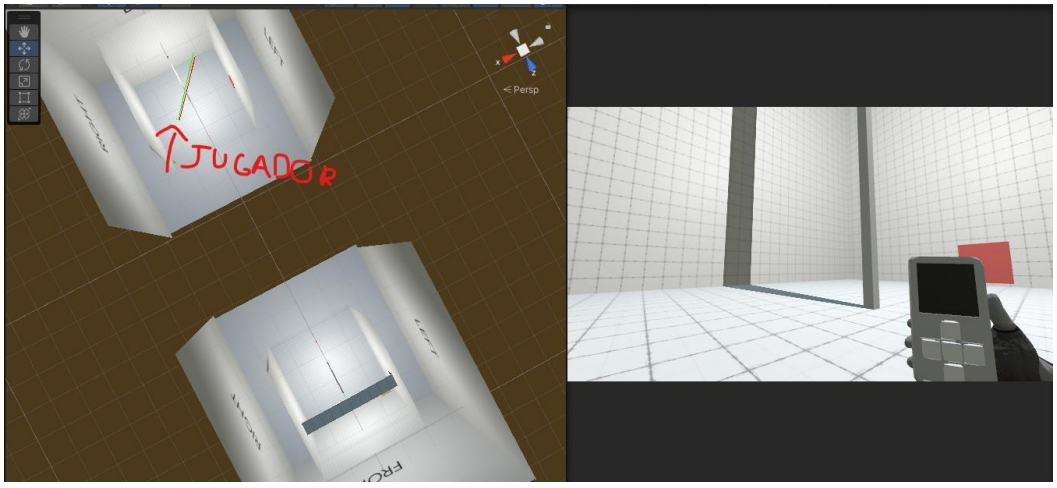

Imagen 46. Resultado del corte en el portal al que se va a teletransportar

Lo mismo aplica para los casos 3 y 4 pero por la parte inversa del portal.

3.5.3. Interacción con los objetos (Físicas)

Para que los objetos interactúen correctamente mientras atraviesan los portales independientemente de en el lugar en el que se encuentren es necesario cortar o modificar los colliders de los objetos que se están teletransportando en ese momento.

En primera instancia la idea fue de cortar los colliders como se estaba haciendo de manera visual, pero por desgracia esto no es posible en Unity. La segunda opción era reduciendo el tamaño de los colliders por la zona del portal en la que estaba siendo teletransportada, pero esto sería un desarrollo muy extenso para la demo además de todos los problemas que causaría a nivel de CPU, ya que nunca sería exacto al 100%.

Por lo que la solución definitiva después de varias ideas viene dada por una función de las físicas de Unity, “Ignore Colliders”.

Esta función nos permite ignorar 2 colliders entre ellos por lo que las físicas no actuaran cuando se superpongan los colliders aun que estos no se puedan atravesar.

Visto de la siguiente forma, al tener desarrollado un sistema de ocultación y cortes de objetos en base a la posición de objetos podemos usar este mismo sistema, pero aplicado a nivel de físicas.

- Caso 1: Cuando el objeto entra por el portal (Se ignora el collider, solo si el jugador está por detrás)
- Caso 2: Cuando la copia del objeto sale por la parte inversa del portal (Se ignora el collider, solo si el jugador está por delante)
- Caso 3: Cuando el objeto entra por la parte inversa del portal (Se ignora el collider, solo si el jugador está por delante)
- Caso 4: Cuando la copia del objeto sale por el otro portal (Se ignora el collider, solo si el jugador está por detrás).

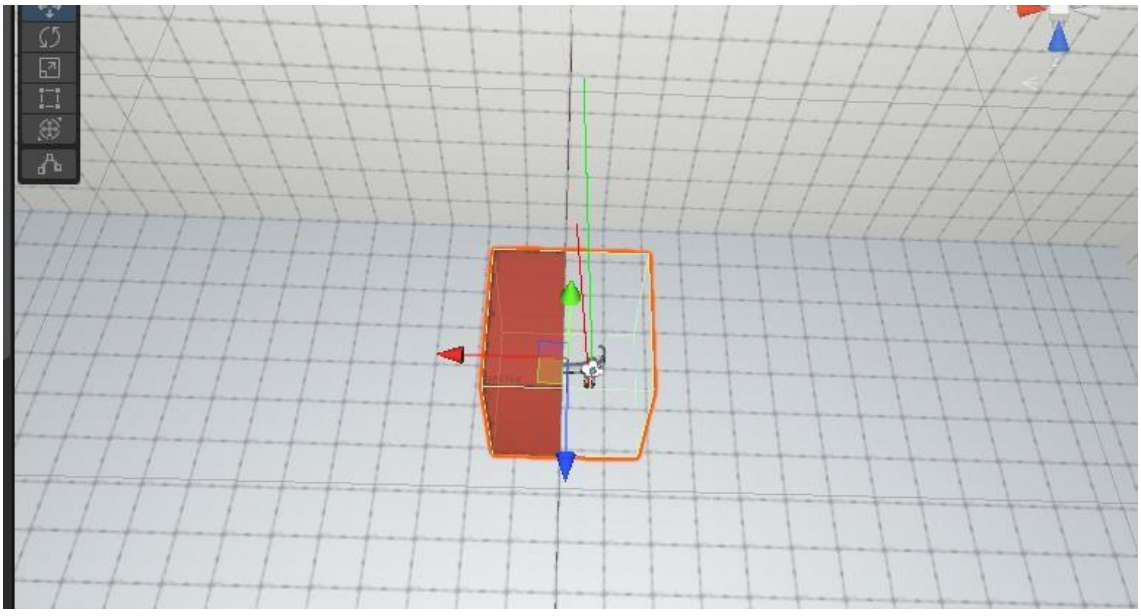

Imagen 47. Caso 1

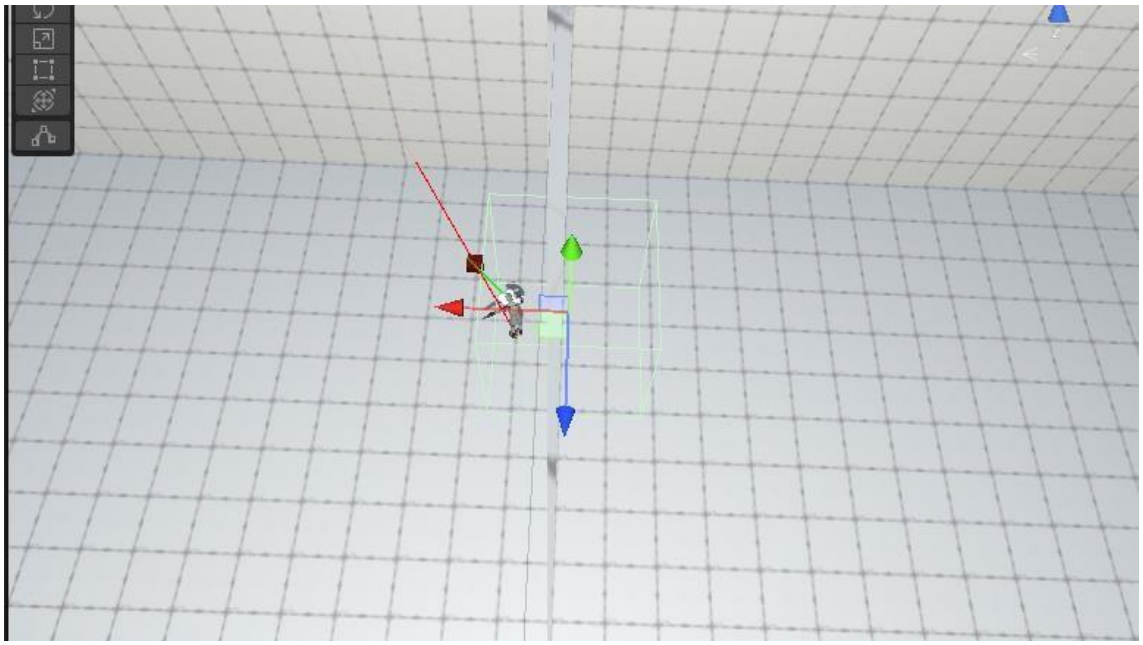

Imagen 48. Caso 2

3.6 Mecánicas secundarias

Tenemos diversas mecánicas secundarias:

- Jugador cambia de paredes y el sentido de la gravedad:

Con esta mecánica podemos hacer que el jugador al pulsar un botón sea capaz de cambiar de "suelo" y hacer que el sentido de la gravedad cambie hacia el nuevo suelo que se ha elegido.

Los objetos siguen la nueva gravedad que ha elegido el jugador, y todos los objetos caerán en el suelo en el que esté el jugador. Esto nos permite cambiar el sentido de la gravedad sin tener la necesidad de rotar el cubo.

- Botones pulsantes por objetos del escenario

Son botones en el suelo del escenario que son pulsados cuando un elemento del escenario cae sobre ellos, estos botones permiten accionar diversos elementos como abrir puertas, crear nuevos elementos en el escenario, accionar plataformas, etc.

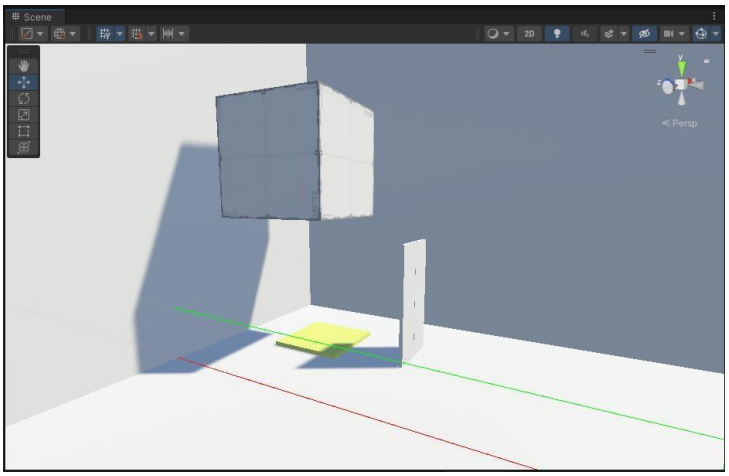

Imagen 49. Objeto antes de accionar el botón amarillo

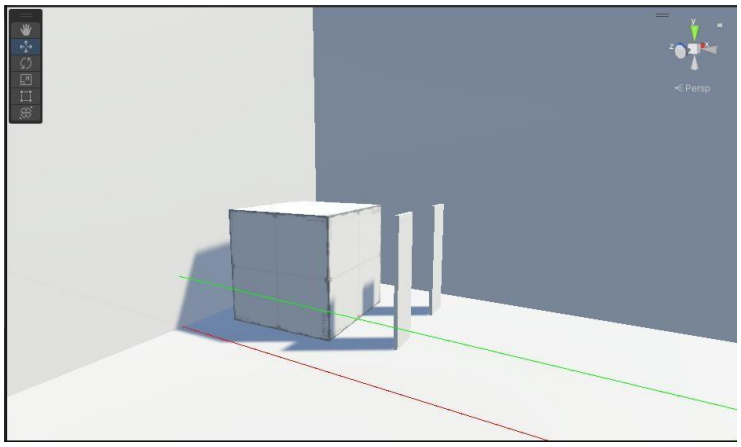

Imagen 50. Objeto accionando el botón amarillo

- Botones pulsantes asociados a colores.

Consiste en la misma mecánica que la anterior, pero con una ligera diferencia, en este caso pude haber un botón accionador en el escenario de cierto color, y para ser accionado, debe ser pulsado por un elemento dinámico del escenario que tenga el mismo color que este elemento.

Si el elemento no tiene el mismo color que el botón accionador, este botón no ejecutará ningún tipo de acción,

- El jugador puede parar el tiempo de ciertos objetos con botones.

Con esta mecánica permitimos que el usuario sea capaz de parar el tiempo de ciertos objetos a través de accionadores.

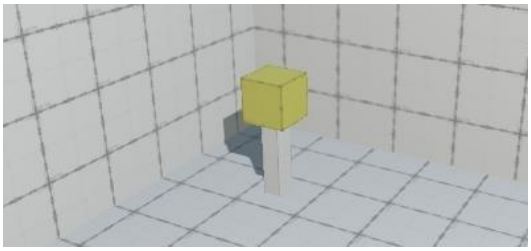

Imagen 51. Botón des/activador gravedad

Un ejemplo de nivel con esta mecánica sea situar un botón que abra una puerta y otro botón para parar el tiempo. El botón accionador de la puerta será pulsado por un elemento del escenario, pero para acceder a la puerta la cual se va a abrir sería necesario rotar el escenario por lo que el cubo accionando la puerta abierta se cerrará. Por eso usamos este botón para parar el tiempo para que el objeto se quede accionando la apertura de la puerta durante un periodo de tiempo para completar el nivel.

- Mover objetos estáticos por el escenario.

Con esta mecánica podemos mover objetos que no son afectados por ningún tipo de gravedad, esto nos permite poder poner los botones accionables anteriores en posiciones “antinaturales” para el dominio del escenario. Ejemplo:

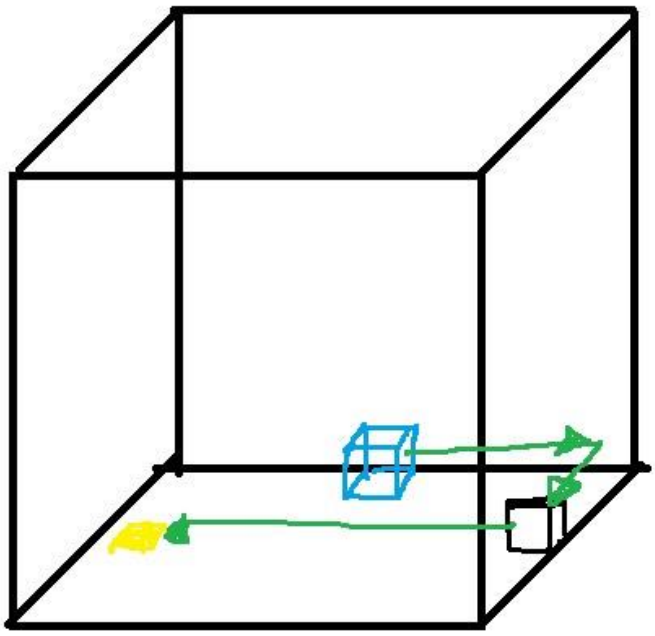

Imagen 52. Ejemplo de uso objetos movibles del escenario

Para este caso tenemos 2 posibilidades de situar el elemento estático (color negro) en 2 sitios distintos, 1 al lado del botón para ser más directo y otro en frente. Depende de la posición que elijamos el objeto dinámico (color negro) debe chocar con este objeto estático y volver a rotar el escenario para caer sobre el botón (representado como amarillo).

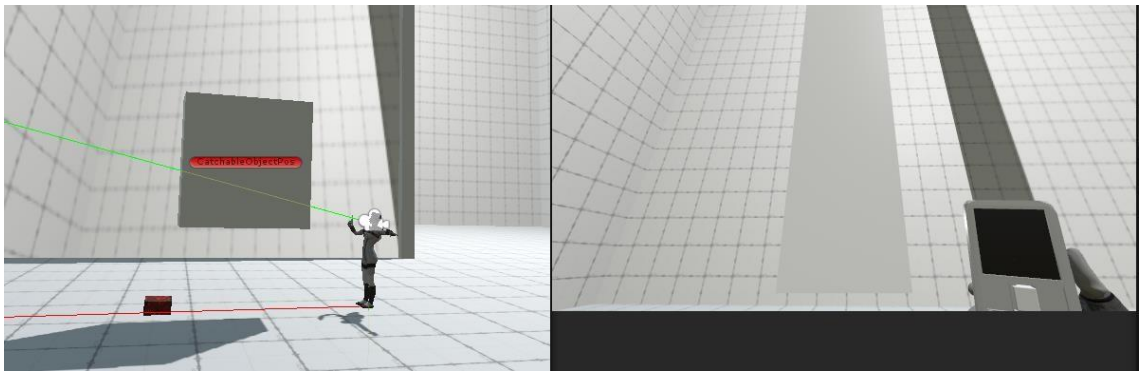

Imagen 53. Jugador cogiendo objeto

4. Resultados

4.1. Objetivos finales del juego

El objetivo del videojuego es poder resolver todos los puzles que se plantean haciendo uso de todas las mecánicas desarrolladas anteriormente.

Se pueden resolver los puzles mediante el uso de:

- Un sistema propio de rotación de escenarios que pueden rotar sobre su propio eje y el jugador puede elegir en qué dirección rotarse.

- Un motor de físicas propio que permite que ciertos elementos del escenario reaccionen a la rotación del escenario sin que estos roten sobre si mismos.

- Un sistema de controles que permite al usuario hacer acciones sobre el escenario de forma intuitiva y sin interrumpir la resolución del puzle.

- Una simulación de figuras no euclídeas / imposibles mediante el uso de portales para la simular la estancia del jugador en 2 salas distintas a la vez.

- Unas mecánicas secundarias como activación de botones con los elementos del escenario y accionadores que permiten desactivar la gravedad de ciertos elementos.

4.2. Controles del jugador (Teclado)

Explicación de los movimientos e interacciones que tiene el jugador sobre el escenario y objetos.

4.2.1. Movimiento

Los controles de movimiento del jugador sobre el videojuego son los básicos que se aplican en todos los videojuegos, usando las teclas “W, A, S, D”.

Movimiento hacia adelante: “W”.

Movimiento lateral izquierdo: “A”.

Movimiento lateral derecho: “D”.

Movimiento hacia atrás: “S”.

4.2.2. Coger elementos

La tecla más utilizada para coger elementos en los videojuegos es la “C”, por lo que por estándar se ha decidido utilizar esta tecla también.

Se usa esta tecla ya que proviene de la inicial de la palabra en inglés “catch” que significa “coger” en español.

4.2.3. Cambiar de pared

Para cambiar de pared con el jugador y también cambiar el sentido de la gravedad en esa dirección se utilizará la tecla “X”.

El usuario al acercarse a una pared aparecerá un indicador que le indicará esta tecla.

4.2.4. Rotación del escenario

Estos controles ya son más independientes del propio videojuego, por lo que para tener una interacción sencilla con el mismo.

Los controles de rotación en base a la pared que está observando el jugador vienen dados por:

Movimientos horizontales:

Rotación 90º hacia la derecha: “E”. Se ha decidido está tecla porque en algunos juegos se usa como movimiento lateral derecho por lo que es más intuitivo.

Rotación 90º hacia la izquierda: “Q”. Se ha decidido está tecla porque en algunos juegos se usa como movimiento lateral izquierdo.

Movimientos verticales:

Rotación 90º hacia delante, “R”. Decisión de usar está tecla porque está encima de la “F”, la cual se usa para rotar hacia atrás.

Rotación 90º hacia atrás, “F”. Decisión de usar está tecla porque está debajo de la “R”, la cual se usa para rotar hacia delante.

4.3. Niveles propuestos

Para la presentación de la demo se ha decidido desarrollar 4 niveles distintos en los que en cada uno de ellos se va a presentar una mecánica distinta, mientras se va a avanzando en el videojuego, las mecánicas aprendidas en los niveles anteriores se aplicarán también en niveles posteriores junto con nuevas mecánicas.

Nivel 1

En el nivel 1, tenemos, un escenario en el que el jugador todavía no puede rotar el cubo. El jugador aparecerá en el escenario, con una especie de maniquí delante de él. El maniquí va a estar encerrado dentro de una capsula hasta pasados 2 segundos, justo después el jugador podrá coger el traje y simular que se lo ha puesto. Con el traje puesto el jugador podrá cambiar de pared. No habrá ningún objeto.

Nivel 2

En el nivel 2, tenemos un escenario en el que hay un botón que se acciona cuando un objeto del escenario se sitúa encima de este. Este botón se accionará con cualquier elemento de cualquier botón. Una vez pulsado este botón se abrirá la puerta que permitirá avanzar al usuario al nivel 3.

Para que sea posible accionar este botón, se pondrá un objeto que el jugador pueda coger para poder moverlo hacia el botón, situándolo de manera que el objeto que se mueve por el escenario choque con este y de esta manera se quede quieto encima del botón.

En este escenario también habrá como una especie de teléfono en el que se indicará al jugador con al tenerlo en la mano, podrá rotar el escenario en cualquiera de sus propios ejes mirando a cualquiera de los laterales del escenario.

Nivel 3

En el nivel 3, tenemos el mismo escenario que el anterior, pero con 2 objetos que interactúan con el sistema de gravedad, uno con color y otro sin color asignado. El objeto con color podrá accionar el botón del mismo color que hay en el escenario.

En ese mismo escenario habrá un activador que lo que hará será detener la gravedad del cubo colorido. Este activador es necesario pulsarlo para que el objeto no se mueva mientras accione el botón, ya que la salida de la puerta al siguiente nivel, la cual se abrirá cuando este botón este accionado, estará en una pared superior para obligar al jugador a cambiar el sentido de la gravedad de los objetos y cambiar de pared.

Nivel 4

En el último nivel tenemos el más complejo de todos en el que incluimos todas las mecánicas anteriores y la más compleja de desarrollar de todas, las figuras no euclídeas.

En este nivel se presentarán 3 escenarios distintos, en el escenario 2 habrá 2 objetos que se pueden mover con la gravedad, 2 de mismo color. En este escenario también estará la puerta por donde el jugador va a salir para completar el juego.

En el escenario 3 habrá 1 accionador, que corresponderán a los objetos que se mueven por estos 3. En estos escenarios también habrá un activador para desactivar la gravedad de cada 1 de los objetos.

Cuando el jugador resuelva el puzle se abrirá la puerta, permitiendo así completar el videojuego.

Video de demostración de paso de los niveles y resultado de la demo:

https://www.youtube.com/watch?v=EfMjwTTp4mw&ab_channel=YerayCandel

5. Conclusiones y propuesta de trabajo futura

Para concluir en el desarrollo de este proyecto podemos decir que el desarrollo de la demo de este videojuego se ha abordado con éxito, desarrollando mecánicas de juego como la rotación del escenario, un sistema de gravedad propio para la resolución de puzles y un sistema de niveles no euclídeos. Durante el proceso del desarrollo se ha explotado la capacidad de explorar los conceptos novedosos y existentes para la creación de este videojuego y crear una experiencia única y desafiante para los jugadores.

La implementación de rotación del escenario y sistema de gravedad propio ha permitido crear desafíos en el juego que van más allá de un videojuego de plataformas o de puzles. Es necesario adaptarse a la física del mundo virtual y escenarios cambiantes para resolver los problemas y de esta forma se fomenta la resolución de problemas y pensamiento de los jugadores a la hora de resolver los puzles.

Además, la implementación de los niveles no euclídeos ha llevado a la creatividad del diseño del nivel a un paso más allá, desafiando las percepciones espaciales de los jugadores. Esta innovación no solo ofrece entretenimiento, sino también incita al jugador a entender cómo es que funciona este espacio no euclídeo y se pueda experimentar a través de un mundo virtual.

En el proceso de desarrollo, se han superado desafíos técnicos de los cuales no hay mucha información en el mundo de la información. Esta superación de los desafíos ha enriquecido la compresión y fortalecido mis habilidades en el mundo de la programación y el diseño de videojuegos. Además, este proyecto sirva para demostrar que todas las ideas que puedas tener en mente las puedes llevar a cabo con esfuerzo y con pasión.

En último punto, este TFG no solo representa mis habilidades de programación, si no las posibilidades que puede ofrecer un motor de videojuegos si sabes darle un buen uso. Al mostrar como la combinación de conceptos innovadores puede transformarse en una experiencia jugable y se espera que este trabajo inspire a otros desarrolladores a seguir explorando límites y aportando nuevas perspectivas al emocionante campo de los videojuegos.

Nada es imposible.

6. Bibliografía

Canal FumetsuHito. (27 de febrero de 2021). Unity - Make Objects Between Camera and Player Transparent (No Shader Manipulation) [Archivo de Vídeo]. Youtube. https://www.youtube.com/watch?v=xMFx9HfRknU&ab_channel=FumetsuHito

Ronja's Tutorials. (2021). *Re:* Clipping a Model with a Plane. https://www.ronjatutorials.com/post/021-plane-clipping/

Canal Mohammad Faizan Khan. (1 de abril de 2022). Unity3d How to Change Pivot | Set Pivot in Unity with offset using a empty parent object [Archivo de Vídeo]. Youtube. https://www.youtube.com/watch?v=cnl6r68y-IE&ab_channel=MohammadFaizanKhan

Canal CodeParade. (14 de agosto de 2018). Non-Euclidean Worlds Engine [Archivo de Vídeo]. Youtube. https://www.youtube.com/watch?v=kEB11PQ9Eo8&t=6s

Canal JayCode. (7 de septiembre de 2020). Non-Euclidean Game using Unity [Archivo de Vídeo]. Youtube. https://www.youtube.com/watch?v=rvAhM9ynbSc&t=140s&pp=ygUZbm9uIGV1Y2xpZGVhbiB3b3JsZCB1bml0eQ%3D%3D&ab_channel=JayCode

Canal Guinxu. (11 de junio de 2020). Creando mi propia versión de Portal en Unity [Archivo de Vídeo]. Youtube. https://www.youtube.com/watch?v=NGeZCAi2bJs&ab_channel=Guinxu

Canal Bala_7. (9 de julio de 2021). Hice PORTAL en UNITY (y fue súper DIFÍCIL 😭😭) [Archivo de Vídeo]. Youtube. https://www.youtube.com/watch?v=NUxRV8a9HXA&t=448s&ab_channel=Bala_7

Canal DA LAB. (22 de enero de 2023). Transparent Material in Unity [Archivo de Vídeo]. Youtube. https://www.youtube.com/watch?v=8x4VnzZIVro&pp=ygURdHJhbnNwYXJlbnQgdW5pdHk%3D&ab_channel=DALAB

Canal LlamAcademy. (19 de julio de 2022). Fade Objects with C# and the Standard URP Shaders | Unity Tutorial [Archivo de Vídeo]. Youtube. https://www.youtube.com/watch?v=vmLIy62Gsnk&t=11s&pp=ygURdHJhbnNwYXJlbnQgdW5pdHk%3D&ab_channel=LlamAcademy

Anexos

1. Tabla ODS

Objetivos de Desarrollo Sostenibles	Alto	Medio	Bajo	No Procede
ODS 1. **Fin de la pobreza.**				X
ODS 2. **Hambre cero.**				X
ODS 3. **Salud y bienestar.**				X
ODS 4. **Educación de calidad.**			X	
ODS 5. **Igualdad de género.**				X
ODS 6. **Agua limpia y saneamiento.**				X
ODS 7. **Energía asequible y no contaminante.**				X
ODS 8. **Trabajo decente y crecimiento económico.**				X
ODS 9. **Industria, innovación e infraestructuras.**				X
ODS 10. **Reducción de las desigualdades**.				X
ODS 11. **Ciudades y comunidades sostenibles**.				X
ODS 12. **Producción y consumo responsables.**				X
ODS 13. **Acción por el clima.**				X
ODS 14. **Vida submarina.**				X
ODS 15. **Vida de ecosistemas terrestres.**				X
ODS 16. **Paz, justicia e instituciones sólidas.**				X
ODS 17. **Alianzas para lograr objetivos.**				X