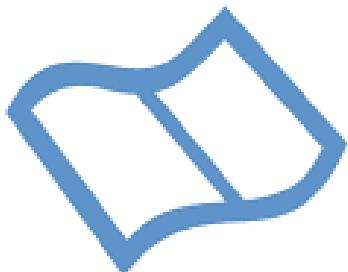


Response:Eve

Documentación



iesperemariaorts

Simón Muñoz Baldó

Índice

Resumen	4
Introducción	5
Objetivos	5
Justificación	6
Estudio económico	7
1. Identificación del emprendedor	7
Capacidad emprendedora	7
Interés en el proyecto	8
2. La idea de negocio	8
Redacción de la idea de negocio	8
La propuesta de valor y objetivo	9
3. Elección de la forma jurídica	9
4. Estudio de mercado	10
Análisis de macroentorno	10
Factores político-legales:	10
Factores económicos:	11
Factores socio-culturales:	12
Factores tecnológicos:	12
Factores medioambientales:	12
Factores internacionales:	13
Análisis del microentorno	13
Competidores	13
Clientes	14
Productos sustitutivos	14
Proveedores	14
Ubicación del negocio	15
Análisis DAFO	15
Plan de marketing	16
Producto	16
Precio	17
Promoción	17
Distribución	18
Plan de producción y recursos financieros	18
Inversiones y gastos	18

Origen de la financiación	19
Ayudas y subvenciones	19
Umbral de rentabilidad o punto muerto	20
Plan económico-financiero	20
Análisis	21
Trámites de constitución	21
Análisis	22
Necesidades del proyecto y planificación temporal	22
Fase 1: Desarrollo de las mecánicas	22
Fase 2: Diseño del personaje y del entorno	23
Fase 3 y 4: Diseño y desarrollo de los niveles	24
Fase 5: Diseño y programación de menús	24
Fase 6: Implementar sistema de puntuación	24
Fase 7: Pulir aspecto visual y sonoro	25
Fase 8: Testear y arreglar bugs	25
Planificación temporal (Diagrama de Gantt)	25
Diagrama de clases	26
-Carpeta Player	27
-Carpeta Tools	28
-Carpeta ToolsComplements	28
-Carpeta Camera	29
Persistencia de datos	30
Entornos de programación	31
Lenguajes y tecnologías que se van a emplear.	31
Implementación	34
Prototipo del proyecto	34
Aspectos relevantes de la codificación	40
Activación de herramientas mediante el apuntado a superficies	40
Despliegue de portales	41
Teletransportación a través de los portales	44
Inversión de la escala de los portales	45
Automatización de los postes eléctricos	47
Pruebas de funcionamiento	49
Manual de despliegue	49
Conclusiones	50
Webgrafía	51

Resumen

Response: Eve es un juego de plataformas 2D en el que el jugador tendrá que atravesar diversos niveles para completar el juego. En cada nivel encontraremos una variedad de obstáculos que superar para completarlos. Estos obstáculos requerirán de habilidad por parte del jugador pero también de inteligencia , puesto que algunas secciones serán más propias de un juego de puzzles que de un plataformas.

El jugador maneja a Eve, un robot del proyecto Response, que como muchos otros de su clase es puesto a prueba en unas instalaciones militares diseñadas para probar sus capacidades. Estos robots están diseñados para adaptarse a su entorno mediante el uso de sensores. Los sensores le comunican al robot los objetos cercanos y estos son capaces tanto de copiar sus propiedades como de manifestar una herramienta que interactúe de forma especial con ellos. Esto le permite a Eve por ejemplo convertirse en un robot hecho de goma o manifestar un imán en presencia de un campo electromagnético. El jugador deberá de aprovechar estas mecánicas para atravesar todos los obstáculos que le esperan.

El juego consistirá en dos niveles en el que se introducirán todas las mecánicas de forma gradual y donde la dificultad irá escalando de igual forma. El objetivo de cada nivel será simplemente llegar hasta la meta y habrá un sistema de puntuación que califique lo eficiente que ha sido el jugador durante dicho recorrido. Luego el jugador podrá comparar las puntuaciones que ha obtenido en cada uno de los niveles con sus amigos o con el resto de jugadores del mundo.

Introducción

Objetivos

El desarrollo de Response:Eve consiste en hacer un videojuego plataformero de móvil y de escritorio dirigido a todo tipo de público y a un precio accesible para cualquiera.

El objetivo del juego es claro: que sea desafiante a la vez que divertido. Para ello, diseñaremos y ofreceremos una gran variedad de mecánicas y de niveles que las aprovechen. Estos niveles irán escalando de dificultad progresivamente a medida que el jugador los vaya completando. Se buscará un equilibrio entre la dificultad de dichos niveles y la habilidad que el jugador medio ha ido adquiriendo con los anteriores niveles. El objetivo es crear una curva de dificultad óptima para evitar que el usuario se sienta abrumado por la dificultad o que se aburra por ser demasiado fácil.

La variedad de mecánicas es otra de las claves del juego. El objetivo es disponer de al menos 5 mecánicas de movimiento distintas. Estas mecánicas se irán introduciendo de forma progresiva en los niveles a medida que el jugador avance en el juego. Esto le dará frescura a la experiencia y tendrá al jugador preguntándose cuál será la siguiente herramienta que le brinde el juego y cómo cambiará la forma en la que completa los niveles.

De nada sirve una gran variedad de mecánicas sin un diseño de niveles a la altura. El objetivo de cada nivel será el de explorar las posibilidades que ofrecen las mecánicas y aprovecharlas para ofrecer una variedad de experiencias al jugador, haciéndole a este preguntarse qué situaciones presentará el juego continuación.

Justificación

He elegido hacer este proyecto debido sobre todo a mi familiaridad con los videojuegos, destacando el género plataformero. Siento que a pesar de la gran abundancia de juegos similares, sobre todo en el mercado móvil, este tipo de juegos suelen ser bastante simples en cuanto a sus mecánicas y su

planteamiento. Como consumidor siempre he querido poder jugar a un juego de plataformas que tuviera la complejidad de un juego de escritorio en el móvil, y todo esto sin la publicidad invasiva a la que tanto nos tienen acostumbrado los juegos de móvil.

El problema de hacer un juego de estas características es que los controles de un juego de escritorio permiten una complejidad que el móvil simplemente no puede igualar. Todo esto me dejó pensando en distintas posibles mecánicas que fueran de control suficientemente simple como para adaptarse a un móvil pero que, a su vez, ofreciera posibilidades complejas.

Si a todo esto le sumamos que en los últimos meses me he dedicado a aprender Unity y a hacer pequeños prototipos de juegos, pues tenemos como resultado la idea de desarrollar Response:Eve.

Estudio económico

1. Identificación del emprendedor

Capacidad emprendedora

No me considero una persona que le guste demasiado asumir riesgos. La verdad es que siempre me he imaginado teniendo un trabajo y sueldo estable trabajando por cuenta ajena en una empresa, aunque sí que es verdad que si en un futuro veo una oportunidad clara de triunfar con una idea de negocio no descarto el lanzarme y llevarla a cabo y quién sabe, tal vez esta pueda ser esa idea....

Tampoco dispongo de mucho capital para empezar el proyecto que se diga , aunque ,todo hay que decirlo, este proyecto tampoco requeriría una grandísima inversión inicial y la tecnología necesaria para realizar la aplicación es muy accesible.

En cuanto a mis habilidades, no dispongo de experiencia laboral en este sector (ni en ninguno) aunque actualmente me estoy formando en este aspecto. Tampoco tengo ni idea de como administrar y gestionar una empresa, aunque este aspecto no me preocupa demasiado ya que me aseguraría de aprender de estos temas antes de iniciar el proyecto.

En cuanto a las habilidades personales y sociales me considero una persona más o menos creativa , capaz de implicarse muchísimo con un proyecto que le interesa y capaz de liderar y motivar a un grupo de personas para alcanzar un fin.

Interés en el proyecto

Mi interés actual por el proyecto es bastante grande ya que pienso que la idea puede tener potencial y creo que puede llegar a destacar entre otros videojuegos de la industria si se ejecuta bien. Los motivos principales para llevar a cabo este proyecto serían, uno, el ganar dinero de forma pasiva a través de la aplicación, y dos, la realización personal que supone acabar un proyecto tan creativo y de algo que me apasiona tanto como es la industria del videojuego.

2. La idea de negocio

Redacción de la idea de negocio

La idea principal de negocio es la de hacer un videojuego multiplataforma llamado Response:Eve. El juego pertenece al género de los plataformas en 2D. En él el jugador tendrá que atravesar diversos niveles para completar el juego. En cada nivel encontraremos una variedad de obstáculos que sobrepasar para completarlos. Estos obstáculos requerirán de habilidad por parte del jugador pero también de inteligencia ,puesto que algunas secciones serán más propias de un juego de puzzles que de un plataformas.

El juego será desarrollado de tal forma que se pueda jugar tanto en el teléfono móvil (Android) como en un ordenador de sobremesa (Windows), aumentando así el número de ventas potenciales.

La propuesta de valor y objetivo

Los factores principales que diferenciarán a mi producto del de la competencia serán:

- La innovación en las mecánicas de juego: La mecánica principal del juego es simple pero permite presentar al jugador un número de posibilidades infinitas. La originalidad de estas mecánicas permitirán al juego destacar de otros juegos del mismo género , y aunque es verdad que el mercado está un poco saturado de juegos plateros (sobre todo el mercado móvil) este tiene un enfoque distinto al juego de móvil genérico que ofrecen las demás compañías. La diferencia principal es que mi juego se enfoca en ofrecer una experiencia de juego más larga y completa (algo parecido a lo que te puedes encontrar en un juego de sobremesa) en vez de la experiencia corta y básica que te ofrecen la mayoría de juegos de móvil.
- El precio: el precio del juego rondará los 2 euros que, comparado con la mayoría de juegos de móvil no es demasiado baja pero que en el mercado de juegos de sobremesa sin duda destacará y lo hará muy apetecible para los consumidores habituales de juegos de PC.

El objetivo durante el primer año sería la de vender suficientes copias como para cubrir los costes del desarrollo: salarios de empleados, equipamiento, software privado, marketing ...

3. Elección de la forma jurídica

La elección jurídica que elegiría, al menos al principio, sería la de empresario individual por diversas razones. En primer lugar, es la forma más simple a nivel de trámites y gestión , además no requiere de capital mínimo de constitución . Otra gran razón es que los impuestos se pagan de forma progresiva dependiendo del capital ingresado, haciéndolo más rentable que pagar el impuesto de sociedades sobre todo al principio. Además , como al principio el volumen de inversión no va a ser demasiado grande, puedo permitirme la responsabilidad ilimitada y pagar las deudas con mis bienes personales. Por último, otra razón es que no dispongo de otros socios para formar la SL , y aunque es verdad que no es estrictamente necesario tenerlos si es recomendable para amortiguar los gastos.

Las exigencias legales para ser un empresario individual son: ser mayor de edad o menor emancipado, que la actividad económica que se realice sea la principal o secundaria fuente de ingresos, que se realice la actividad de forma habitual, no estar de alta en la seguridad social , cotizar como autónomo y estar registrado en Hacienda.

La responsabilidad hacia terceros y la fiscalidad ya están explicadas en el apartado anterior y al tratarse de un empresario individual no hace falta mencionar el número de socios o el capital social.

Obviamente, si el volumen de ingresos aumenta considerablemente y la empresa crece, pues lo lógico sería cambiar la forma jurídica de la empresa a una Sociedad Limitada. Esto es debido a que ,entre otras cosas, saldría más rentable en cuanto a impuestos, ya que el impuesto de sociedades es fijo (nunca más del 25%) mientras que el IRPF que paga el empresario individual puede llegar hasta el 45% si tienes un gran volumen de ingresos. Otra razón es que cuanto más grande es la empresa mayores son las deudas que tiene que asumir, por lo que a partir de cierto punto es peligroso poner tu patrimonio personal en juego. La responsabilidad limitada de la SL acabaría con este problema, separando así tu patrimonio personal del de la empresa.

En el caso de que la empresa creciese muchísimo también se podría plantear cambiar la forma jurídica a una Sociedad Anónima.

4. Estudio de mercado

Análisis de macroentorno

Para analizar el macroentorno económico haremos uso del famoso análisis **PESTIM** (Político, Económico, Social , Tecnológico, Internacional y Medioambiental).

Factores político-legales:

Debido al crecimiento que ha sufrido el sector del videojuego en España en los últimos años, en los Presupuestos Generales del Estado de 2022 se ha presentado una propuesta conocida como plan “España, HUB Audiovisual de Europa”.

Esta estrategia realizada por el Gobierno pretende convertir a España en un destino de referencia en inversión a la hora de producir cualquier tipo de contenido audiovisual y también fomentar el desarrollo de videojuegos nacionales. Es por ello que se ha querido destinar una serie de ayudas enfocadas a la promoción de videojuegos y la creación digital.

De esta forma está previsto que se lleve a cabo una inversión de ocho millones de euros en 2022 y una cuantía similar en el ejercicio presupuestario que se presentará para 2023. Todo ello servirá para apoyar el desarrollo, producción, edición, distribución, comercialización y difusión de los videojuegos y otras formas de creación digital.

Además, la reforma fiscal amplía el concepto de software avanzado en I+D, del mismo modo que la animación y los videojuegos se consideran innovación tecnológica, por lo cual se minorará de la base el 100% de las subvenciones recibidas.Las empresas de videojuegos y animación tienen la posibilidad de conseguir una deducción fiscal del 12% del gasto de desarrollo de un proyecto nuevo; gastos directamente relacionados con en el proyecto como por ejemplo: gastos de personal interno, colaboraciones externas,amortización de equipos, materiales fungibles...

A nivel legal habría que tener en cuenta la Ley 1/1996, también conocida como Ley de Propiedad Intelectual que protege los derechos que tiene un autor sobre su obra. A pesar de que esta ley no ampara a los videojuegos , sí que protege el derecho de autoría sobre aspectos de este como el código fuente, música, historia, diseños ...

Otro apartado legal a tener en cuenta es el de la protección de datos personales de los clientes, descrito en la Ley Orgánica 3/2018 de Protección de Datos Personales y garantía de los derechos digitales.

Factores económicos:

La economía española creció un 5% en 2021. Tras la brutal caída de la producción en 2020 por la pandemia, del 10,8%, el año pasado la actividad se reanudó a pesar de las restricciones y los problemas que todavía acarrea el coronavirus. Se trata de una tasa de crecimiento robusta, la mayor en 21 años, y muy positiva si se tienen en cuenta los lastres que la covid todavía impone a la economía. Eso sí, la persistencia del virus está dificultando que se recobre de forma plena el producto interior bruto.

Con respecto al sector que nos ataña, como hemos mencionado anteriormente, el sector del videojuego en España está en auge. Según la Asociación Española de Videojuegos (AEVI) en 2021, el sector facturó 1.795 millones de euros, un dato que revela un crecimiento del 2,75% con respecto al año anterior. El número de jugadores en España se calcula que es de unos 18,1 millones de usuarios y crece año tras año sin señal de que vaya a parar en un futuro cercano.

Factores socio-culturales:

Los videojuegos son una de las aficiones favoritas de los españoles, los cuales le dedican una media de 8,1 horas a la semana.

El videojuego siempre ha estado presente como una de las formas de entretenimiento más habituales entre los españoles, sobre todo entre los jóvenes. En los últimos años, el número de personas que juegan ha aumentado considerablemente, sobre todo en 2020 debido al confinamiento que sufrió la población por el Covid-19. Esta cuarentena hizo que la gente consumiera más videojuegos junto con otros tipos de entretenimiento como los servicios de streaming.

Factores tecnológicos:

Las tecnologías para desarrollar videojuegos están continuamente en evolución y cada vez aparecen nuevos motores de juegos en el mercado que, si bien no dejan obsoletos a los anteriores, sí que presentan mejoras sustanciales con respecto a estos. Por suerte el motor que utilizaré será Unity 3D el cual se renueva cada cierto tiempo adaptándose a las necesidades del mercado.

Factores medioambientales:

Por suerte, el impacto medioambiental de desarrollar videojuegos es muy bajo, ya que el único recurso que se gastaría que pudiera tener un impacto ambiental sería la luz necesaria para el funcionamiento de los equipos y demás sistemas eléctricos de la oficina.

Aún así como empresa es importante tener una buena conciencia medioambiental, por ello la mayoría de empresas tecnológicas promueven buenas prácticas como el reciclaje de equipos o el uso de equipos más eficientes energéticamente.

En nuestro país el Real Decreto 208/2005, de 25 de febrero, establece la normativa sobre reciclaje de aparatos eléctricos y electrónicos, así como la gestión de sus residuos. El ámbito de la ley es amplio y propone sistematizar la recogida y aprovechamiento de todo tipo de aparatos que funcionen gracias a la corriente eléctrica o los campos magnéticos.

Factores internacionales:

De acuerdo con los datos del último informe global del mercado de los videojuegos presentado por Newzoo, el mercado mundial de videojuegos habría aumentado en un reseñable 9,6%, pasando de una facturación de 134.900 millones de dólares (119.606 millones de euros) en 2018 a 152.100 millones de dólares (133.670 millones de euros) en 2019. Este crecimiento estuvo impulsado, sobre todo, por un espectacular incremento del 11,7% en Norteamérica, y del 11,5% en Europa, Oriente Medio y África y del 11,1 en América Latina.

China y Estados Unidos han seguido liderando los datos de facturación mundial en 2016, seguidos de lejos por Japón, Corea del Sur y Alemania.

Análisis del microentorno

Para analizar el microentorno haremos uso del **Análisis Porter de las fuerzas competitivas**, el cual consiste en analizar a los competidores, los clientes, los

proveedores, los posibles productos sustitutivos y la aparición de nuevos competidores.

Competidores

El mercado de los videojuegos es uno de los más competitivos del entretenimiento actual. Cada año salen miles de juegos que compiten por la atención de los jugadores y solo unos pocos acaban destacando. Al tratarse de un mercado tan globalizado, al sacar un videojuego estás compitiendo con todos los juegos del mismo género en todo el mundo por lo que la competencia es altísima y ofrecer un producto distintivo es muy importante.

Los competidores principales de Response:Eve serían todos aquellos juegos plataformas simples de móvil y de PC tales como Geometry Dash, Temple Run y miles de otros más que serían imposible de enumerar.

Clientes

La cantidad de gente que pasa su tiempo libre jugando videojuegos va aumentando año tras año (como hemos visto en el análisis del macroentorno). El tener tal cantidad de compradores potenciales aumenta las posibilidades de recuperar el capital invertido siempre que el juego sea lo suficientemente atractivo para ellos y se haga una campaña de marketing decente.

Además, en los últimos años ha habido un “boom” en la consumición de juegos “indies” (de bajo presupuesto), lo cual es una buena noticia para nosotros ya que nuestro juego caería en esta categoría. Lo malo es que el género de plataformas es el más frecuente entre los juegos indies y no es de los que más venden dentro de este grupo por norma general.

Productos sustitutivos

Si hablamos de los videojuegos en general, no hay ningún producto dedicado al entretenimiento que se le asemeje. Dentro del propio mundo de los videojuegos, un posible producto sustitutivo de los videojuegos de plataformas en 2D económicos, serían los videojuegos conocidos como “Infinite Runners”, que cumplen una función similar a plataformas en el sentido de que ambos proponen retos mecánicos para el jugador.

Proveedores

En cuanto a las proveedores, no existe una gran variedad de empresas que proveen el software y los servicios necesarios para desarrollar videojuegos por lo que los desarrolladores indies dependen de los pocos que existen y están sujetos a sus políticas. Los ejemplos más populares son Unity Technologies con su motor de juego Unity Pro y Epic Games con su motor Unreal Engine.

Nuevos competidores

La barrera de entrada a la hora de desarrollar un videojuego depende de la magnitud de este. Si hablamos de juegos AAA (alto presupuesto) la barrera de entrada es muy grande por lo que la aparición de nuevos competidores no es tan habitual. En el caso de los indies la barrera es mucho más baja (de hecho cualquiera puede desarrollar un videojuego con un presupuesto mínimo). Si a eso le sumamos que los canales de distribución son muy accesibles (cualquiera puede publicar un juego en Steam o en la Play Store) pues como resultado cada día, y cada vez más, aparecen nuevos competidores.

Ubicación del negocio

Al tratarse del desarrollo de una aplicación, el lugar de ubicación de la empresa no es relevante puesto que el producto no se vende de manera local, sino a nivel global por Internet.

Por lo tanto, a la hora de elegir la ubicación los parámetros a tener en cuenta serían el precio de alquiler y la cercanía con los empleados. Por ello elegiría una oficina en la zona más económica de Alfaz del Pi. Un ejemplo de posible oficina sería la en la siguiente ubicación:



Análisis DAFO

Debilidades	Amenazas
<ul style="list-style-type: none"> - La falta de experiencia en el sector - La escasa capacidad de gestión empresarial - La dificultad para encontrar plantilla experimentada en la zona 	<ul style="list-style-type: none"> -Alta competitividad en el mercado - Saturación de videojuegos del mismo genero en el mercado - La actual crisis económica
Fortalezas	Oportunidades
<ul style="list-style-type: none"> - Conocimiento amplio sobre el género plataformero - La calidad del producto - El precio del producto - Disponer de la tecnología apropiada - Pasión por el proyecto 	<ul style="list-style-type: none"> - El "boom" reciente de los juegos indies y de los plataformas - Aumento en la popularidad del financiamiento por Kickstarter

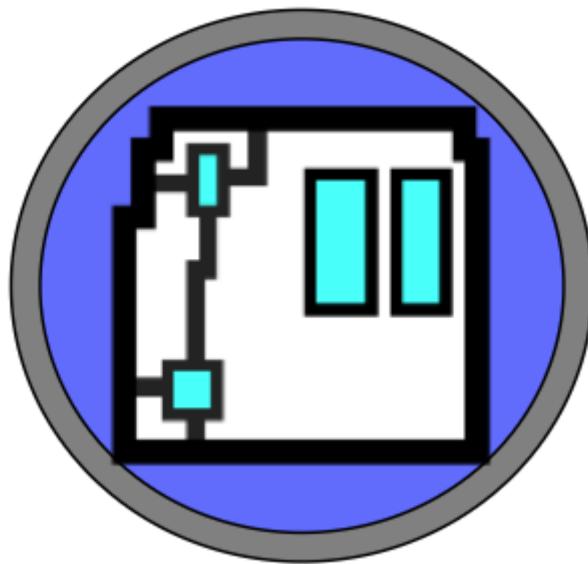
Plan de marketing

Producto

Las características del producto, sin entrar en detalle en su diseño, será la de un juego plataformero con un total estimado de 10 grandes niveles, con la posibilidad de añadir más en futuras actualizaciones. Además este dispondrá de varias opciones de configuración que permitirán personalizar más la experiencia del jugador.

El producto que vamos a vender sería considerado un bien, debido a que una vez lo compras lo tendrás almacenado de forma permanente en tu dispositivo y no necesitarás pagarle de forma recurrente para poder jugarlo. La necesidad que cubre es puramente de entretenimiento para el usuario.

El logotipo del producto aún está por determinar, aunque un prototipo del logo podría ser el siguiente:



Precio

Si nos fijamos en el mercado, el precio para un juego de móvil de estas características rondaría los 2, 3 o incluso 4 euros (si no contamos con los juegos gratuitos). Debido a la poca predisposición que tiene la gente a pagar por juegos de móvil en general lo más sensato sería que el juego tuviera un precio relativamente bajo. Por ello el juego en Android costaría 0,99€ y 2,99€ en PC (en esta plataforma la gente suele estar dispuesta a gastar más dinero). Estos precios variarán según la economía del país y el valor que tenga su moneda. Esto es una práctica común

debido a que si no se ajustan los precios, las ventas en países con economías más pobres suelen ser catastróficamente bajas (por ejemplo en los países latinoamericanos).

Podemos permitirnos poner este precio tan bajo debido a que los gastos de producción de las copias digitales son nulos y el gasto de distribución va en función de las copias vendidas (las plataformas de distribución se llevan un porcentaje de las ventas por publicar tu producto). Este precio tan bajo nos permitirá destacar frente a la competencia y conseguir un mayor número de ventas.

Promoción

La estrategia de promoción es sencilla y consistirá en comprar anuncios en redes sociales, ya que estás nos brindan la oportunidad de dirigirnos a un público en específico (los jugadores) a través de sus algoritmos. Por ejemplo, en Youtube podemos poner nuestros anuncios en videos que tengan que ver con la temática de videojuegos plataformeros y así captar la atención de nuestros potenciales consumidores con más precisión. También dispondremos de los servicios de publicidad de Google Play el cual se encargará automáticamente de publicitar nuestro juego dentro de otros juegos gratuitos que ganen dinero a partir de anuncios.

Distribución

Cuando se trata de videojuegos digitales las opciones de distribución son muy limitadas. En el mercado de PC la opción más evidente es Steam, la plataforma más popular con diferencia de videojuegos digitales en PC. Por distribuir el videojuego en su plataforma, Steam se queda con el 30% de cada videojuego vendido.

Para la versión de Android la única opción viable es la Play Store de Google, la cual se queda con el 15% del primer millón de dólares y con el 30% a partir de ahí.

Plan de producción y recursos financieros

Inversiones y gastos

Inversiones	Gastos
Equipos informáticos: 3000€	Alquiler local: unos 600€/mes
Mobiliario de oficina: 2000€	Luz: 120€/mes
	Agua: 40€/trimestre
	Internet + Linea de telefono:45€/mes
	Gas: 30€/mes
	Sueldos (2 trabajadores): 2500€/mes
	Instalación fibra óptica: 60€
	Seguros sociales de los trabajadores: 714,29€

Origen de la financiación

Teniendo en cuenta los gastos y las inversiones calculadas en el apartado anterior, suponiendo que el desarrollo del videojuego tuviera una duración estimada de 6 meses, nos harían falta unos 29.492 € en total para poder financiar el proyecto.

De esos 29.492 € yo podría aportar 10.000€ sacados de mi bolsillo y otros 10.000€ por préstamos familiares. El otro 33% lo intentaría sacar a través de Kickstarter, una plataforma web que permite financiar tu proyecto a través de las donaciones de los usuarios. Para incentivar que los usuarios inviertan en tu proyecto se suele utilizar un sistema de recompensas en el que se ofrecen extras a los usuarios que superen cierta cantidad invertida, como niveles exclusivos, sus nombres en los créditos o incluso la oportunidad de participar de forma creativa en el desarrollo. Cada vez más

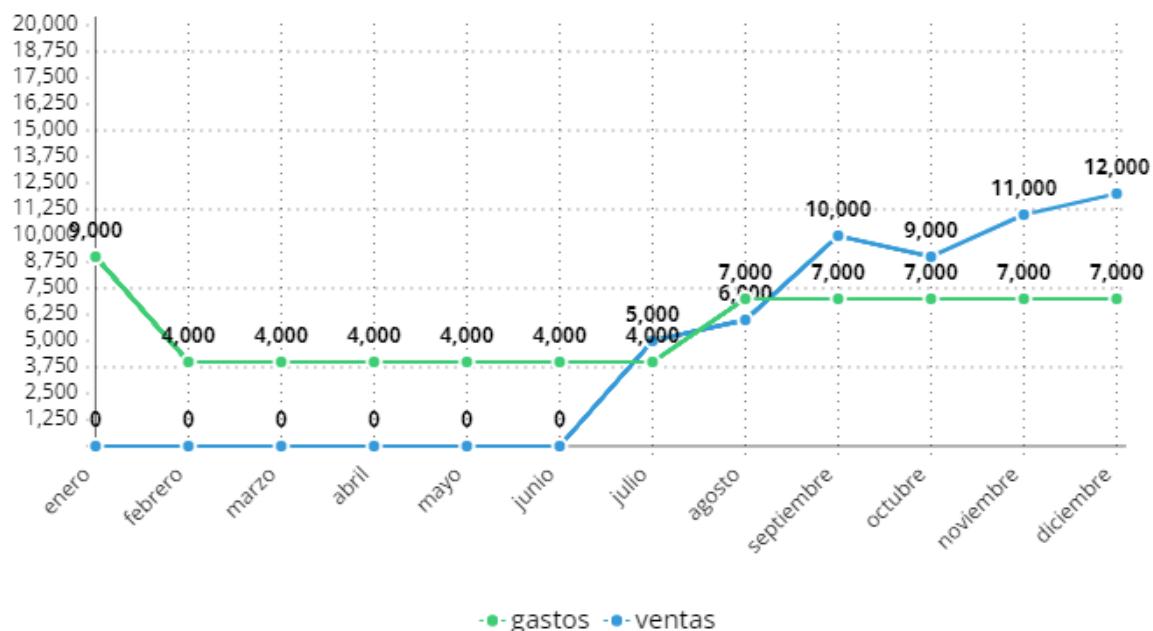
el Kickstarter es el método de financiación preferido en el mundillo de los juegos indies y por eso he pensado que sería la mejor opción.

Ayudas y subvenciones

Al empezar con el proyecto empresarial, al haber elegido la forma jurídica de empresa individual y ser la primera vez que me doy de alta como autónomo, tengo derecho a acceder a la tarifa plana en la cuota de autónomos. Esta se divide en tres tramos:

1. Primer tramo (12 meses): 60 € al mes durante los primeros 12 meses. Dicho de otra forma, un total de 720 € anuales.
2. Segundo tramo (13 a 18 meses): Durante los siguientes seis meses, se me aplicará una reducción equivalente al 50% de la cuota. Esto significa que la cuota mensual ascenderá a 143,10 €.
3. Tercer tramo (19 a 24 meses): Podré optar a una reducción equivalente al 30% de la cuota. En este punto la reducción de la cuota de la tarifa plana empieza a terminar y ya estaré pagando 200,30 €.

Umbral de rentabilidad o punto muerto



Como podemos observar en el gráfico anterior, al principio el número de ventas será nulo ya que los primeros 6 meses será el período de desarrollo de la aplicación. Una

vez pasados esos 6 meses el videojuego saldrá a la venta, y las ventas irán incrementando mes a mes con el incremento de la popularidad del juego.

Los gastos empezarán arriba debido a la inversión inicial que se tiene que hacer para el material de oficina. Luego, durante los 6 meses de desarrollo, los gastos se mantendrán estables mes a mes. Y por último, una vez el juego haya salido, habrá un incremento en los gastos debido a que dará comienzo la campaña de marketing del producto, en la que se invertirán unos 3000€ mensuales.

Plan económico-financiero

(Está en el excel adjunto)

Entradas	Ene.	Feb.	Mar.	Abr.	May.	Jun.	Jul.	Ago.	Sep.	Oct.	Nov.	Dic.	Total
Aportación del empresario	10000	0	0	0	0	0	0	0	0	0	0	0	10000
Préstamo Familiar	10000	0	0	0	0	0	0	0	0	0	0	0	10000
Financiación Kickstarter	10000	0	0	0	0	0	0	0	0	0	0	0	10000
Ventas	0	0	0	0	0	0	5000	6000	10000	9000	11000	12000	53000
Intereses de la cuenta corriente	16,73	16,73	16,73	16,73	16,73	16,73	16,73	16,73	16,73	16,73	16,73	16,73	184,03
Total entradas	20016,73	16,73	16,73	16,73	16,73	16,73	5016,73	6016,73	10016,73	9016,73	11016,73	12016,73	61184,03
Salidas													
Alquiler	600	600	600	600	600	600	600	600	600	600	600	600	7200
Compra de equipo de trabajo	5000	0	0	0	0	0	0	0	0	0	0	0	5000
Salarios	2500	2500	2500	2500	2500	2500	2500	2500	2500	2500	2500	2500	30000
Publicidad	0	0	0	0	0	0	3000	3000	3000	3000	3000	3000	18000
Seguridad Social	714	714	714	714	714	714	714	714	714	714	714	714	8568
Seguros	300	0	0	0	0	0	0	0	0	0	0	0	300
Suministros	153	153	153	153	153	153	153	153	153	153	153	153	1683
Total salidas	9267	3967	3967	3967	3967	3967	6967	6967	6967	6967	6967	6967	70904
Entradas menos salidas	10749,73	-3950,27	-3950,27	-3950,27	-3950,27	-3950,27	-1950,27	-950,27	3049,73	2049,73	4049,73	5049,73	-9719,97
Saldo en el banco.c.c	10749,73	6799,46	2849,19	-1101,08	0	0	0	0	0	0	0	0	2296,76
Saldo en la cuenta de crédito	0	0	0	0	-5051,35	-9001,62	-10951,89	-11902,16	-8852,43	-6802,7	-2752,97	0	

(Captura del plan de tesorería)

Análisis

Como podemos observar en el plan de tesorería, durante los primeros meses empezaremos teniendo pérdidas debido a que no estaremos obteniendo ingresos directos por desarrollar el videojuego. Una vez pasados estos seis meses, empezaremos a tener suficientes ingresos como para ir afrontando la deuda y acabaremos el año con un balance total positivo.

Trámites de constitución

Organismo	Trámites
Hacienda	<ul style="list-style-type: none">- Alta del I.A.E- Declaración censal (darse de alta en el IRPF)
Ayuntamiento	<ul style="list-style-type: none">- Licencia de actividad (Comunicación previa)- Licencia de apertura
Tesorería General de la Seguridad Social	<ul style="list-style-type: none">- Inscripción de la empresa en la seguridad social- Alta en el régimen de Autónomos (RETA)- Alta de los trabajadores en el régimen general- Comunicación de apertura en la Consejería de Empleo o Trabajo de la comunidad
Registro Mercantil	<ul style="list-style-type: none">- Libro de visitas

Análisis

Necesidades del proyecto y planificación temporal

En este apartado hablaremos de los puntos por los que tiene que pasar el desarrollo, el orden de dichos puntos, las fechas, y la importancia de cada uno.

Fase 1: Desarrollo de las mecánicas

Durante esta fase me encargaré de programar todas las mecánicas y a testearlas en un nivel cerrado. Las mecánicas de movimiento en un principio serán cinco: el tirón y el empuje magnético, la bala teletransportadora, la goma, los portales y el gancho.

-El tirón y el empuje magnético serán las mecánicas básicas de movimiento. Al apuntar a una superficie magnética, el jugador será atraído o repelido en la dirección de apuntado, lo que le permitirá impulsarse libremente por el aire o por el suelo.

-Al apuntar a una superficie de bala teletransportadora el jugador emitirá un proyectil en la dirección de apuntado y si apunta al mismo tipo de superficie o a otra superficie que no tenga un efecto determinado (como el suelo) el jugador se teletransportará a la posición del proyectil.

-Si el jugador apunta a una superficie de goma adquirirá la propiedad de la superficie y rebotará una vez contra la siguiente superficie con la que impacte. Además se volverá invencible al daño eléctrico.

-Al apuntar a una superficie de portal, se generará un portal en el punto al que apuntemos de la superficie. Si apuntamos a otra superficie de portal distinta , generaremos otro distinto. Si tenemos una pareja de portales , al atravesar uno nos teletransportaremos a la posición del otro, conservando nuestros vectores de velocidad y posición relativos.

-Si el jugador apunta a una superficie de gancho, lanzará un gancho con cuerda que le llevará a la posición del agarre, pudiendo, eso sí, soltarse en cualquier momento y mantener su momento lineal.

A estas mecánicas debemos de sumarle otras adicionales (como la superficie polarizadora la cual cambiará los polos de los imanes) y otras básicas como las mecánicas de muerte (trampas y objetos que te matan).

Al no disponer de ningún sprite, utilizaré figuras básicas como placeholders para poder testar las mecánicas.

Fase 2: Diseño del personaje y del entorno

Una vez programadas las mecánicas, técnicamente ya podríamos empezar con el diseño de niveles . Sin embargo, empezaré antes con el diseño de los personajes y el escenario. Esto es debido a que, en mi opinión, es preferible disponer de los assets antes de diseñar niveles por una cuestión de ahorro de tiempo. Ya que de otra forma tendríamos que repasar los niveles ya creados y sustituir los placeholders por sprites (se puede automatizar si lo organizas bien, pero siendo realistas algo de tiempo sería desperdiciado).

El diseño del protagonista y el entorno será pixelart debido a la relativa facilidad y rapidez en comparación al dibujo tradicional.

Fase 3 y 4: Diseño y desarrollo de los niveles

En esta fase me dedicaré, en primera instancia, a diseñar los niveles con lápiz y papel en una libreta y una vez que esté satisfecho con los bocetos, me encargaré de trasladarlos al motor de juego para testearlos y hacer reajustes en caso de que fuera necesario.

Fase 5: Diseño y programación de menús

Una vez desarrollado los niveles me encargaré de realizar la interfaz de los menús. En total serán 4: el menú principal,el de pausa,el menú de configuración y el de selección de niveles.Una vez implementada la parte visual pasaré a implementar la parte funcional de estos:

- El menú principal dispondrá de dos opciones: empezar a jugar (el cual nos llevará al selector de niveles) y configuración (el cual nos permitirá hacer ajustes de sonido, brillo ...)
- El menú de pausa nos permitirá seguir con la partida, resetear el nivel o salir al menú principal.
- El menú de selección de niveles nos permitirá seleccionar los niveles que tengamos desbloqueados.

Fase 6: Implementar sistema de puntuación

Durante esta fase implementaré el sistema de puntuación que consistirá en puntuar como de eficiente ha sido el jugador al pasarse el nivel (teniendo en cuenta por ejemplo el número de muertes o el tiempo transcurrido).

Fase 7: Pulir aspecto visual y sonoro

En este período me encargaré de pulir animaciones, efectos visuales, sprites, movimiento del personaje ... e introduciré los efectos sonoros del juego, los cuales sacaré de una librería gratuita en internet.

Fase 8: Testear y arreglar bugs

Y por último, antes de terminar iniciaré el período de testeo que consistirá en probar el juego durante mucho tiempo, intentando encontrar fallos en la programación y arreglarlos obviamente.

Planificación temporal (Diagrama de Gantt)

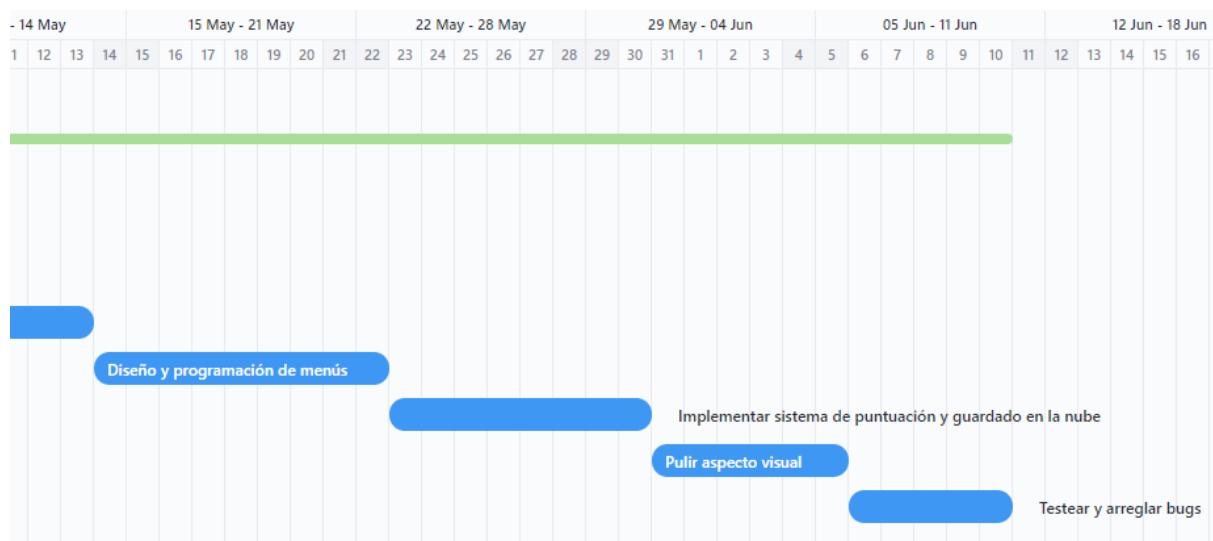
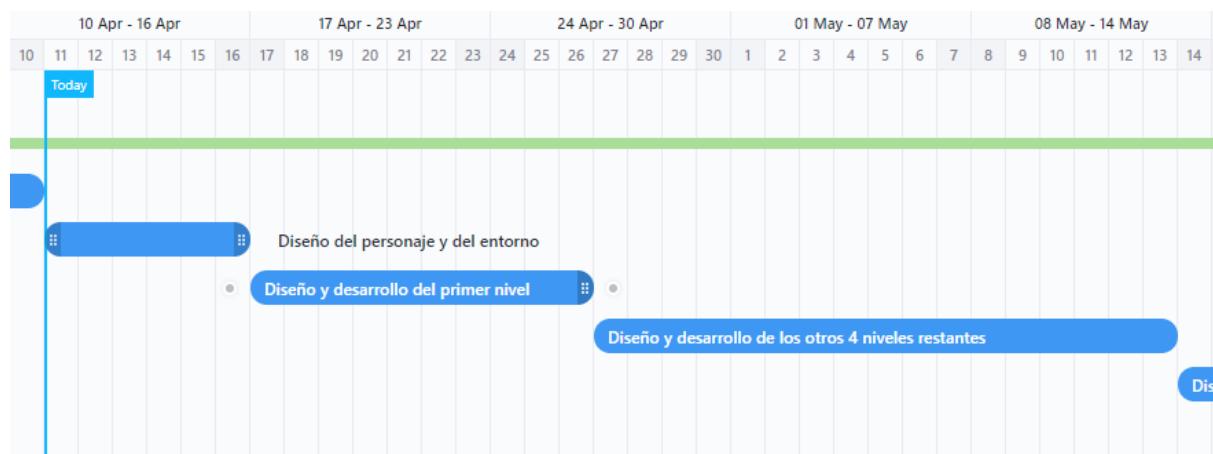
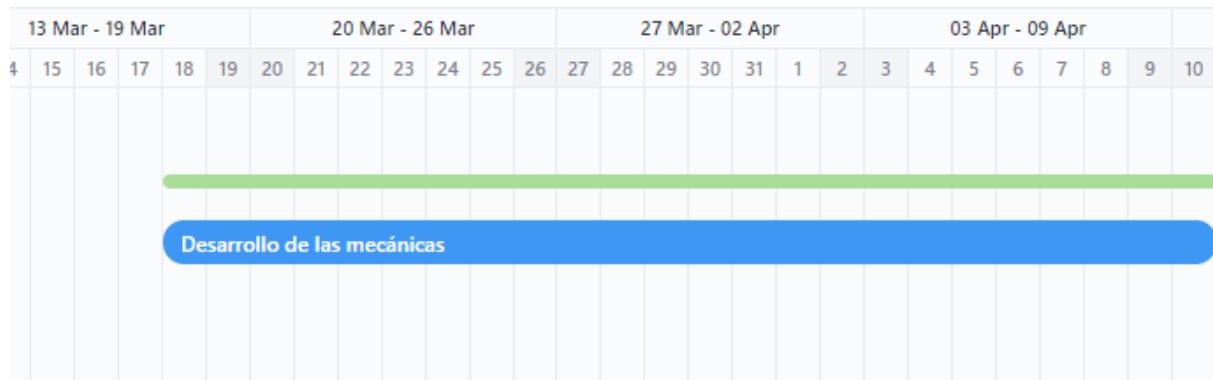
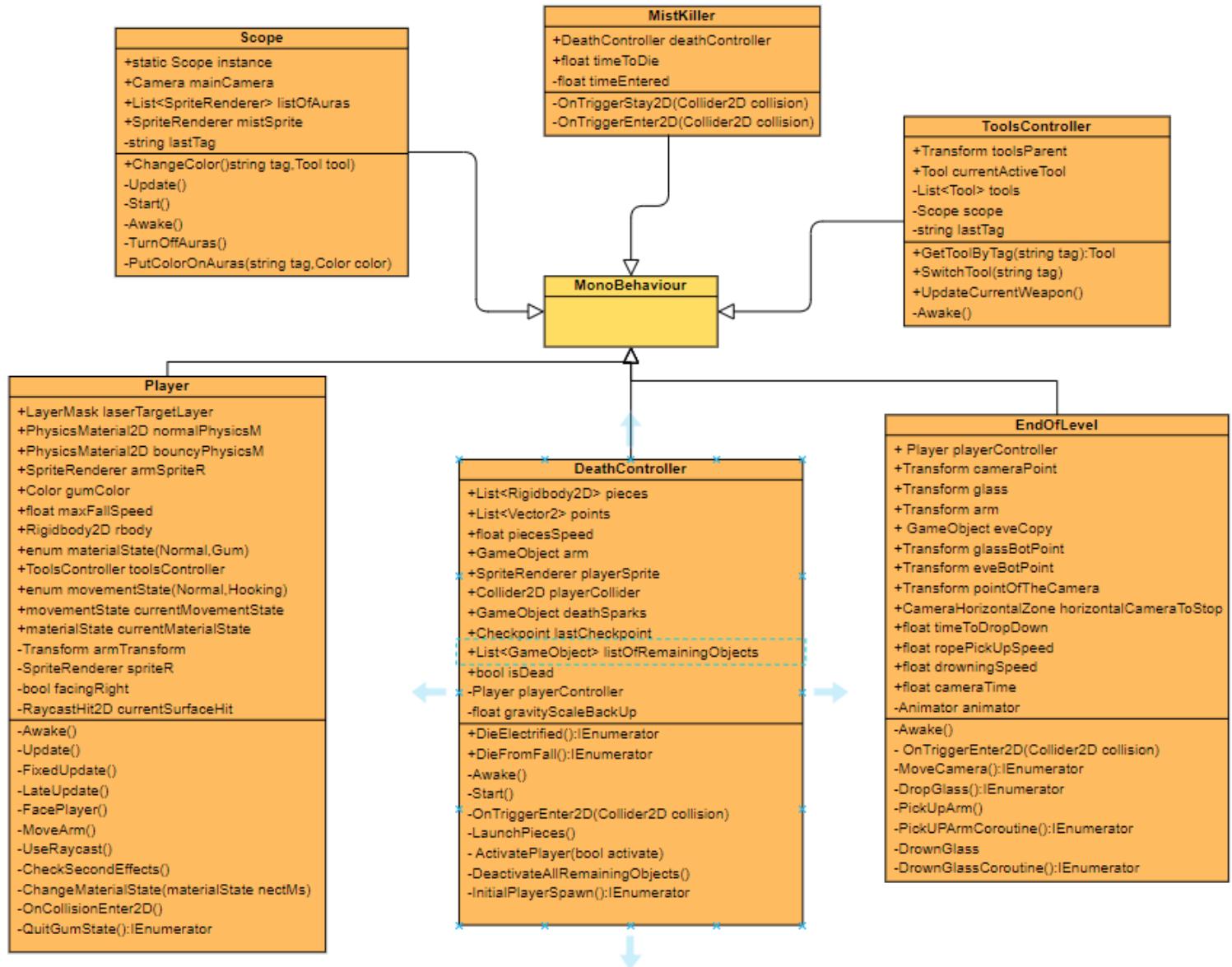


Diagrama de clases

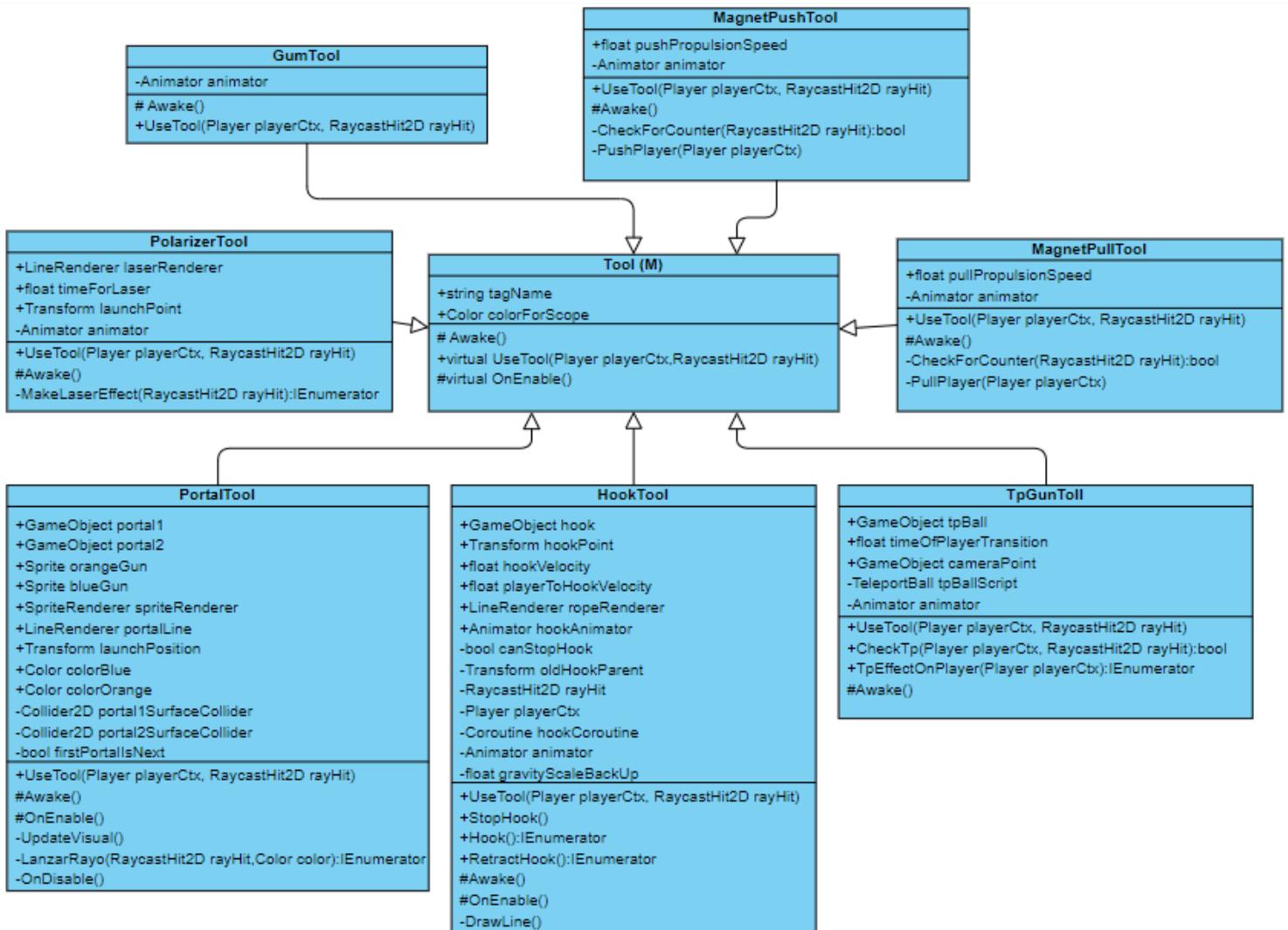
Para entender mejor el UML, dividiré las clases teniendo en cuenta la estructura de carpetas que he utilizado.

-Carpeta Player

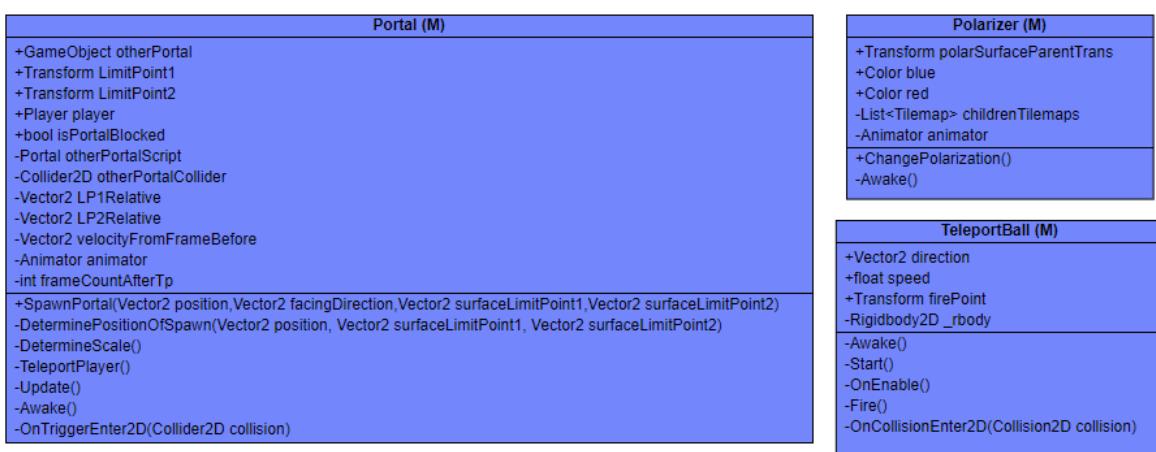


Como la mayoría de las clases heredan de MonoBehaviour en los siguientes diagramas marcaré con (M) todas las clases que lo hereden.

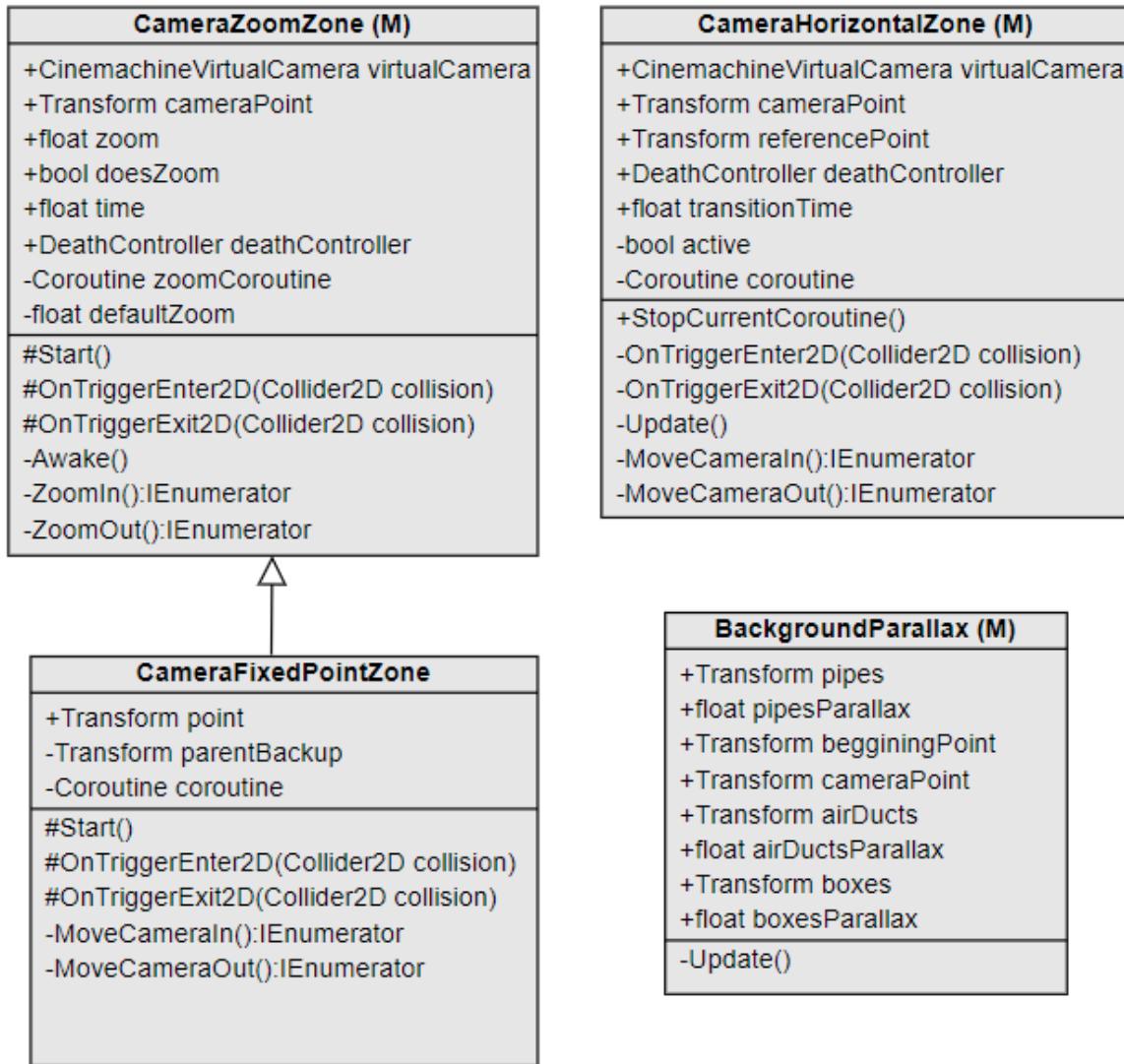
-Carpeta Tools



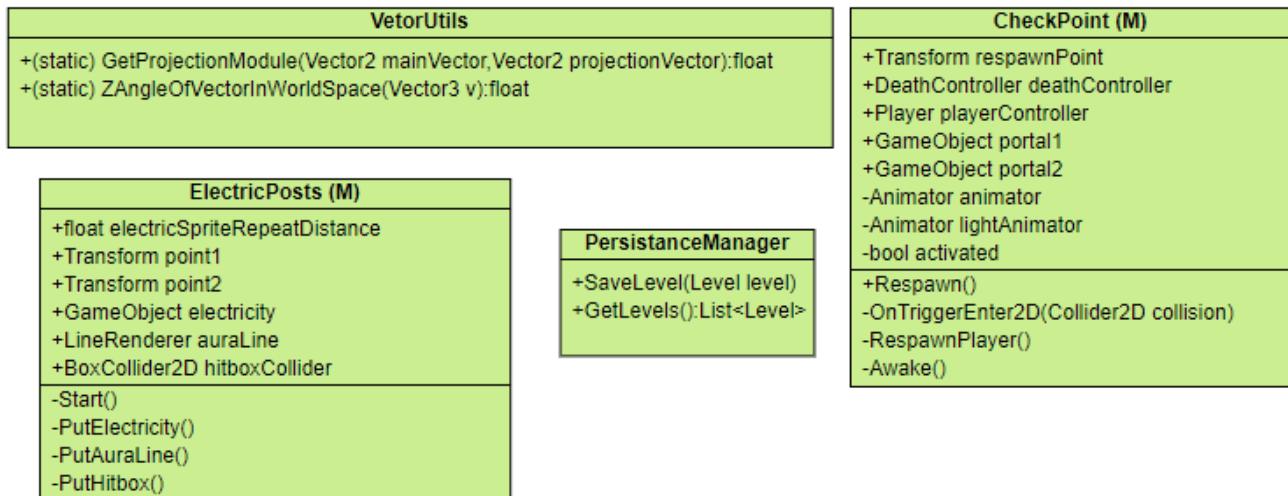
-Carpeta ToolsComplements



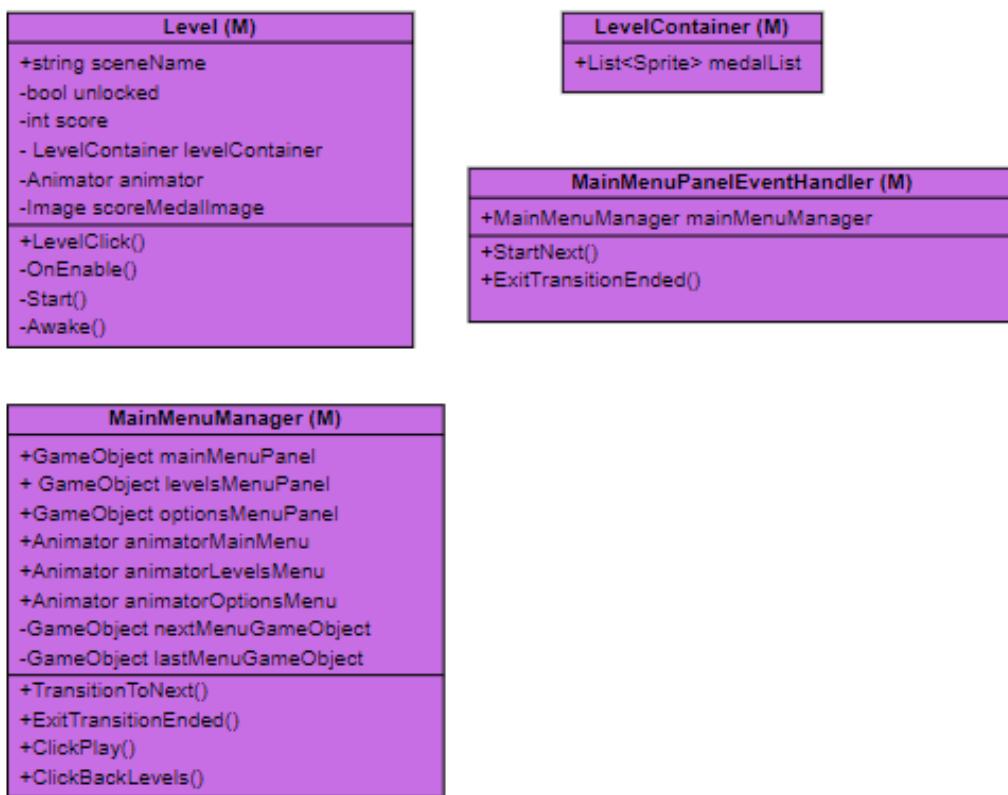
-Carpeta Camera



-Carpeta Extras



-Carpeta Menús



Persistencia de datos

Para la persistencia de datos utilizaremos un sistema llamado PlayerPrefs. PlayerPrefs es un sistema de guardado de datos que viene implementado por defecto en Unity. Este funciona como un diccionario, es decir, podemos guardar datos simples (string, float e int) en una lista que almacenará el valor y una clave para identificarlo. Luego dicho valor se puede recuperar o borrar a conveniencia.

Este sistema es ideal para el proyecto ya que la cantidad de datos a guardar es muy baja. Si no fuera este el caso, probablemente utilizaría un modelo no relacional a través de archivos JSON.

Entornos de programación

Lenguajes y tecnologías que se van a emplear.

Unity:



Unity es un motor de videojuego multiplataforma creado por Unity Technologies. Unity está disponible como plataforma de desarrollo para Microsoft Windows, Mac OS, Linux. La plataforma de desarrollo tiene soporte de compilación con diferentes tipos de plataformas entre las cuales se encuentran Windows y Android. Unity es utilizado por el 58% de la población de desarrolladores indies convirtiéndose así en el motor de videojuegos más popular en el mundo.

C#:



Unity utiliza Mono, el cual es una implementación multiplataforma del .NET framework de Microsoft, por lo que su lenguaje primario y en el que están escritas todas sus librerías es C# y será el lenguaje que utilice.

"C#" es un lenguaje de programación multiparadigma desarrollado y estandarizado por la empresa Microsoft como parte de su plataforma .NET. Su sintaxis básica deriva de C/C++ y utiliza el modelo de objetos de la plataforma .NET, similar al de Java, aunque incluye mejoras derivadas de otros lenguajes.

Visual Studio Community 2015:

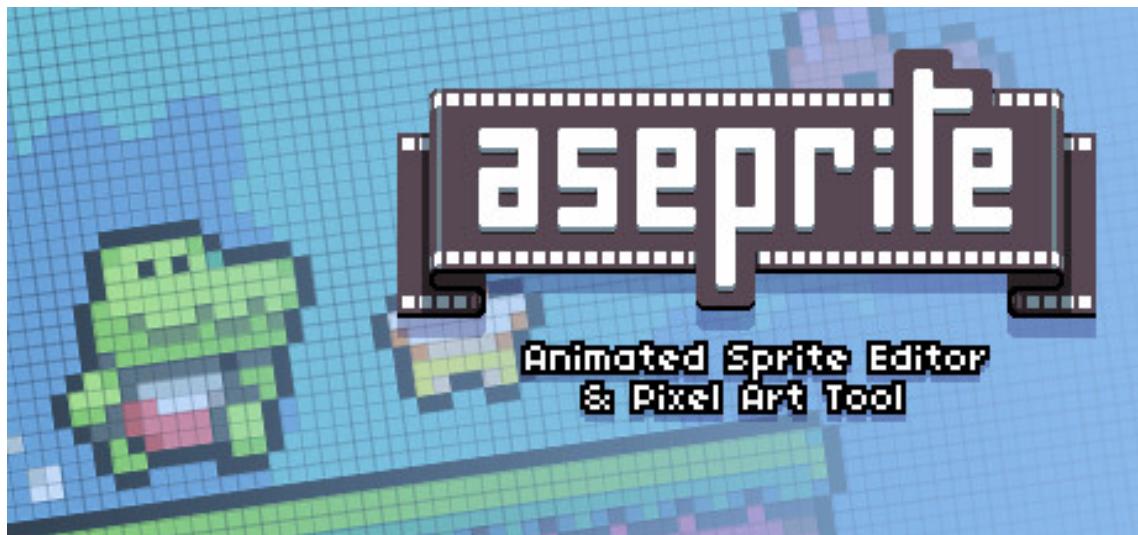


Microsoft Visual Studio es un entorno de desarrollo integrado (IDE, por sus siglas en inglés) para Windows y macOS. Es compatible con múltiples lenguajes de programación, tales como C++, C#, Visual Basic .NET, F#, Java, Python, Ruby y PHP, al igual que entornos de desarrollo web, como ASP.NET MVC, Django, etc., a lo cual hay que sumarle las nuevas capacidades en línea bajo Windows Azure en forma del editor Mónaco.

Visual Studio permite a los desarrolladores crear sitios y aplicaciones web, así como servicios web en cualquier entorno compatible con la plataforma .NET (a partir de la versión .NET 2002). Así, se pueden crear aplicaciones que se comuniquen entre estaciones de trabajo, páginas web, dispositivos móviles, dispositivos embebidos y videoconsolas, entre otros.

Utilizaré Visual Studio para escribir el código de los scripts de Unity.

Aseprite:

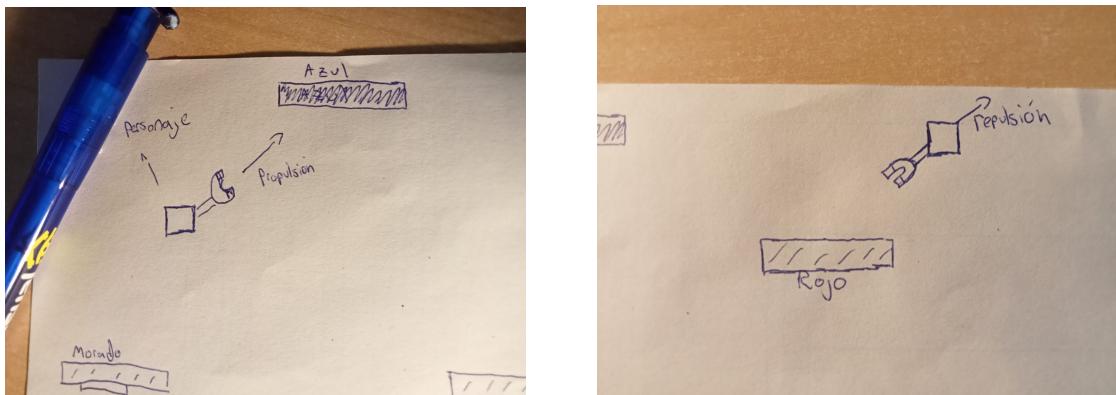


Aseprite (antiguamente llamado Allegro Sprite Editor) es un editor de gráficos rasterizados para Windows, macOS y Linux, creado por David Capello. Está diseñado específicamente para crear y editar Sprites y animaciones píxel art.

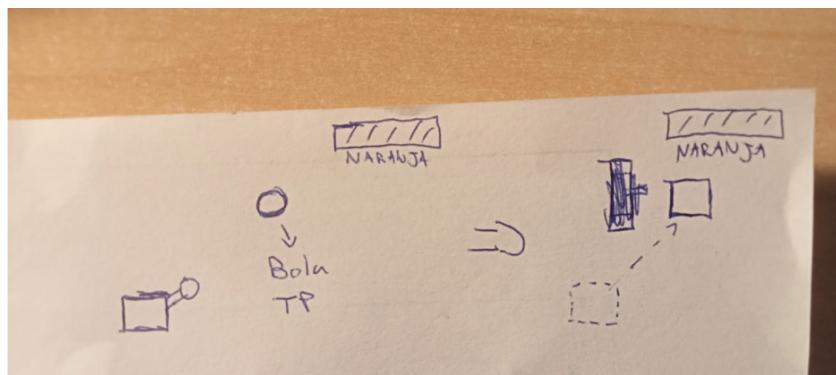
Implementación

Prototipo del proyecto

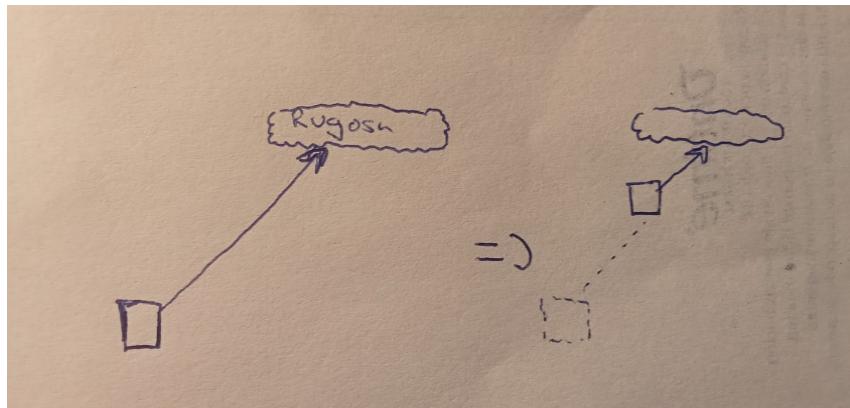
Lo primero en lo que empecé a trabajar fue en las dos mecánicas principales de movimiento, la atracción y la repulsión. Estas mecánicas son las que más se utilizarán a lo largo del juego debido a que el protagonista no puede moverse de forma convencional y necesitará de estos impulsos para pasar de una zona a otra y al ser tan versátiles nos permitirán tanto movernos horizontalmente como verticalmente.



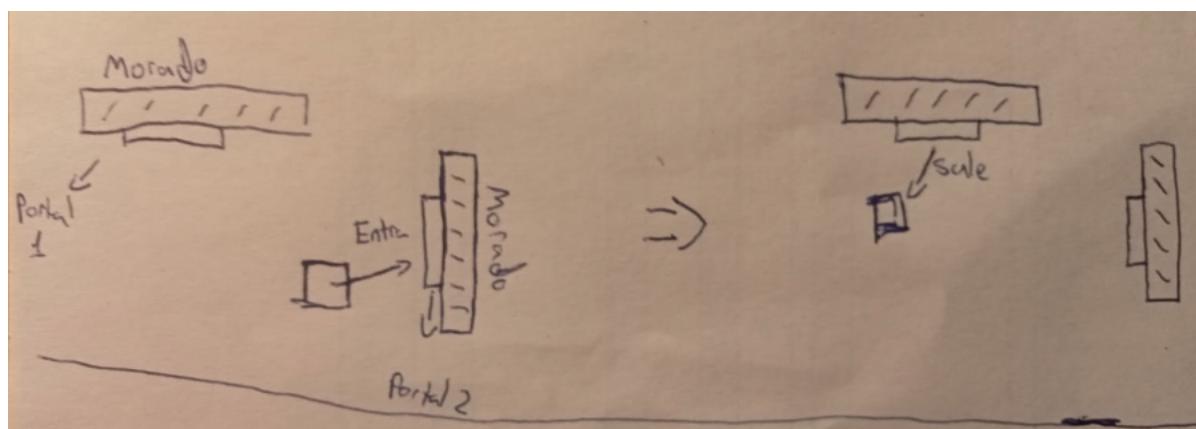
Una vez programadas y ajustadas las mecánicas de los imanes empecé con las mecánicas más complejas. La primera que hice fue la bala teletransportadora, la cual consiste en una pistola que lanza una bola de energía que al reactivarse te teletransporta a su posición. Pensé que esta mecánica sería interesante porque requeriría de un timing preciso para aterrizar en el lugar deseado y porque es un buen sustituto de la mecánica de atracción en el caso de que quieras colocar a tu personaje en un lugar en específico debido a su mayor precisión.



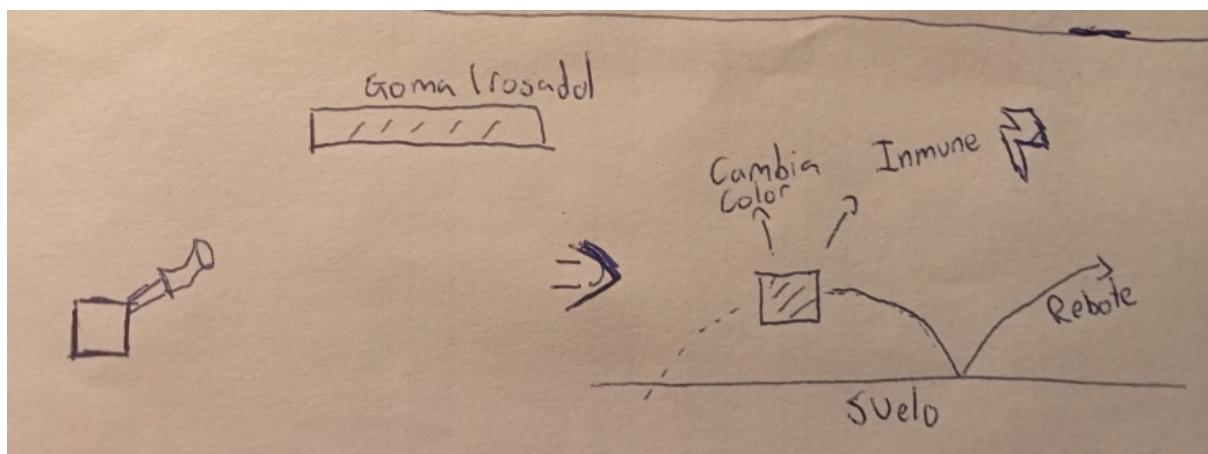
Después quería hacer otra mecánica sustitutiva del imán que te permitiera atravesar grandes distancias en línea recta porque pensé que podía dar lugar a niveles interesantes, de ahí nació la idea del gancho. Además de hacer lo que he mencionado, le añadí la posibilidad de poder soltar el gancho en cualquier momento del recorrido para mantener tu momento lineal y poder saltar obstáculos.



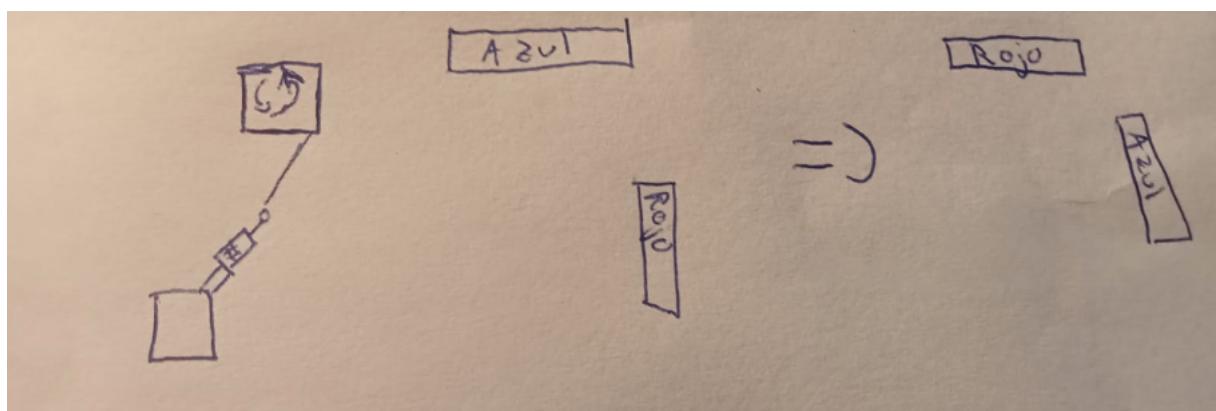
Una vez terminada la mecánica anterior, quise probar a realizar una mecánica más compleja que presentase un gran número de posibilidades a la hora de diseñar niveles. Entonces pensé en adaptar la mecánica de portales del famoso juego "Portal" para un entorno de dos dimensiones. La mecánica consiste en crear dos portales en superficies distintas y cuando el jugador atraviese uno de ellos aparecerá en el otro manteniendo su posición y velocidad relativas al primero. Esté fue sin duda la mecánica con más complejidad a la hora de programarla, pero ya hablaremos de los detalles más adelante.



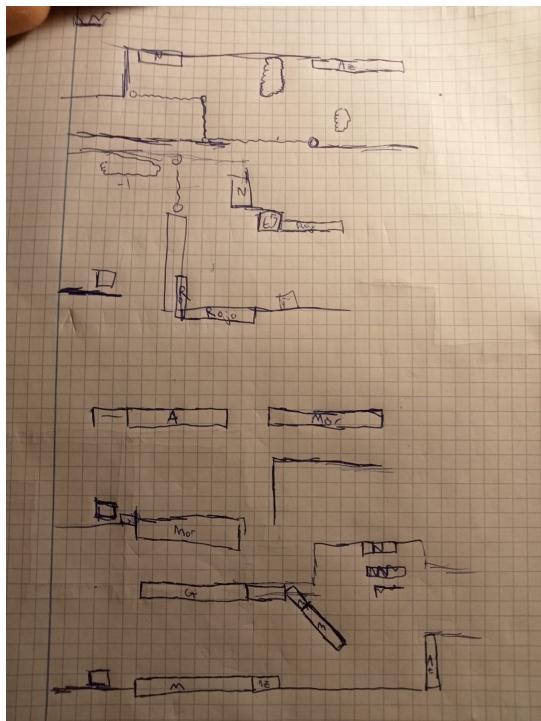
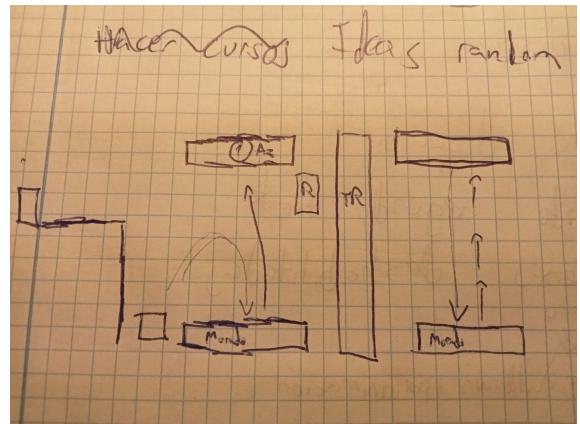
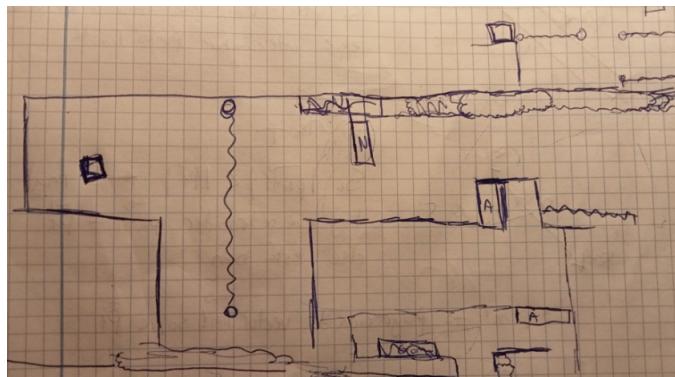
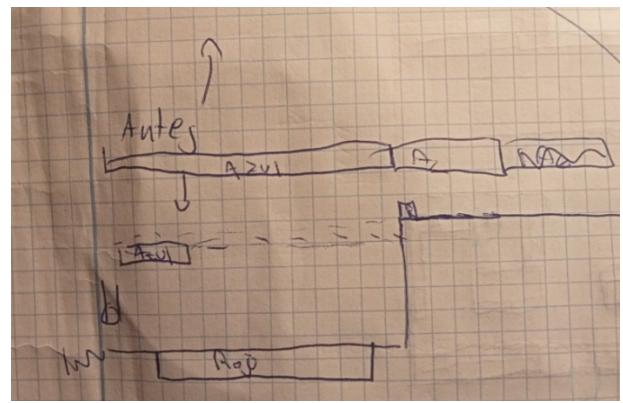
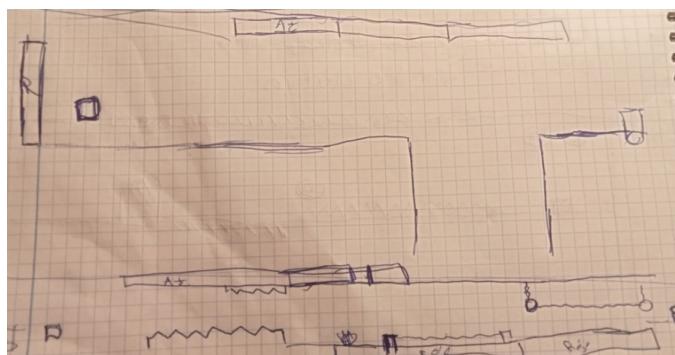
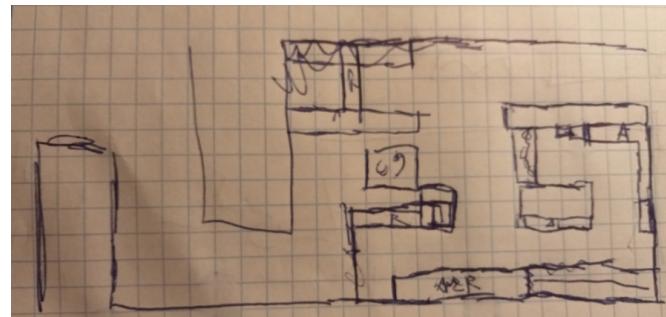
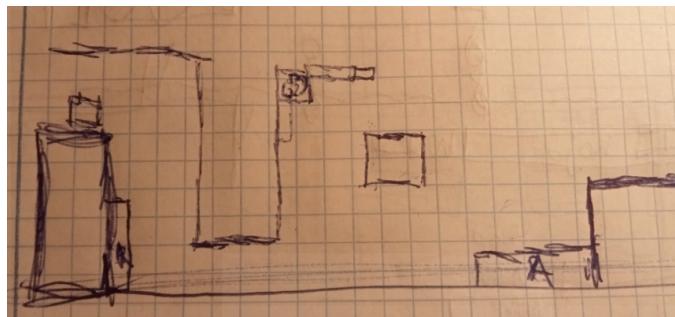
Otra mecánica que quería incluir en Response:Eve era la de poder cambiar la composición del robot a través de las superficies. En un principio pensé en incluir varios tipos de materiales, pero acabé decidiendo que con un material que fuera versátil sería suficiente. Me acabé decantando por la propiedad de la goma, la cual me permitiría cambiar la forma en la que se mueve el personaje porque rebota con las superficies y a su vez le vuelve invulnerable a los obstáculos eléctricos (esto último además tenía sentido físico)



Una vez terminadas las mecánicas me dí cuenta que hacer un nivel solamente dedicado a los imanes iba ser complicado debido a la falta de complejidad de estas mecánicas, sobre todo si quería meter puzzles en cada uno de los niveles. Por ello se me ocurrió la idea del polarizador: una superficie que al activarse cambiase la polarización de los imanes (cambiase el rojo por el azul y viceversa) Esto permitiría presentar situaciones más complejas en las que el jugador tuviera que pensar más.



Una vez diseñadas las mecánicas me puse a hacer prototipos de niveles para comprobar el verdadero potencial de estas.



Después de haber diseñado las mecánicas y algunos niveles tocaba trabajar en el diseño del personaje principal y del entorno. Se me ocurrió la idea de que el personaje principal fuera un robot por dos grandes razones.

La primera es la gran variedad de diseños que un robot puede tener, ya que estos pueden adoptar cualquier forma, tener facciones extrañas, tener un número de extremidades variables... Esto último me resultaba especialmente útil ya que un aspecto fundamental que debía de tener mi personaje es que no tuviera piernas para caminar. Esto es debido a que si las tuviera no tendría sentido que no pudiese desplazarse libremente por el suelo o saltar, y esto es un aspecto fundamental del juego.

La segunda es que encajaba perfectamente con las mecánicas del juego y me permitía crear un contexto simple para este. Si hubiera elegido a un humano como protagonista, tendría que justificar de dónde saca las herramientas, cuál es su objetivo ... En cambio, cuando pensé en un robot como protagonista se me ocurrió la idea de que este estuviera siendo puesto a prueba por un doctor en un laboratorio para probar su eficiencia, y que este estaba diseñado para adaptarse a su entorno. Simple y efectivo.

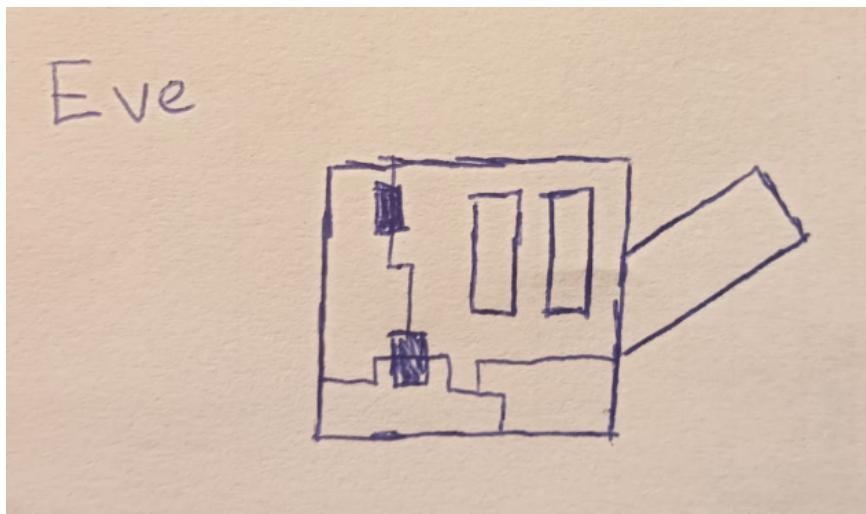
Una vez decidido que mi personaje principal sería un robot empecé a pensar en el diseño de este. Al tratarse de un juego de plataformas tenía claro que mi personaje tenía que ser pequeño y de forma cuadrada o esférica.

Pequeño, porque, en un juego de plataformas, no quieres que tu personaje ocupe mucho espacio en pantalla ya que tu campo de visión tiene que ser el mayor posible para ver los obstáculos. Sí que es verdad que se podría hacer un personaje más grande y alejar la cámara, pero eso requería detallar más al personaje y yo estaba buscando más un estilo minimalista.

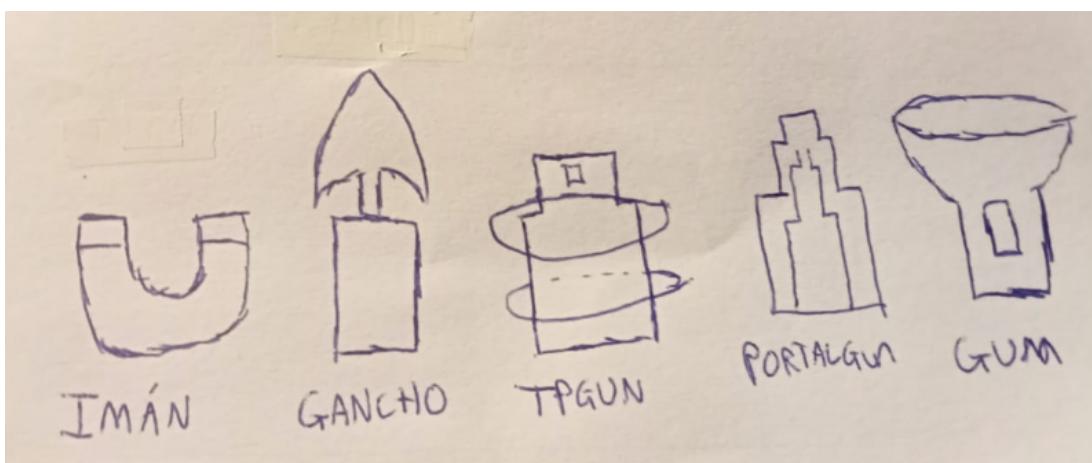
La forma del personaje tenía que ser cuadrada o esférica ya que en un juego de plataformas en el que necesitas precisión de movimiento tanto vertical como horizontal, es importante que el jugador se familiarice con la hitbox del personaje, y es mucho más fácil que esto ocurra cuando el ancho y la altura son iguales. (De hecho la mayoría de juegos plataformeros tienen personajes que miden más o menos lo mismo de ancho que de altura).

Y por último quería que mi personaje tuviera un diseño único que fuera distingible a primera vista. Además quería que fuera lo que algunos calificarían como “mono”, porque como hemos visto estos últimos años en la

industria del videojuego, esos tipos de diseños venden (véase Hollon Knight, Ori o Celeste). Para conseguir esto pensé en simplemente añadirle unos ojos grandes que, junto con su cuerpo diminuto en proporción, debería de ser suficiente para conseguir lo que busco.



Boceto temprano de Eve



Boceto de las herramientas

Y por último, me faltaba saber cómo acabar el nivel. Lo primero que pensé fue en poner una cinta transportadora que al caer encima de ella te llevase al siguiente nivel, pero luego me vino la idea de hacer algo más inesperado para sorprender al jugador. La idea consistía en poner una trampa justo delante de una aparente salida y que cuando te capturase, apareciese el doctor que te está probando para reírse de tí a través de un televisor.



Boceto final de nivel

Aspectos relevantes de la codificación

En este apartado hablaremos de los aspectos del diseño cuya implementación ha sido más complicada o importante.

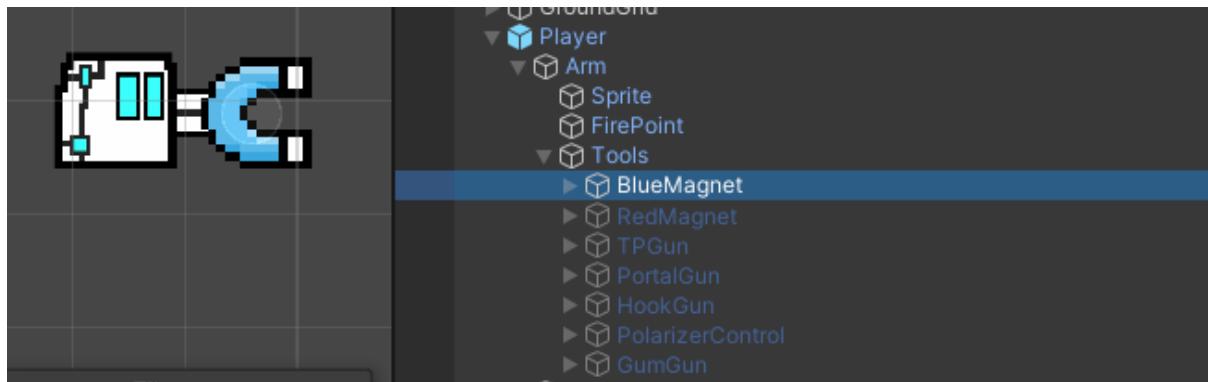
Activación de herramientas mediante el apuntado a superficies

Para empezar hablaremos de como he implementado el sistema de detección de superficies.

El sistema es simple. Dentro del GameObject del Player tenemos el GameObject del brazo, que está programado para que rote de tal forma que siempre apunte a la dirección del puntero o “Scope”, controlado por el ratón. Dentro del brazo tenemos los GameObjects de las herramientas. Cada una de las herramientas tiene un script que determina su función, y todos estos scripts heredan de la clase Tools.

Cada vez que el juego ejecuta un Update() el Player lanzará un raycast en la dirección a la que apunta su brazo y detectará el tag de aquellos objetos de la capa de colisiones del láser con los que choque. Luego el Player mandará estos datos a un script llamado ToolsController. Esta clase se encarga de gestionar la activación y desactivación de las armas y de devolverle al player

la Tool que coincide con el tag que le ha pasado o null en el caso de que golpee una superficie que no tenga asignada ninguna herramienta (como el suelo por ejemplo). Una vez obtenida la Tool, mediante el uso de polimorfismo, podemos hacer que el player llame a la función “UseTool()” de la herramienta, sea cual sea, cuando le dé click al ratón.



Despliegue de portales

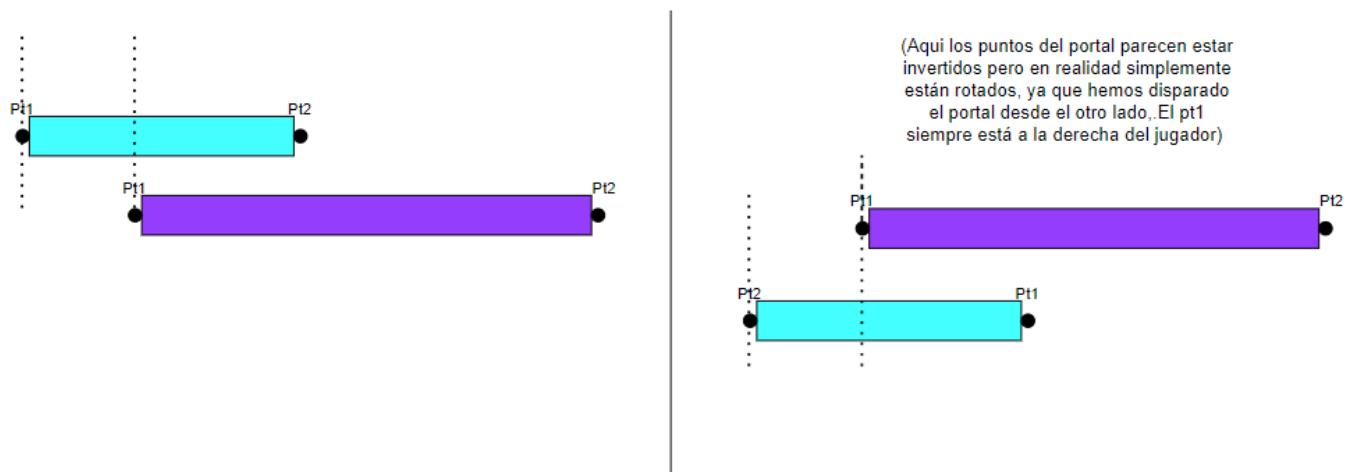
Una de las cuestiones más difíciles de implementar fue la colocación de los portales en las superficies. Cuando se trata de colocar un portal en el centro de una superficie, es tan sencillo como colocarlo en el punto en el que ha impactado el raycast y rotarlo con respecto a la normal de la superficie. El problema viene dado cuando lo colocamos en el borde de una superficie ya que si utilizamos el método anteriormente descrito, el portal sobresaldrá de dicha superficie.



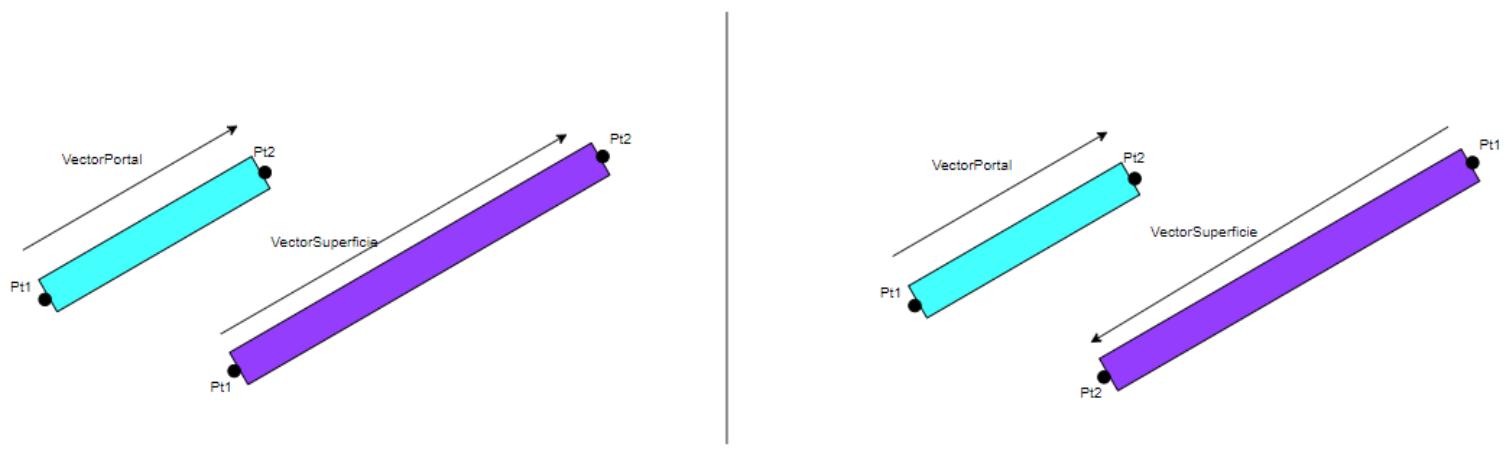
Para solucionar esto, primero teníamos que definir los extremos, tanto del portal como de la superficie. Para ello, simplemente coloqué puntos tanto en los extremos de los portales como de todas las superficies. Definir los extremos así es posible gracias a que solo podemos colocar portales en dos de las caras del rectángulo que conforma la superficie (en el caso de la imagen de arriba serían la cara de arriba y la de abajo). En el caso de que

fueras un cuadrado en el que se pueden colocar portales en cualquiera de sus 4 caras, habría que utilizar el mismo método que voy a describir pero considerando a cada cara como un rectángulo a parte, pero por suerte esto no es necesario.

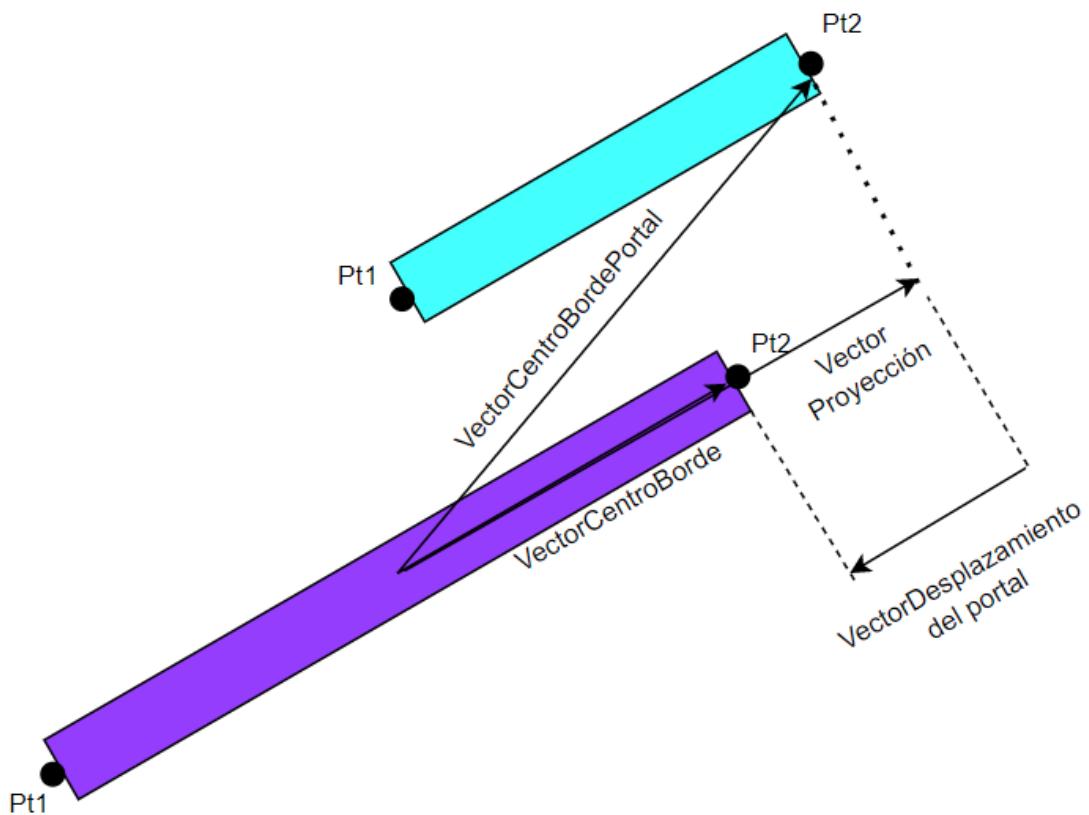
Una vez colocado los puntos en los extremos tenemos que calcular mediante código, que punto del portal hay que comparar con qué punto de la superficie. Si, por ejemplo, solo se pudiese colocar el portal en una de las caras de la superficie, simplemente habría que comprobar si el punto 1 del portal sobresale del punto 1 de la superficie, pero si se coloca en la otra parte de la superficie, entonces tendríamos que comprobar si el punto 2 del portal sobresale al punto 1 de la superficie.



Para emparejar los puntos que se tienen que comprobar simplemente cogemos los vectores que salen como resultado de restar los puntos de los extremos, es decir, los vectores que nos indican la dirección tanto del portal como de la superficie y calculamos el producto escalar de ambos vectores. Si el resultado es mayor que cero significa que están alineados en el mismo sentido (como en la primera imagen), si da menor que cero significa que están colocados al revés (como en la segunda imagen)



Una vez emparejados los puntos que tenemos que comparar, tenemos que definir que significa que un punto sobresalga de otro. Si se tratase de calcularlo en una sola orientación, como la del eje X por ejemplo, simplemente tendríamos que calcular si el punto de la izquierda del portal está más a la izquierda que el punto de la izquierda de la superficie, y moverlo hacia la derecha según esa diferencia de distancias. Pero como la superficie puede tener cualquier orientación lo que tengo que calcular es si la proyección del vector que va del punto central de la superficie al punto del portal sobre el vector que va del centro de la superficie al punto de la superficie emparejado con el punto de portal, es mayor que la de este último vector. Si el valor de la proyección es mayor, entonces desplazar en la dirección contraria el vector resta resultado del vector proyección y el vector y el vector que va desde el centro de la superficie al punto de la superficie (que ya hemos calculado antes).



Y una vez le sumamos al vector de posición del portal el vector de desplazamiento, ya tenemos el portal puesto en el lugar correcto, sin sobresalir de la superficie.

Teletransportación a través de los portales

Una vez terminada con la colocación correcta de los portales en las superficies, tocaba teletransportar al jugador a través de ellos.

El funcionamiento de los portales es el siguiente. Cuando un jugador entra en un portal, si el otro está activado el jugador se teletransporta al otro portal. Cuando este se teletransporta, mantiene su velocidad y posición relativa al otro portal. Esto quiere decir que si entramos en un portal con un ángulo de 45° (con respecto al portal), a una velocidad de 5m/s y medio metro a la izquierda, saldremos del otro portal también a 45°, con la misma velocidad y por su izquierda también. Para conseguir este efecto tuve que seguir los siguientes pasos.

1. Coger el vector velocidad que nos proporciona el Rigidbody del personaje y el vector posición del personaje relativo al portal (restando la posición del personaje y la posición del portal)
2. Cambiar los vectores al sistema de coordenadas del propio portal. Hacer esto de forma manual sería un quebradero de cabeza muy grande, pero por suerte, la clase Transform tiene un método que te permite cambiar cualquier vector o punto del sistema de coordenadas de WorldSpace(el de la escena) a el sistema de coordenadas locales del Transform.
3. Invertir el vector velocidad en el eje x relativo al portal, ya que si no el vector velocidad con el que entra (que está apuntando hacia al portal) sería el mismo con el que sale, cuando debería de ser al revés (el vector de salida debe de estar apuntando hacia afuera del portal)
4. Una vez hecho esto, teletransportamos al jugador al otro portal (siempre que esté activo claro) y hacemos las operaciones que hemos hecho antes de forma invertida, es decir pasar de coordenadas relativas (en este caso del segundo portal y no del primero) a las coordenadas de la escena y aplicarle el vector velocidad y posición resultantes al jugador.

```

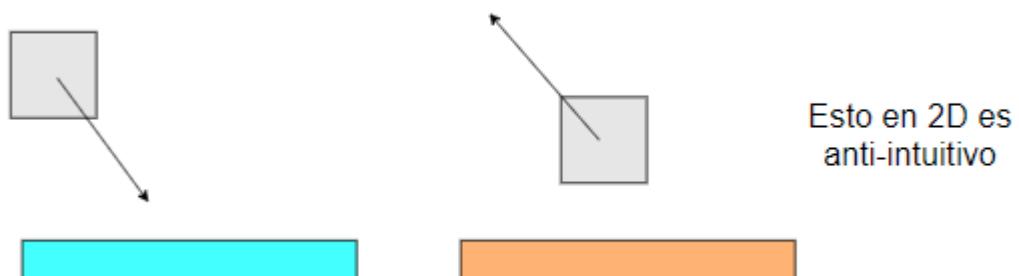
player.transform.position = otherPortal.transform.TransformPoint(this.transform.InverseTransformPoint(player.transform.position));
Vector2 velocityRelativeToFirstPortal = this.transform.InverseTransformVector(velocityFromFrameBefore);
player.rbody.velocity = otherPortal.transform.TransformVector(new Vector2(velocityRelativeToFirstPortal.x,velocityRelativeToFirstPortal.y * -1));

```

Código que realiza lo que he mencionado en los 4 pasos anteriores

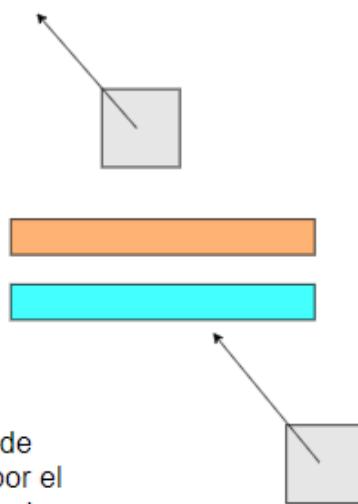
Inversión de la escala de los portales

Una vez programados los portales, me dí cuenta de que, a pesar de que teóricamente funcionaran bien, en un juego de dos dimensiones el resultado de atravesar uno era a veces confuso y antiintuitivo, sobre todo si los portales se encontraban con la misma orientación.

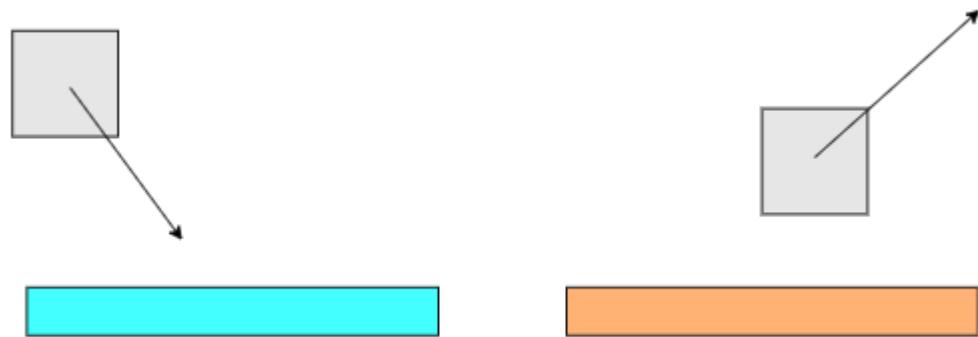


Pero en verdad esta funcionando como debería ya que si juntamos los portales en el mismo plano (como se hace en portal por ejemplo) vemos lo siguiente:

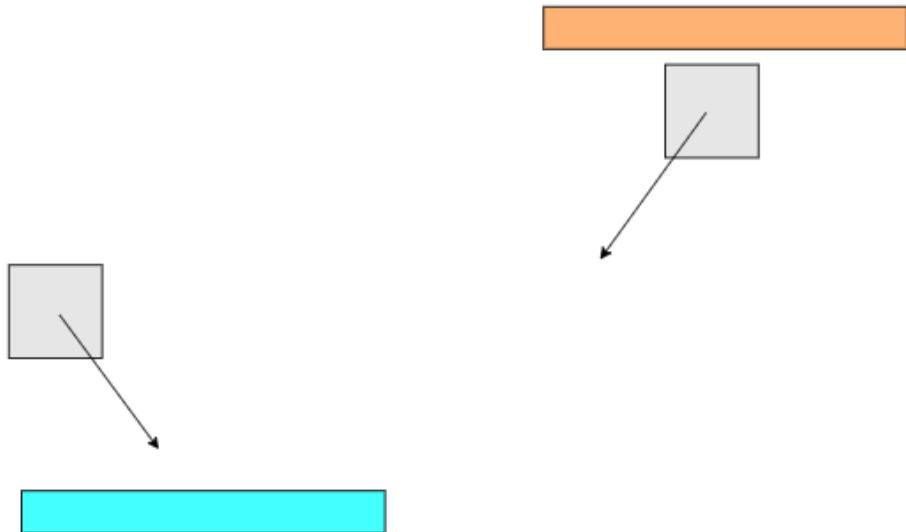
Estamos entrando de derecha a izquierda por el azul y estamos saliendo igual por el naranja



A pesar de que técnicamente estaba bien hecho, el jugador encontraría muy anti-intuitivo que ocurriese lo de la primera imagen. Por ello me decanté por cambiar la escala en el eje x del segundo portal e invertirlo. Ahora el resultado es el siguiente:



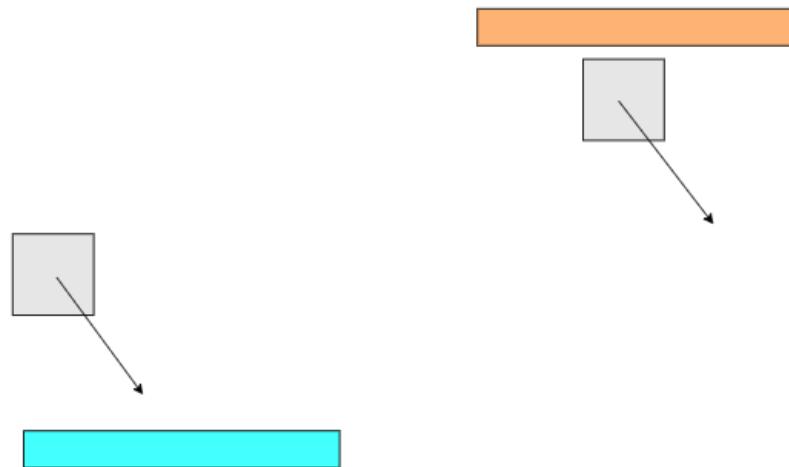
Y justo cuando pensaba que ya lo tenía todo solucionado apareció otro problema. Y es que el cambiar la escala del segundo portal producía una situación anti-intuitiva que no ocurría antes. Ahora al poner un portal con orientaciones contrarias (por ejemplo, uno en el suelo y otro en el techo) se producía el siguiente efecto:



Esto cuando no cambiaba la escala del segundo portal funcionaba de forma intuitiva pero ahora estaba funcionando de forma rara. Durante un momento pensé que sí o sí tenía que decidirme entre que ocurriese este efecto o el que ocurría cuando estaban en el mismo plano. Luego se me ocurrió que si solamente invertía la escala del segundo portal en función del ángulo de la

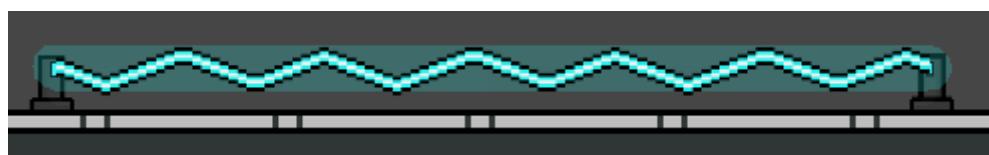
superficie podría hacer que ambas situaciones se dieran de forma intuitiva. Entonces programé que en toda superficie cuyo ángulo fuera menor a 179°, la escala X del portal cambiaría a -1.

Ahora cuando colocaba dos portales en orientaciones opuestas ocurría lo siguiente:



Automatización de los postes eléctricos

Los obstáculos principales del juego son unos postes eléctricos que van por parejas y la electricidad los recorre en línea recta.



El problema con este obstáculo es que colocarlo en el escenario era una tarea que llevaba mucho tiempo ya que: primero tenías que colocar los postes donde tú querías (pueden ir en cualquier dirección y estar a cualquier distancia), colocar el haz de luz, ajustar la hitbox, y lo más difícil, colocar el sprite de electricidad. Esto es debido a que el sprite de electricidad se coloca en forma de "tiles". Esto quiere decir que la electricidad es en verdad un patrón que se repite continuamente (ya que no sabemos la distancia que va a

tener que abarcar). El problema con esto es que si la distancia entre los postes no era un múltiplo de lo que medía el sprite del patrón, el resultado era horroroso (la altura de la corriente no coincidía con la rejilla de salida que tiene el poste). La solución a este problema fue estirar el sprite cambiando su escala de tal forma que el trozo que le faltaba para caber de forma perfecta se salvaguardaba estirándose.

Repetir este proceso para cada rayo me consumía demasiado tiempo, hasta que se me ocurrió que tal vez podría programar un script que hiciera todo este proceso de manera automática.

El script consiste básicamente en recibir dos puntos (los de los postes) y a partir de su vector distancia calcular la longitud y la rotación de la hitbox, la luz y el sprite de la electricidad.

```
void PutElectricity()
{
    electricity.transform.position = point1.position;
    electricity.transform.localRotation = Quaternion.FromToRotation(electricity.transform.right, point2.position - point1.position) * electricity.transform.localRotation;
    int numberOfraps = Mathf.FloorToInt(distanceBetweenPts / electricSpriteRepeatDistance);
    float sizeWithoutStretch = numberOfraps * electricSpriteRepeatDistance;
    electricity.GetComponent<SpriteRenderer>().size = new Vector2(sizeWithoutStretch, 2.56f);
    electricity.transform.localScale = new Vector3( distanceBetweenPts / sizeWithoutStretch, 0.4103171f, 1f);
}

void PutAuraline()
{
    auraLine.SetPosition(0, point1.position);
    auraLine.SetPosition(1, point2.position);
}

void PutHitbox()
{
    hitboxCollider.transform.position = point1.position;
    hitboxCollider.transform.localRotation = Quaternion.FromToRotation(hitboxCollider.transform.up, point2.position - point1.position) * hitboxCollider.transform.localRotation;
    hitboxCollider.size = new Vector2(0.3749752f, distanceBetweenPts);
    hitboxCollider.offset = new Vector2(0f, distanceBetweenPts / 2f);
}
```

Este script me permitió ahorrar una gran cantidad de tiempo y me proporcionó una comodidad increíble a la hora de desarrollar niveles ya que, ahora, para colocar un obstáculo era tan fácil como colocar los dos postes en el lugar en que los quería y ya.

Pruebas de funcionamiento

Se han realizado dos grandes pruebas a Response:eve, a parte de las que hago yo de forma continua. Las pruebas han consistido en poner a dos sujetos a jugar el videojuego de principio a fin y evaluar su experiencia.

En cuestión de bugs, solo el segundo sujeto experimentó uno en el que al morir no reaparecía en el punto de control y otro en el que la cámara se movía de forma extraña al hacer una transición. Ambos bugs han sido solucionados.

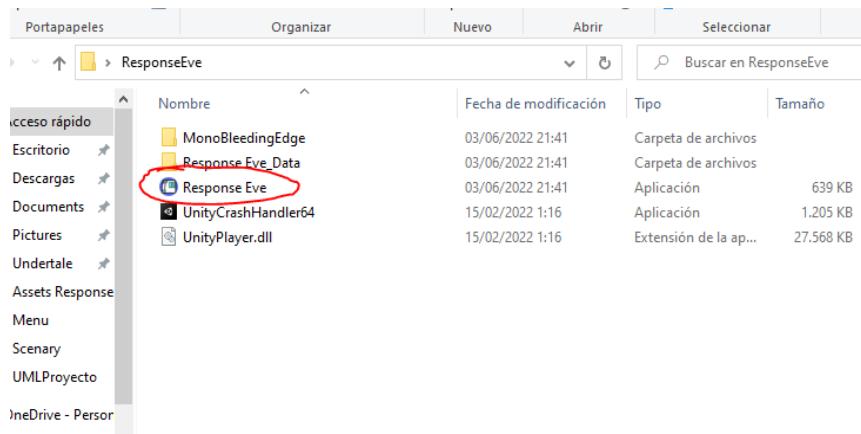
A nivel de manejo por la interfaz, ninguno de los sujetos tuvo problemas para moverse por ella y ambos las describieron como “atractiva”.

A nivel de experiencia de usuario, mientras jugaban al juego el primero describió la experiencia como frustrante en algunos tramos pero divertida por lo general. Al segundo sujeto le gusto mucho más y lo describió como un juego desafiante pero gratificante, con puzzles sencillos pero bien pensados. Ambos destacaron que les gustó la estética minimalista de los entornos y del personaje.

Manual de despliegue

La forma de ejecutar el proyecto es muy sencilla. Para ello tienes que seguir los siguientes pasos:

1. Descomprimir el archivo.
2. Abrir la carpeta..
3. Ejecutar el archivo ResponseEve.exe.



Conclusiones

Soy consciente de que mi juego tiene muchas limitaciones, tanto artísticas como a nivel de diseño del propio juego. Hay pocos sprites, falta pulido en los sprites de los escenarios, las mecánicas del juego no están mal pensadas del todo pero pueden llegar a ser un poco toscas, el nivel de dificultad puede ser demasiado alto y el juego puede llegar a ser frustrante para algunas personas. Por lo general esto último siempre me pasa, siempre me cuesta balancear mis juegos porque a mí no me parecen tan difíciles. Aún con todo esto, estoy medianamente contento con el resultado. He sido capaz de resolver muchísimos problemas a nivel de código y a nivel de diseño, y el programar cosas como los portales me ha dado mucha confianza en mis capacidades para desarrollar mecánicas. Artísticamente, el resultado me ha sorprendido para bien, ya que para no ser un artista y dárseme mal el dibujar, no ha quedado del todo mal (sobre todo el diseño de mi personaje). Supongo que el elegir una estética simple me ha ayudado bastante. También estoy muy contento con cómo han quedado los menús del juego.

Tras terminar de desarrollar Response:Eve he llegado a la conclusión de que, efectivamente, hacer videojuegos es difícil, sobre todo si lo haces sólo. Encargarse de dibujar todos los sprites y animaciones, programar todas las mecánicas, diseñar niveles que sean interesantes, arreglar bugs tanto causados por mí como por el motor del juego, organizar bien las clases, diseñar y programar la interfaz, buscar durante multitud de horas en librerías de efectos sonoros en busca de aquellos sonidos que se adapten

perfectamente al juego, diseñar herramientas para la construcción de niveles... han sido cosas que he tenido que hacer por mi cuenta en estos últimos tres meses. Aún con toda esta cantidad de trabajo puedo decir que he disfrutado mucho del proceso.

Desde mi punto de vista, los seres humanos somos seres creativos. Hay algo inherente al hecho de crear cosas que salen de nuestra mente que nos llena, sobre todo si se trata de algo que nos gusta. Siempre he considerado a los videojuegos como una parte fundamental de mi vida y llevo desde pequeño jugándolos y creandolos en mi cabeza. El año pasado decidí pasar de crearlos en mi cabeza a crearlos de verdad y me lancé a aprender Unity. Durante el proceso descubrí que hacer videojuegos me encanta, y aunque sé que probablemente no me pueda dedicar a ello, tengo claro que seguiré haciéndolo como hobby durante muchos años.

Webgrafía

-<https://forum.unity.com/>

-<https://stackoverflow.com/>

-<https://docs.unity3d.com>

-<https://mixkit.co/free-sound-effects/game/>

-<https://freesound.org/browse/tags/game-sound/>