Universidad San Jorge

Escuela de Arquitectura y Tecnología

Grado en Diseño Y Desarrollo de Videojuegos

Proyecto Final

Estudio del efecto de diferentes parámetros de diseño de videojuegos en el *Game Feel* o Sensación de Juego.

Autor del proyecto: Alejandro Berraondo

Soria

Director del proyecto: Daniel Blasco Latorre

Zaragoza, 9 de Septiembre de 2021

universidad SANJORGE
GRUPO SANVALERO

Este trabajo constituye parte de mi candidatura para la obtención del título de Diseño y Desarrollo de Videojuegos por la Universidad San Jorge y no ha sido entregado previamente (o simultáneamente) para la obtención de cualquier otro título.

Este documento es el resultado de mi propio trabajo, excepto donde de otra manera esté indicado y referido.

Doy mi consentimiento para que se archive este trabajo en la biblioteca universitaria de Universidad San Jorge, donde se puede facilitar su consulta.

Firma **Fecha**

11/07/2021

Dedicatoria y Agradecimiento

Me gustaría agradecer a todo el mundo involucrado en mi vida su ayuda tanto activa como pasiva en la realización de este proyecto, a mi familia y amigos por el apoyo moral y por transmitirme su energía en los momentos más difíciles tanto de la carrera como de este proyecto.

Querría también agradecer especialmente a Daniel Blasco Latorre toda la ayuda que me ha aportado, no solo durante este trabajo, sino también como tutor y por su docencia durante todos estos años en las distintas asignaturas de las cuales he sido alumno.

Tabla de contenido

Tabla de figuras

Tabla de tablas

Tabla de términos técnicos

Resumen

A la hora de crear un videojuego debemos tener en cuenta el efecto que queremos generar en el público que lo compra ya que si no logra transmitir las sensaciones prometidas puede generar que nuestro juego fracase o acabe siendo mediocre. Este efecto o sensaciones que buscamos generar puede ser conseguido de muchas maneras diferentes, ya sea a través de sistemas dentro del propio juego o incluso desde la manera en la que el jugador interactúa con nuestro videojuego a través de, por ejemplo, mandos o pantallas táctiles.

El objetivo principal de este trabajo es crear y comparar una demostración de un juego de terror con otros proyectos anteriores o juegos del mercado triple A para contrastar y demostrar que el diseño de una buena sensación de juego puede cambiar completamente como nuestro proyecto es visto por los consumidores. Esta demostración será construida desde cero y cada decisión tomada que pueda cambiar o afectar a la sensación de juego del consumidor será elaborada y explicada con todo lujo de detalles usando métodos ya existentes o ejemplos ya creados de la industria del videojuego.

Abstract

When creating a video game we must take into account the effect we want to generate on the public that buys it, since if it fails to convey the promised sensations, it can cause our game to fail or end up being mediocre. This effect or sensations that we seek to generate can be obtained in many different ways, either through systems within the game itself or even from the way in which the player interacts with our video game through, for example, controls or touch screens.

The main objective of this work is to create and compare a demo of a horror game with other previous projects or games on the triple A market to contrast and show that the design of a good game feeling can completely change how our project is seen by the consumers. This demo will be built from scratch and every decision made that may change or affect the consumer's gaming sensation will be elaborated and explained in great detail using existing methods or ready-made examples from the video game industry.

1. Introducción

Para entender la Sensación de Juego debemos primero encontrar una definición para esta puesto que es algo que puede ser confundido con otros términos que se usan dentro de la industria del videojuego para definir términos similares. Cada diseñador de videojuegos puede tener una definición para este término, pero la que más poder tiene dentro de las creadas y la más popularizada es la que podemos encontrar en el libro de Steve Swink 'Game Feel: A Game Designer's Guide to Virtual Sensation' que define Sensación de Juego como:" Real-time control of virtual objects in a simulated space, with interactions emphasized by polish." [1].

Por ello siempre que se habla de Sensación de Juego se suele hacer referencia a Steve Swink ya que gracias a esta definición y a su libro tenemos una definición bastante acertada de lo que la Sensación de juego es y lo que implica para el jugador.

La definición de Sensación de Juego no es fija, pero en lo que la mayoría concuerda es en que el objetivo de la creación de una buena Sensación de Juego es favorecer siempre la inmersión del jugador para que siempre se sienta dentro del juego. Esto es realmente complicado ya que cualquier mínimo detalle que no funcione como debe sacará al jugador de este estado de inmersión, y si esto sucede demasiado es posible que pierda completamente el interés en nuestro proyecto.

La Sensación de juego es principalmente algo subconsciente, algo que sucede debido a la combinación de múltiples variables de diferentes aspectos. De nuevo, Steve Swink en un artículo para Gamasutra comenta: "Game Feel is a combination of sights, sounds, and instant response to action. It's one of those 'know it when you feel it' kinds of things. If it's off by just a little bit, a game's goose is cooked. If it's "responsive", "tight", and "deep", it can be magical. " [2], y es que tiene mucha razón, la Sensación de Juego es uno de los aspectos que más se suele ignorar a la hora de crear un videojuego y es uno de los apartados dónde más tiempo hay que invertir.

Podemos también tener en cuenta lo que dice Brandon A Kidwell de la Sensación de Juego "Game Feel is an interconnected web of systems, animations, SFX, VFX, and more. There are many different ways to accomplish the desired Game Feel and there is a layering effect that improves the feel with each layer. These layers are affected by budget, time, competency.

There is a base layer though, and even without the additional layers, the game should feel great. If I removed all of the sounds and visual but left you with a black box that can move around and shoot other black boxes it should still feel good. "[3]. Y es que como bien dice, si quitamos todo el pulido de un videojuego, nos quedamos con el esqueleto del proyecto (en lugar de personajes capsulas etc.) el cual debería seguir siendo funcional si aprovechamos al máximo lo que podemos aportar aplicando un correcto uso de la Sensación de Juego, pero puede que el jugador no sienta lo mismo.

Teniendo las palabras de Steve Swink en cuenta se puede entender con un ejemplo muy simple de *que sirve aplicar una* correcta Sensación de Juego a un videojuego comercial. Shigeru Miyamoto, el creador de Super Mario o The Legend of Zelda, es todo un referente en lo que a Sensación de Juego se refiere. Desarrollando Super Mario 64 [4] se encontraron con el desafío de crear el primer juego de Mario en tres dimensiones. Para ello, Miyamoto quiso estar seguro de que controlar a Mario era una sensación placentera incluso sin tener ningún nivel creado, para ello estuvieron tuneando su movimiento hasta que el simple hecho de andar y saltar en un espacio sin desafíos de plataformas alguno fuese satisfactorio.

Miyamoto, también comenta que en los niveles acuáticos no quería que el jugador viese tedioso tener que sumergirse en el agua, sino que fuese una manera más de atravesar un nivel complementaria a como lo haría en un nivel sin agua. Centrarse en como Mario debía sentirse siendo controlado por el jugador hizo que Super Mario 64 sea uno de los juegos mejor valorados de la historia y que se sigue teniendo en cuenta incluso 25 años después de su lanzamiento.

Comparando el sistema de plataformeo de Super Mario 64 con el primer Tomb Raider basado en controles de tanque salta a la vista que Miyamoto fue un visionario que vino a romper todos los esquemas que llevaban siguiendo los videojuegos. Pero, como ya hemos mencionado, la Sensación de Juego puede aplicarse también a los controles, y es que, que el primer Tomb Raider tenga controles de tanque añade cierta dificultad extra que hace que navegar por tumbas y templos llenos de trampas de verdad se sienta un desafió digno de una saqueadora de tumbas profesional, pero el tiempo le dio la razón a Mario y los controles de tanque poco a poco fueron desapareciendo haciendo así que la dificultad que suponía manejar a un personaje con controles de tanque fuese trasladada a otras facetas del diseño del videojuego como el diseño de niveles.

Cierto es que cada sistema de movimiento se adaptó al método con el cual el jugador interactuaba con el mundo digital. En el momento en el que Tomb Raider salió al mercado la PlayStation original aún no tenía un mando con un *thumbstick* [1] a diferencia de la Nintendo 64, por lo cual su sistema de movimiento tuvo que adaptarse al que podía ofrecer un *d-pad* [2] en un entorno tridimensional.

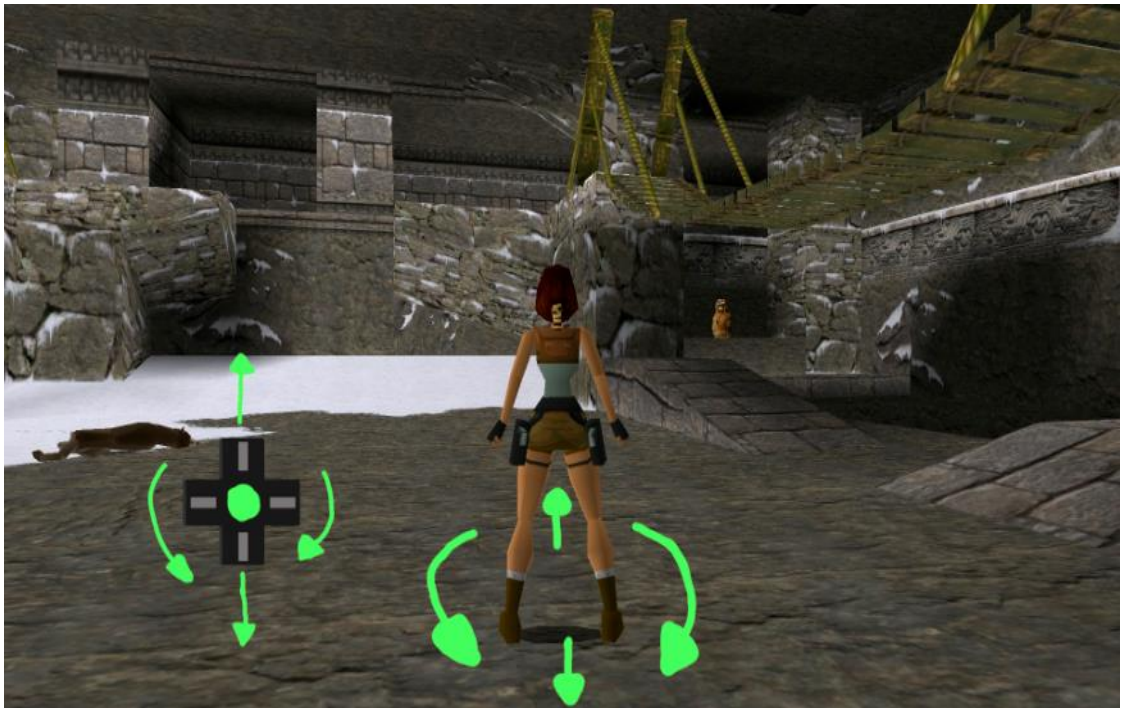


Figura 1: Tank Controls de 'TOMB RAIDER' (1996)

Con el d-pad simplemente podíamos desplazarnos hacia delante o hacia atrás, los botones derecha e izquierda simplemente hacían que Lara pivotase en la dirección que pulsásemos por lo que momentos de plataforma que requerían una agilidad un poco más elevada a la que te acostumbra el juego se hacen realmente agobiantes.

Como se ha comentado anteriormente, siguieron manteniendo este esquema de movimiento en futuras entregas debido a la sensación que generaba en el jugador. Como podemos ver nos podemos remontar a ejemplos realmente viejos y es que la Sensación de Juego lleva estando con nosotros desde la creación del primer videojuego de la historia. En PONG el movimiento de las palas se ajustaban a un input bastante bien pensado para el juego, pero se fue adaptando a otros esquemas de controles conforme iban saliendo versiones posteriores del juego.

Algunos de los nuevos esquemas de control hacían que el juego fuese más difícil y menos preciso que con el esquema original y es que como ya hemos visto con los ejemplos anteriores, el cómo recibe un input el juego es realmente importante para transmitir una Sensación de Juego deseable.

Realmente, y como cuenta Steve Swink en su definición, necesitamos 3 grandes pilares para poder definir sensación de juego, y en el caso de PONG nos falta uno de esos pilares o su representación es un poco limitada debido a la época de salida del juego, estoy hablando de la simulación de un espacio. Se podría interpretar que en PONG estamos en una mesa de ping pong simplificada a su máxima expresión, lo cual tendría sentido, pero deja demasiados a la imaginación.

Como veremos, esto no significa que un juego sea malo o bueno, simplemente en nuestra definición de Sensación de Juego carecería de uno de los pilares en la que la basamos y por lo tanto podíamos decir que a el juego que le falten uno o más pilares no tiene Sensación de Juego.

El paso del tiempo es algo que puede afectar a la Sensación de Juego de un videojuego, pero en algunos casos como el de Super Mario 64 puede seguir dando la misma sensación que daba antaño puesto que ningún otro juego plataformas (Que no sea otro super Mario posterior) se puede comparar a la innovación y sensación de movimiento que nos brindó Miyamoto en el 96. El hardware se ha ido perfeccionando con el tiempo para ser cada vez más intuitivo

En este Trabajo de Fin de Grado ahondaremos principalmente en el pilar de la Sensación de Juego llamado "Real-Time Control" o Control en Tiempo Real y como una simple variable como puede ser la velocidad máxima que nuestro personaje puede alcanzar, puede cambiar completamente como el consumidor percibe el juego. Para ello compararé sistemas de juegos de la industria y de juegos similares creados por mí anteriormente para demostrar como esta serie de detalles que normalmente no se suelen explicar ni enseñar pueden afectar de manera drástica en el resultado de nuestro proyecto.

A la hora de crear mi demostración de Sensación de Juego he usado un sistema similar al descrito en el artículo de Steve Swink llamado "The Garden's Ecosystem" el cual consiste en realizar los siguientes pasos a la hora de testear y crear nuestra Sensación de Juego.

Swink describe este ecosistema de jardín como una zona de juego aislada donde testear mecánicas de juego de manera aislada para evitar que posibles problemas referentes a la Sensación de Juego se mezclen y empeoren otros aspectos de nuestro juego. Los aspectos de la Sensación de Juego se pueden dividir de diferentes maneras, pero tanto en su libro como en este artículo Swink divide estos aspectos en los siguientes puntos (los cuales definiremos más tarde con detalle y veremos cómo medirlos):

"1-Input -- How the player can express their intent to the system.

2-Response -- How the system processes, modifies, and responds to player input in real time.

3-Context -- How constraints give spatial meaning to motion.

4-Polish -- The interactive impression of physicality created by the harmony of animation, sounds, and effects with input-driven motion.

5-Metaphor -- The ingredient that lends emotional meaning to motion and provides familiarity to mitigate learning frustration.

6-Rules -- Application and tweaking of arbitrary variables that give additional challenge and higher-level meaning to motion and control." [2].

En la siguiente figura podemos observar estos apartados de la Sensación de Juego relacionados con las diferentes partes del campo de percepción.

Swink, de nuevo, no garantiza que esta sea la mejor manera posible de analizar la Sensación de Juego posible pero actualmente es la que más sentido tiene y la más estandarizada entre desarrolladores de videojuegos además hay más posteriormente.

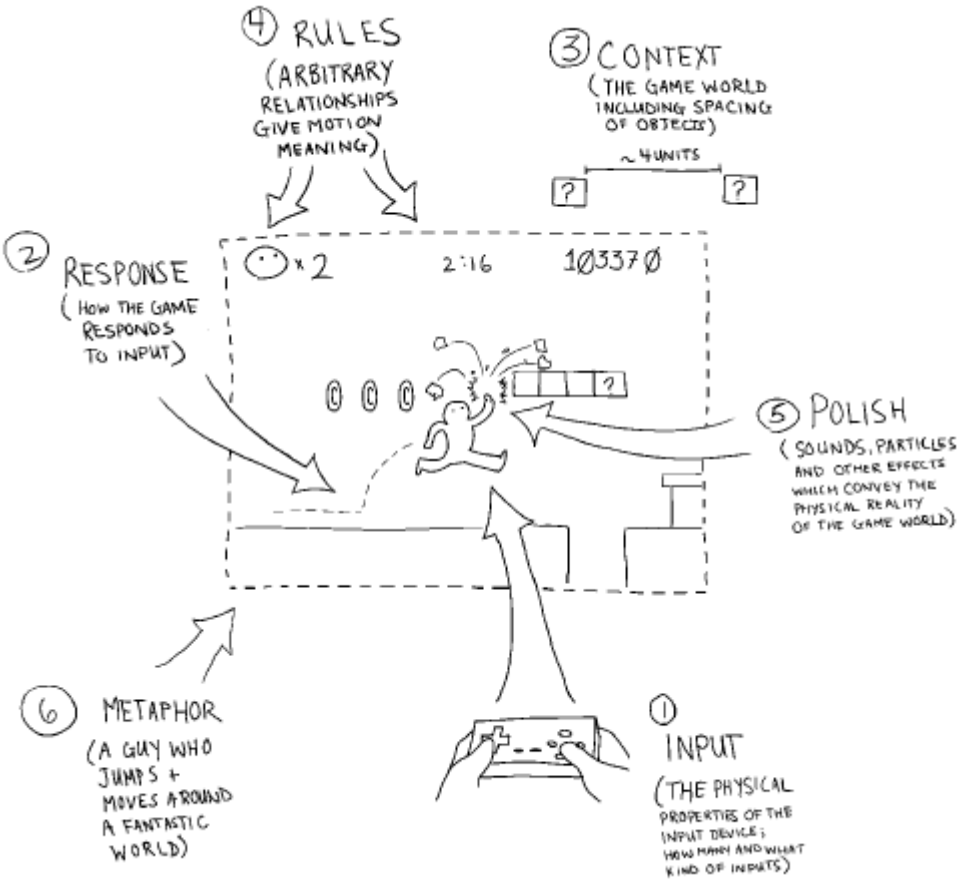


Figura 2: Componentes de la Sensación de Juego 'Game Feel: A Game Designer's Guide to Virtual Sensation' (2008)

En este Proyecto de Fin de Grado estudiaremos las diferentes variables capaces de alterar la capacidad perceptual del jugador en la Sensación de Juego. También se dividirán y se aplicaran los tres pilares principales de la Sensación de Juego definidos por Steve Swink, los cuales se pueden observar en la Figura 3, además en la Figura 4 se pueden observar los objetivos propuestos y que se pretenden alcanzar con la realización de este proyecto.

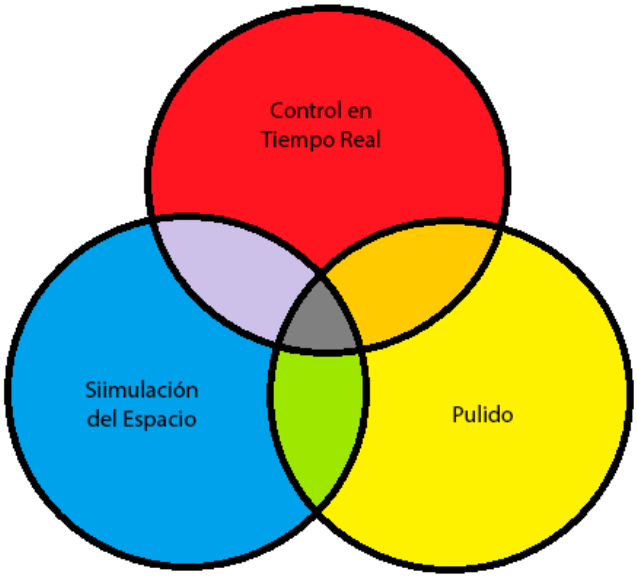


Figura 3: Pilares de los que se compone la Sensación de Juego, basada en la figura de Steve Swink en su libro Game Feel: A Game Designer's Guide to Virtual Sensation (2008).

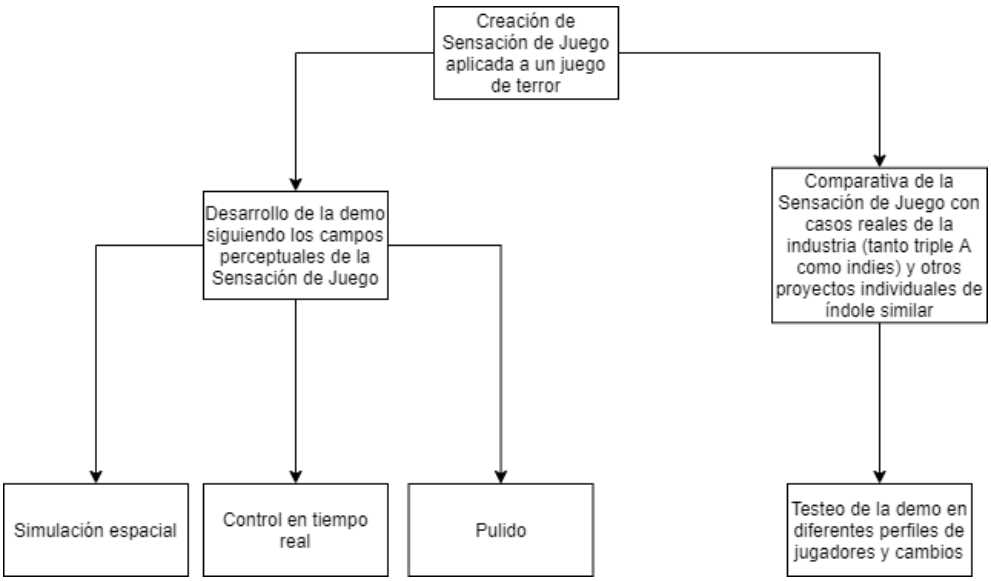


Figura 4: Metas y objetivos de este proyecto.

El proyecto está dividido en dos ramas: el desarrollo de una demostración jugable de un juego de terror creado desde cero y aplicando todas las bases de las que se forma la Sensación de Juego.

En la segunda rama haré un caso de estudio de cómo implementar correctamente una Sensación de Juego correcta basándome en ejemplos del mercado, estos ejemplos serán principalmente similares a lo que se pretende mostrar en la demostración jugable, pero si es necesario se usarán ejemplos de otros géneros videojugables.

2. Estado del arte

En esta sección voy a explicar cómo funciona el modelo del campo de percepción de la Sensación de juego y en qué consisten cada uno de los pilares de la misma los cuales han sido mencionados en la sección de introducción. Real-Time control (control en tiempo real), Spatial Simulation (simulación del espacio o entorno) y Polish (pulido) y como crear un campo de percepción adecuado para el jugador modificando cada uno de sus parámetros de manera adecuada.

Al ser temas que entre sí forman un todo, pero están compuestos de diferentes apartados, los dividiremos en 3 apartados explicando primero con detalle como son y funcionan todas las partes del campo de percepción y los tres pilares de la Sensación de juego.

2.1. Modelo de interactividad: el campo de percepción.

El modelo de interactividad de la Sensación de Juego tenemos varios elementos que analizar comenzando con la parte humana la cual denominaremos Procesamiento Humano. Este está formado por tres procesadores: el perceptual, el cognitivo y el motor. Estos están unidos entre sí ya que componen el ciclo desde que una persona percibe un suceso hasta que responde, este tiempo no debe ser superior a 240ms (100ms de procesamiento perceptual, y 70 en procesamiento cognitivo y motor) para mantener la ilusión de que todo sucede en respuesta a algo que ha realizado el jugador.

Estas medidas de tiempo son rescatadas del modelo de procesador humano de Stuart K. Card, Thomas P. Moran, & Allen Newell [5] el cual nosotros aplicaremos en contexto de la Interacción Persona Computador.

Swink resume esto en hacer creer al jugador constantemente que todas sus acciones tienen una respuesta o consecuencia "If an action passes out through the motor processor and the response is perceived in the same perceptual frame, there is a strong bias toward experiencing action and response as causal ("My action caused this result ")." [1].

Si el proceso dura más por diferentes causas como *input lag* [3] del propio sistema de juego o el mal funcionamiento del programa, se rompe esta sensación de causa efecto generada en el jugador necesaria para crear una Sensación de Juego fluida.

Los músculos serían la segunda parte de todo este proceso. Simplemente se ocupan de traducir los impulsos recibidos por el procesador humano al mundo real. También, en el caso de nuestros dedos (ya que es con lo que normalmente interactuamos en un videojuego) trasmiten *feedback* [4] al procesador perceptual mediante el sentido del tacto y propiocepción.

A partir de la tercera parte ya entraríamos en todo lo relacionado con lo no humano (menos que sería puramente humana, pero depende de cómo hayan reaccionado las partes no humanas), es decir, entraríamos en una zona del campo de percepción donde se pueden modificar los medios y los resultados (en el caso de los humanos sería imposible cambiar cualquiera de nuestro sistema de procesamiento o como nuestros músculos reaccionan a estímulos). Comenzamos con el método de input, es decir, con que interactúa el jugador para traducir sus intenciones de su cabeza al mundo ficticio. Cada input que se use tendrá sus pros y sus contras a la hora de representar como se va a controlar nuestro juego y por ende pueden jugar un papel importante a la hora de crear una Sensación de Juego correcta.

En el cuarto punto encontramos lo que Swink define como Computador, pero realmente se refiere a cualquier sistema en el cual estemos creando nuestro software (en este caso un videojuego). Sobre la Sensación de Juego comenta lo siguiente: "For game feel to occur uninterrupted, input from the player's muscles needs to travel through the controller, be processed and come back as changes in pixels and sounds before one entire cycle of the player's perceptual processor has finished." [1]. Para que esto se pueda conseguir, el computador debe realizar su ciclo más rápido de lo que el jugador pueda percibirlo para que el movimiento que el jugador observa en la pantalla sea percibido como la reacción por su input a través del mando.

La quinta parte del diagrama es el mundo del juego. El mundo del juego existe principalmente en la mente del jugador ya que el mundo del juego que existe en la consola es matemático y concreto. La proyección de este mundo que tiene el jugador en su mente son las experiencias que vive mientras juega mediante el uso de su consola y periféricos como la pantalla que serán el nexo entre ambos (jugador y mundo del juego). "A game world slots itself into the player's action → perception → cognition cycle, replacing the physical world's roles of accepting input and returning feedback". [1] Siendo el feedback devuelto el resultado de la acción realizada a través del sistema de input.

La penúltima y sexta parte del diagrama se trata de los periféricos de output. Estos han sido brevemente mencionados en el punto anterior y son aquellos que hacen posible que el jugador sienta los juegos. Mandan feedback visual auditivo e incluso aural al jugador mediante los diferentes tipos que existen (altavoces, monitores, mandos…).

Un ejemplo de cómo estos sistemas son útiles sería el siguiente. Siguiendo todos los pasos hasta ahora, un jugador avanza en super Mario 64 a una velocidad superior a la necesaria para realizar un salto preciso a una plataforma, como el jugador recibe esta información en un periodo de tiempo lo suficientemente rápido reacciona dejando de pulsar o arrastrar con tanta fuerza el botón de movimiento del personaje para así perder velocidad y poder realizar el salto.

La última parte serían los sentidos los cuales simplemente perciben los cambios en el mundo del juego realizados anteriormente por el jugador mediante el tacto, visión, audición y propiocepción.

Gracias a este modelo podemos crear Sensación de Juego desde cero e incluso podríamos diferenciar y trazar una línea entre los juegos que tienen Sensación de juego y lo que no tienen.

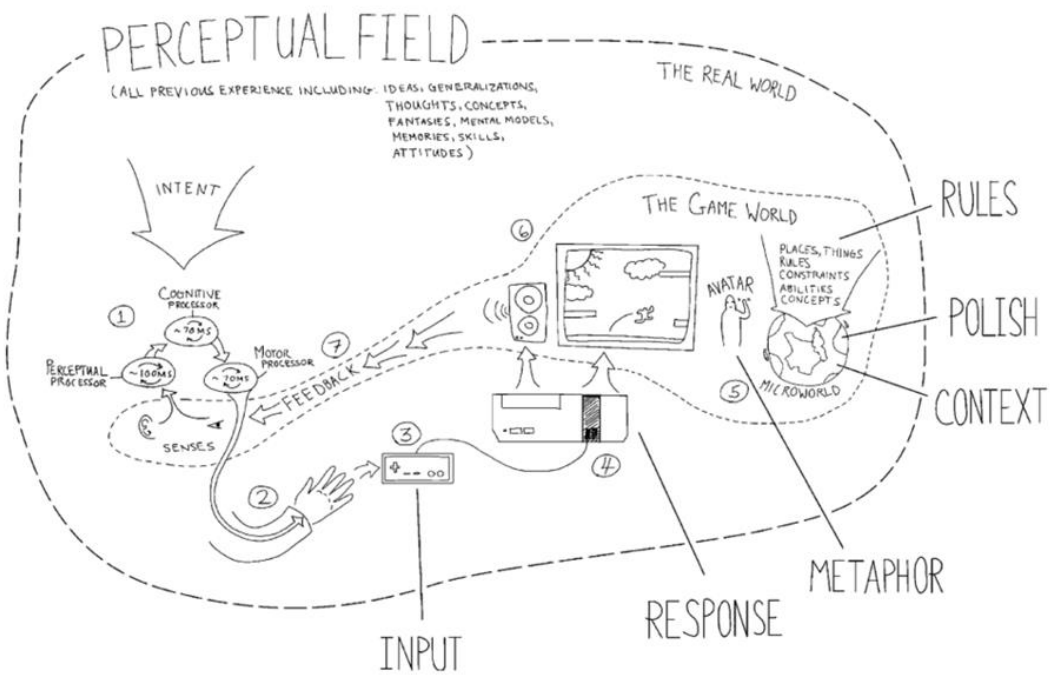


FIGURE 3.1 **The model of interactivity brings together all the elements of the the gamer, the game and the world around him or her.**

Figura 5: Campo perceptual y los componentes del Game Feel 'Game Feel: A Game Designer's Guide to Virtual Sensation' (2008)

2.2. Los tres pilares de la Sensación de Juego

Swink una vez menciona los 3 pilares iniciales comenta que cualquier juego que no contenga en alguna medida parte de los tres no tendrá sensación de juego. Realmente, esto dejaría fuera gran cantidad de juegos, los cuales según este criterio ya estarían fuera de tener Sensación de Juego.

En nuestro caso esta definición es plenamente informativa ya que en nuestro ejemplo todos los pilares estarán presentes, pero para poder aplicarlos habrá primero que conocer bien que se componen cada uno de ellos y como se relacionan.

Empecemos por la simulación del espacio, simplemente, es tener un espacio reconocible en el cual se junta nuestro juego toma lugar. Sin esto, no podríamos decir con seguridad si nos estamos moviendo o no en un juego ya que no tendríamos puntos de referencia que tomar. No hace falta que sean hiperrealistas, como bien podemos ver en juegos de NES, perfectamente pueden ser reducidos al absurdo y dejar que la mente del jugador rellene los huecos como, por ejemplo, en el anteriormente mencionado PONG. Juegos que simplemente tengan este pilar son realmente escasos, podríamos mencionar a Polybridge donde no existe control en tiempo real ya que toda acción es simulada una vez el puente ha sido construido.

El segundo pilar a tener en cuenta es el pulido, suele ser la última parte que se realiza a la hora de hacer un juego. Son las animaciones, partículas, texturas, sonidos etc, que se añaden para dar personalidad a un juego.

Aquí encontramos lo que más puede aportar al 'Juice' de un videojuego (o españolizando el termino, darle sustancia). Y, ¿qué es el Juice de un videojuego? Es un término ligado al Game Feel y tampoco tiene una definición exacta, cuando hablamos de juice nos referimos a grandes cantidades de feedback visual enviados al jugador para crear una sensación positiva en el mismo. Gabler, Kyle, Kyle Gray, Matt Kucic, y Shalin Shodhan mencionan en su artículo de gamasutra de cómo prototipar un juego en 7 días "It makes the player feel powerful and in control of the world, and it coaches them through the rules of the game by constantly letting them know on a per-interaction basis how they are doing." [6] "It" siendo el Juice.

Aunque sea lo último que explican en su artículo el juice (y los pilares para la Sensación de Juego) tienen un desarrollo paralelo.

Encontrar juegos solo con este pilar presente es complicado, pero no tanto, podemos hablar de juegos como Candy Crush, donde, no hay un espacio simulado como tal y no hay control en tiempo real (una vez mueves un caramelo no hay corrección posible, aunque tenga algunas de las condiciones que comprende el control en tiempo real, no cumple todas por tanto no podemos decir que tenga).

Y por último y el más importante para este proyecto, el pilar del control en tiempo real. Swink no le da más importancia que a otros pilares, pero cualquier modificación de este pilar puede ser tan importante que incluso puede cambiarnos el género del juego que estemos creando o ser motivo de devolución en tienda por no estar lo suficientemente bien trabajado, esto puede pasar con los otros dos pilares, pero este es el que más afecta al gameplay de manera directa.

¿Cómo podemos saber si nuestro juego tiene control en tiempo real? Swink tiene tres condiciones que sirven para todos los juegos, pero sería muy difícil aplicarlo en el esquema de interactividad de un videojuego, por lo que lo mejor es dividir nuestro juego en mecánicas y luego contrastarlos con esas condiciones para determinar si nuestro juego tiene control en tiempo real o no. Estas condiciones son las siguientes:

1. Necesitamos un mínimo de diez cuadros por segundo para que las acciones mostradas en la pantalla parezcan relacionadas entre sí.

2. Un tiempo de respuesta menor a 100ms.

3. Un bucle de feedback continuo

Realmente no existen juegos en los que solo exista control en tiempo real para poner de ejemplo ya que este siempre viene acompañado de alguno de los otros dos pilares para sostenerse.

2.3. Métricas para el Game Feel, ¿Cómo se mide?

Como hemos podido ver anteriormente, el Game Feel se compone de varios apartados los cuales podemos medir. El input, la respuesta, la metáfora, el contexto, el pulido y las reglas.

En el Input podemos medir varias cosas, en nuestro caso argumentaremos el porqué de los inputs que usaremos en nuestra demostración, el tipo de controlador dependiendo de las dimensiones en las que nuestro juego va a desarrollarse (2D, 2.5D, 3D puesto que algunos controles pueden adaptarse más a unas dimensiones que a otras), si el control va a mandar señales continuadas o momentáneas, como mapeamos las acciones de nuestro juego a las partes de nuestro input y el sistema de input en general (Estructura en mano materiales etc.).

El Input es algo que siempre debemos tener en cuenta puesto que es algo que va a limitar las acciones máximas de nuestro juego. Como dice Tadeusz Stach, T.C. Nicholas Graham, Matthew Brehmer, Andreas Hollatz en *Classifying Input for Active Games* "when designing an active game, developers must first consider what input hardware the game will use. Therefore, active game designers are limited to the capabilities of a specific hardware device." [7]. Ellos se refieren a inputs en juegos que requieren movimientos físicos del jugador, pero el mismo principio se puede aplicar a juegos normales. Por ejemplo, no es lo mismo sacar un juego para Nintendo DS con su pantalla táctil y su directional pad que para la PlayStation 3, por lo que antes de empezar a realizar nuestro videojuego tendremos que pensar en que plataformas podemos sacarlo y si luego pensamos en adaptarlo a otras plataformas tendremos que pensar en cómo podemos adaptar los esquemas de controles de un sistema a otro.

Por eso muchas veces se hacían varias versiones de un mismo juego para abarcar más mercado, podemos encontrar casos así en juegos de licencias de películas como Spider-Man 2 la cual tuvo adaptaciones en casi todas las plataformas disponibles en su época con 5 versiones diferentes del juego. Estas versiones llegaban a distar mucho entre sí ya que debían adaptarse a hardware muy diverso con lo que conllevaba, un sistema de inputs distinto en cada consola.

Además, no todas estas versiones estaban realizadas por la misma empresa, sino que se encargaban algunas versiones que pudieran tener menor repercusión en el mercado a estudios menores.

Estas adaptaciones muchas veces eran mediocres ya que, tal y como dijo Miyamoto, la manera en la que se percibe una película y un videojuego son muy diferentes, el cine es un medio de entretenimiento pasivo en el que el consumidor simplemente observa, mientras que en los videojuegos son interactúan constantemente con el producto. [8].

En cuestión de medir la respuesta, de nuevo, encontramos métricas duras y blandas, Swink distingue las siguientes:

"-Hard

1. How many objects the player controls.

2. The dimensions, type and frame of reference for the movement of each avatar.

3. The ADSR envelope representing each modulation of a game parameter by an input over time

-Soft

1. The overall sensitivity of the system as a function of its input and response sensitivity" [1].

Alguna de las posibles respuestas que podemos obtener de nuestros inputs son la modificación de los parámetros, también pueden crear instancias como cuando creamos bombas con Bomberman, o simplemente iniciar una animación o cambiar de estados a nuestro avatar. Por ejemplo, en The Legend Of Zelda Ocarina of Time cuando saltamos al agua todo nuestro esquema de movimiento cambia y la respuesta que los inputs anteriores nos daban unos resultados ahora nos dan otros (en vez de andar, nadar o no poder atacar con la espada en el agua).

La modificación del parámetro seguirá una especie de curva, Swink lo compara a la del envolvente acústico ya que podemos diferenciar un Atack, Decay, Sustain y Release (aunque en algunas ocasiones no contamos con Decay). [1]

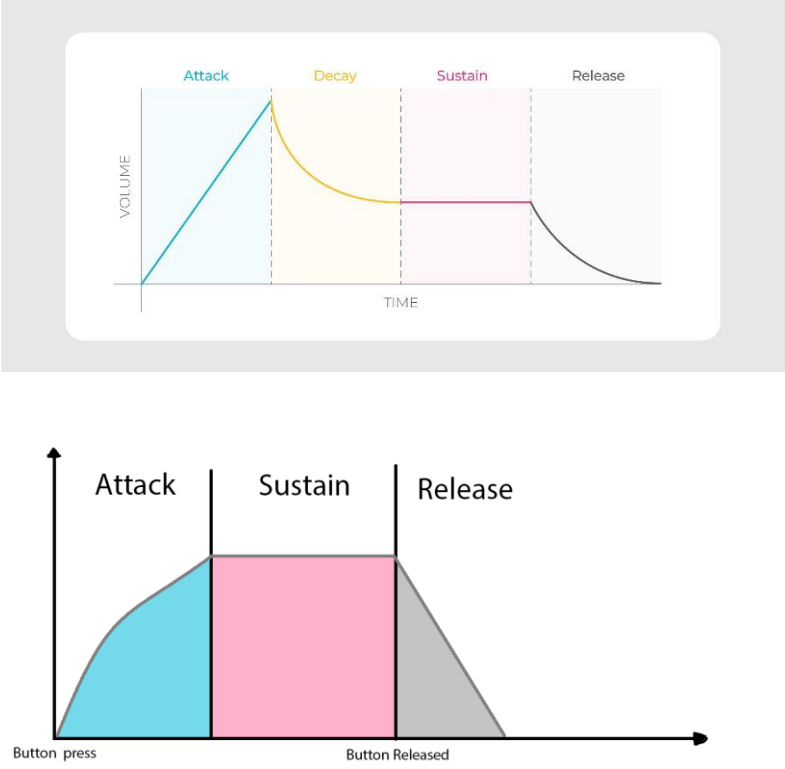


Figura 6: ADSR en sonido vs ASR Respuesta a un input

Con esto podemos calcular cuánto tiempo debe durar cada parte de la respuesta para generar la sensación que estemos buscando.

El siguiente aspecto que podemos medir del Game Feel es el contexto el cual podemos medir en tres niveles:

1- Nivel alto (Impresión de espacio, movimiento y escala, inherente a la creación del mundo del juego)

2- Nivel medio (Interacción del avatar con el mundo alrededor suyo)

3- Nivel bajo (La interacción más táctil entre objetos) [1]

Como podemos ver por la descripción de cada uno de estos niveles, en lo que a contexto se refiere no tenemos unas métricas duras como en anteriores apartados debido a lo subjetivo del tema que estamos tratando. Pero eso no quiere decir que no se puede medir ya que las experiencias de los jugadores como métricas blandas nos ayudarán de igual manera.

Para entender mejor como interpretar estos niveles simplemente tenemos que pensar que vamos de más general a más concreto. En el nivel alto encontramos el mundo como tal, la escala del mismo y de los objetos que contiene, si se mueven o no…

En el nivel medio empezamos a incorporar al jugador y como su posición respecto a objetos del mundo puede afectarle, por ejemplo, si el *layout* [5] del mapa es un bosque muy frondoso generará más agobio que una pradera vacía.

Y por último en el nivel bajo encontramos las matemáticas en las colisiones y la relación entre colisiones de objetos. Swink propone de ejemplo el World of Warcraft dónde el mundo es gigante pero las interacciones entre el avatar, piedras del entorno, hojas, árboles etc. Son nulas y el mundo acaba sintiéndose estéril en contraste con lo enorme que es [1]. En cambio, en Red Dead Redemtion 2, cada paso que da nuestro avatar interactúa con el entorno de alguna manera, modificando la hierba del camino, creando huellas en el suelo si hay barro o nieve, y recibiendo toda clase de palabras de los *NPCs* [6] que pasen al lado nuestro generando una sensación de que todo con lo que interactuamos reacciona de una manera u otra.

El pulido un aspecto del Game Feel que aporta mucho al mismo ya que engloba varios apartados del diseño y desarrollo de videojuegos. El pulido hace que el jugador crea en la fisicidad de las cosas dentro del mundo del juego y se mide con términos como, pesado, liviano, mojado, afilado…

De nuevo, son métricas blandas las cuales son subjetivas, pero dependiendo de cómo describan como se sienten los jugadores con cierto apartado de nuestro videojuego sabremos como debemos tunearlo para arreglarlo.

No afecta a las físicas de la simulación que hayamos creado, sino que es un añadido más que crea en el jugador una manera de entender el mundo del juego con sus mecánicas y sus físicas de una manera más amigable.

Los diferentes tipos de pulido que podemos encontrar y lo que generan son: animaciones, efectos de sonido, efectos visuales, efectos cinemáticos, y efectos táctiles. No es que sea estrictamente necesario que nuestros juegos tengan que cubrir cada uno de estos apartados, pero como ya hemos mencionado, ayudan al jugador a entender nuestro mundo y en ocasiones es muy necesario prestar especial atención a estos apartados.

Tras el pulido encontramos la metáfora la cual se centra en la representación y tratamiento de objetos, interacciones etc. Este aspecto del Game Feel se centra en no sacar al jugador de la experiencia de juego por culpa de disonancias. Me explico, si un objeto aparentemente pesado (una caja de metal) está cayendo de una altura elevada cerca del jugador, este esperará que haga un sonido estruendoso cuando toque el suelo, pero si esta no hace sonido alguno el jugador quedara extrañado ante esta situación. Como ya comenté, si sucede esto en varias ocasiones seguidas es muy probable que el jugador deje de jugar a nuestro juego porque su experiencia se ve interrumpida por la falta de consistencia en nuestro mundo virtual.

No es que todo tenga que reaccionar de manera super realista, pero sí que todo tiene que mantener una consistencia dentro del mundo que estamos creando, por ejemplo, si el mundo de nuestro videojuego es todo de lana como por ejemplo el de Yoshi's Woolly nos choca menos que Yoshi en vez de poner huevos ponga ovillos de lana. [9]

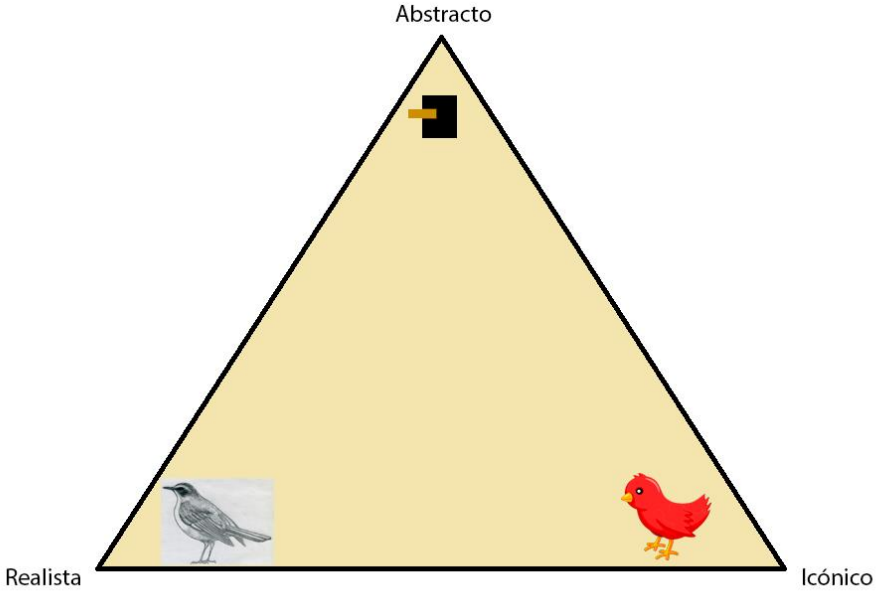


Figura 7: Diferentes tipos de representación, basado en la figura del libro Game Feel: A Game Designer's Guide to Virtual Sensation, de Steve Swink (2008)

Por último, solo nos quedan las reglas, las cuales como el contexto se dividen en bajo medio y alto nivel [1]:

1- Reglas de alto nivel: Son un set de reglas general que incentivan a que el jugador juegue de una manera sin alterar objetivos principales del juego. Podemos encontrar como ejemplo coger 100 monedas en cualquier juego de Mario para obtener una vida, no tener el mini mapa activo en The Binding of Isaac o incluso no tener HUD en Battlefield 1 [10], estas tres hacen que el jugador pueda cambiar su estilo de juego dependiendo de las circunstancias en las que se encuentre.

2- Reglas de medio nivel: Reglas relacionadas con objetos en específico los cuales pueden dar un significado a mecánicas o acciones del juego. Medallones de templo en The Legend Of Zelda: Ocarina Of Time que sirven para avanzar en la trama o la bandera de cualquier juego con un modo de capturar la bandera. Refuerzan el conocer el entorno y el movimiento relativo a ciertos objetivos.

3- Reglas de bajo nivel: Definen las propiedades físicas de los objetos, por ejemplo, cuantos golpes le lleva al jugador destruir cierto objeto o enemigo. Donde se aplican los famosos "Balanceos" en juegos multijugador como por ejemplo Call Of Duty dónde cambiar el valor del daño de un arma puede hacer que todo el mundo la use para aprovecharse de su poder o para que armas que no son lo versátiles que deberían vuelvan a ser populares entre los jugadores [11].

Con esto hemos acabado de definir como se mide cada aspecto que conforma el Game Feel, esto nos será útil a la hora de decidir cómo vamos a actuar al crear nuestra demostración para lograr nuestro objetivo, que no es otro que generar una experiencia en el jugador.

3. Objetivos

En el tercer apartado vamos a revisar los objetivos que este proyecto pretende alcanzar. Comenzaremos explicando el desarrollo de la creación de la demo del juego de terror y conforme vayamos avanzando en la creación de esta, iremos justificando cada decisión con ejemplos de la industria y con otros proyectos antiguos para demostrar que la opción elegida es la correcta o la que mejor resultados puede dar con los recursos actuales.

Dicho esto, los objetivos de este trabajo de fin de grado serán los siguientes:

1. Creación de una demostración jugable de un juego de terror basándonos enteramente en dar una Sensación de Juego especifica al jugador.

2. Analizar juegos en el mercado del mismo género y contrastar con datos las decisiones tomadas.

3. Comparar con proyectos personales anteriores y por qué la nueva es mejor en cuestión de Sensación de juego

4. Analizar con estadísticas los resultados obtenidos y obtener una conclusión basada en experiencia de usuario.

4. Metodologías

En este cuarto apartado vamos a revisar como han sido realizadas las fases de este proyecto, las herramientas que han sido utilizadas y como ha sido estructurada la demostración jugable.

4.1. Fases del proyecto

Las fases en las que el proyecto ha sido dividido principalmente ha sido en dos. Por un lado, tenemos la implementación de la demostración jugable y por otro investigación y recolección de datos para el análisis y la comparación con otros juegos o proyectos.

El proyecto comenzó desde principios del segundo cuatrimestre, y duró hasta la fecha de entrega de este. Las reuniones con el Tutor del proyecto dependían de la disponibilidad de cada uno, pero siempre se hacían con un propósito, ya sea definir objetivos, resolver dudas o revisar que el proyecto no se aleje de lo que debe mostrar ya que al ser un término tan nuevo y subjetivo es fácil desviarse del tema principal haciendo hincapié en cosas que tal vez no requieran tanto interés. Si hubiese dudas puntuales, estas se resolvían mediante correo electrónico.

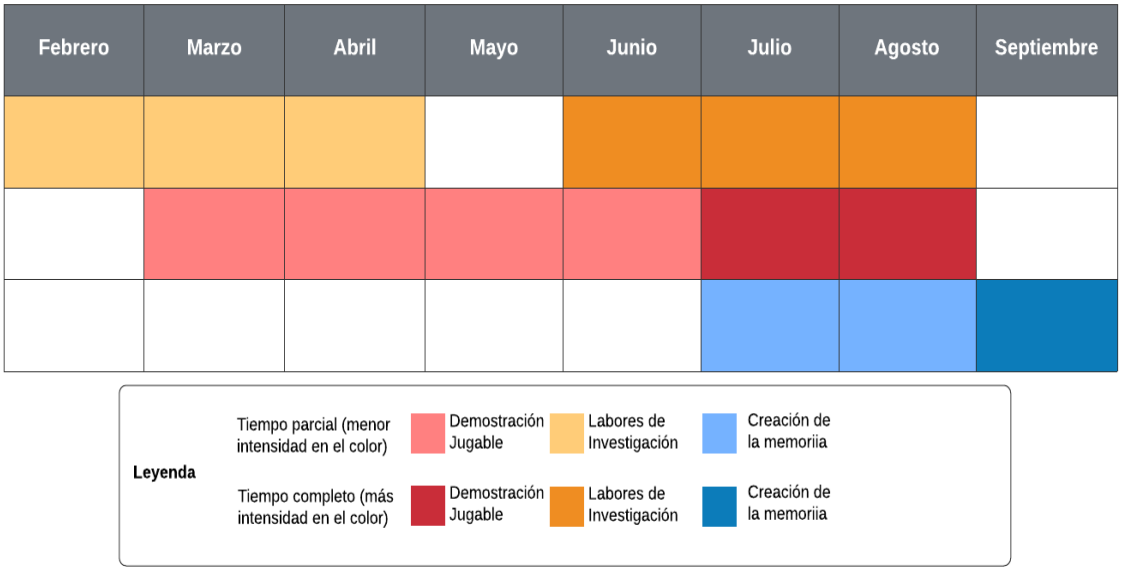


Figura 8: Cronograma de este TFG

Cabe destacar que, aunque el proyecto se haya empezado a realizar en febrero de 2021, hasta el final del curso no se le pudo dedicar el tiempo necesario para ver avances notorios tanto en la demostración como en la recogida de datos. El proyecto comenzó en febrero, pero hasta mediados de Julio no se empezó a desarrollar la memoria del mismo la cual concluyó en septiembre. Todos estos detalles se pueden ver reflejados en el cronograma de la figura 8.

4.2. Herramientas utilizadas

Las herramientas que han sido utilizadas durante el desarrollo de este trabajo ha sido principalmente el motor llamado Unity usado principalmente para la creación de videojuegos [12].

Para la creación de animaciones se ha usado el animador de Unity el cual permite animar de manera sencilla cualquier modelo 3D [13].

De la asset store (tienda virtual del motor de Unity) se han adquirido gratuitamente algunos paquetes que mejorarán la experiencia de testeo y la de la demo en general. Estos paquetes son: Texturas, el modelo de la pistola, sonidos…

De una página de assets online gratuitos y sin copyright conseguí el modelo de los brazos con el *rigging* [7] listo para comenzar a animar.

Para el Post Procesado usado en la demostración se han usado tanto el compatible con Universal Render Pipeline como el compatible con Built-In Render Pipeline, pero al final como se optó por usar la Built-In render pipeline se usó el último mencionado [14].

Unity por defecto usa el editor de código Visual Studio y es el que usé para la creación de la demo en su versión de 2019.

Para documentar cualquier avance en el proyecto use mi canal de YouTube en el cual iba subiendo videos conforme la demostración iba avanzando, creé una lista de reproducción para que todos los videos sean ordenados cronológicamente y el proceso de creación sea más fácil de segur [15].

He tenido experiencia con el control de versiones de Unity, pero debido al limitado espacio de almacenamiento que ofrece decidí crear copias de seguridad cada cierto tiempo que se iban subiendo a Google Drive y en un disco duro secundario.

Por último, las reuniones con el tutor se realizaron a través de la plataforma de Microsoft Teams.

4.3. Estructura de la demostración jugable

La estructura de la demostración jugable ha sido creada de manera que cubramos varios dispositivos para demostrar cómo el dispositivo usado para jugar puede afectar como vemos el juego. Todas estas versiones fueron creadas en el motor Unity y codificadas en C# siguiendo las siguientes reglas:

- Las funciones, clases, constantes, estructuras y enumeradores deben empezar siempre con letra mayúscula.

- Todo el código debe seguir una indexación correcta y estructurada en niveles.

- Los nombres de las variables deben seguir descriptivos, explicando mayormente su aporte en el código. Su tipo de valor especificado. El uso de los guiones bajos debe ser evitado.

- Las llaves y paréntesis deben empezar o terminar en nueva línea. Si el contenido de estos puede resumirse en una sola línea, se omitirán las llaves y paréntesis, aunque el contenido debe estar en una sola línea e indexado correctamente.

5. Desarrollo

En este apartado explicaré el desarrollo del proyecto y el porqué de cada una de las decisiones tomadas en el mismo.

Lo primero de lo que es necesario hablar es de los antecedentes que me llevaron a elegir crear una demostración basada en un juego de terror y crear una sensación de juego basada en ello.

5.1. Antecedentes

En el segundo año de carrera se creó SILENCE un juego creado para la asignatura de diseño 3D, yo fui el encargado de la jugabilidad y no quedé muy contento con el resultado ya que al ser un programador más novato y SILENCE tratarse de nuestro primer juego en tres dimensiones acabó saliendo con ciertas carencias en cuestión de Game Feel que ahora veo y no deberían ser así [16].

Al ser un ejemplo propio creado en el pasado de la mejor manera que pude, vi interesante la idea de comparar y contrastar con mi yo pasado un proyecto de índole similar, pero aplicando todo el conocimiento adquirido durante mis labores de investigación en la materia de sensación de juego.

Dos años más tarde para la asignatura de videojuegos y simulación para investigación y educación creamos una experiencia en primera persona para ayudar a la gente con claustrofobia a superar su miedo, ese mismo cuatrimestre me empecé a informar y descubrí este término y empecé a aplicarlo en mis juegos para que la experiencia creada fuese lo mejor posible. [17]

De nuevo fui el encargado de crear la jugabilidad del juego, pero esta vez el resultado fue muy diferente a pesar de haber gastado el mismo tiempo en ambos proyectos. Al tratarse de un juego para tratar una fobia no debíamos influir en el juicio del jugador demasiado, por lo que la espinita de crear una demostración de un juego de terror con toda la información que conocía y que me faltaba por conocer me llamaba mucho la atención.

Por otra parte, el género de videojuegos de terror es uno del cual tengo bastante experiencia y conocimiento ya que he jugado a gran cantidad de títulos de este por lo que referentes no me faltan para crear una demo.

5.2. Desarrollo de la demostración jugable en PC y dispositivos móviles.

La demostración jugable tenía unos objetivos que cumplir desde un comienzo. Realizar y cumplir todas las características mínimas para poder decir que nuestro juego tiene Sensación de Juego y una vez este consolidado le iré dando un enfoque que acompañe en armonía a la jugabilidad de la demostración.

Para comenzar deberemos definir los objetivos que debe cumplir nuestra demostración y la experiencia que debe generar en el jugador. El juego será en primera persona, ya que, para el género de terror, es una perspectiva bastante recurrente ya que te pone en medio del peligro constantemente. El control será todo lo responsivo que se pueda dejando de lado delay en inputs, con esto el jugador tendrá la sensación de que cada acción realizada en el juego tiene una respuesta inmediata debido a su interacción con el juego y, por ende, que son causadas por él.

Las mecánicas básicas de movimiento deberán ser lo suficientemente interesantes como para que el jugador no se aburra ya que serán las que más tiempo esté ejecutando, por ello, necesitaremos de un pulido extra en ese aspecto, por ejemplo, con animaciones elaboradas que creen en el jugador la sensación de que está manejando a una persona y no a una cámara que se mueve por un entorno virtual.

La demostración contará con un sistema de disparos, este deberá generar en el jugador la sensación de que de verdad está tratando con un arma, de nuevo, animaciones detalladas serán requeridas junto con unas pistas de audio para crear dicha sensación.

Dicho esto, lo primero que debemos hacer es crear un proyecto de Unity seleccionando el render pipeline [18] que vamos a usar teniendo para elegir 3 cada una de ellas con sus pros y sus contras. Como se puede ver en la web de Unity, cada una de estas render pipelines está enfocada para diferentes tipos de proyectos, nuestro proyecto pretende ser una demostración jugable de cómo crear un movimiento y algunas mecánicas que se sientan como del género de terror.

Todas las herramientas extras que ofrece la High Defnition Reder Pipeline no nos serán muy útiles puesto que nuestras plataformas objetivo son principalmente ordenadores y móvil (la High Definition Render Pipeline apunta a ordenadores potentes y consolas). Por lo que solo nos quedan dos opciones para elegir, la Built In render pipeline y la Universal Render Pipeline.

El proyecto comenzó con la idea de usar la Universal Render Pipeline puesto que es más personalizable y rápida que la Built-In, pero, acabo cambiándose tras un par de meses debido a varios problemas de compatibilidad entre los diferentes sistemas de post procesado y la Render Pipeline, por lo que al final acabé usando la Built -In que a pesar de ser menos flexible es con la que he estado trabajando todos estos años y sé manejar mejor.

Una vez elegida el render pipeline, la demostración inició con la creación de un cubo en Unity, lo modifiqué para que pareciese que tuviésemos un suelo. Solo con este simple proceso ya tenemos un mundo por el cual movernos, bastante simple, pero un mundo, al fin y al cabo. Como estamos creando un movimiento en primera persona deberemos llenar este mundo de ciertos obstáculos para testear nuestro avatar.

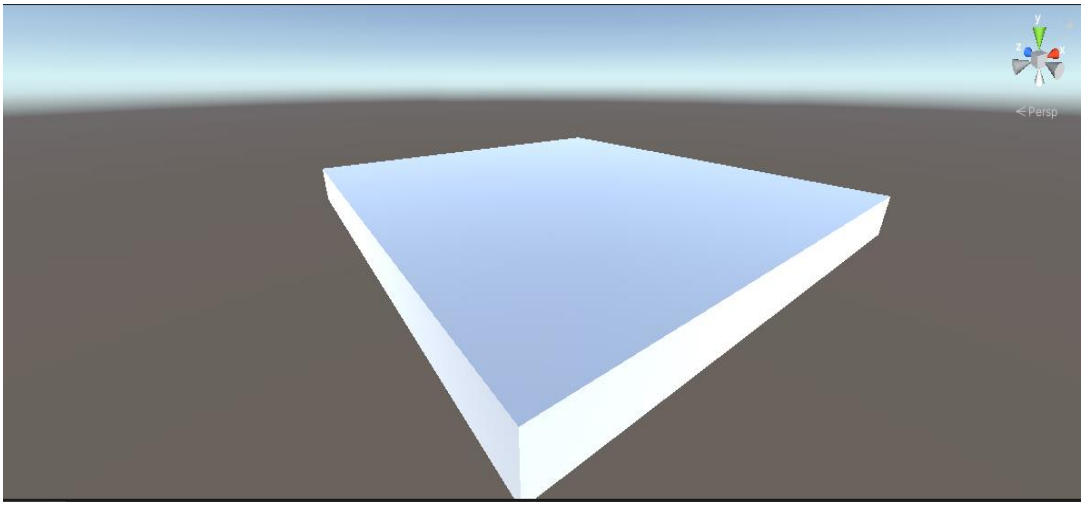


Figura 9: El comienzo de nuestro mundo del juego

La segunda incógnita con la que nos encontramos en el proyecto fue, ¿Cómo creo el movimiento en primera persona para este proyecto? Normalmente, en Unity, tenemos dos caminos que tomar a la hora de crear un movimiento en primera persona, usar el componente llamado 'Character Controller' [19] o crear un movimiento más personalizado a partir del uso de componente de 'Rigid Body' [20].

Existen más ventajas y desventajas para la creación de juegos en primera persona usando cualquiera de esos dos métodos, pero tras informarme y haber revisado los anteriores ejemplos creados para la universidad al final la decisión estaba tomada [21].

En SILENCE se optó por usar el componente de character controller ya que era el que más facilidades nos daba en aquel momento, pero conforme íbamos avanzando en el proyecto, nos dimos cuenta de que tal vez sería mejor cambiar la manera en la que crear el movimiento en primera persona, y eso hicimos, en tercero creamos un plataformas [22] en primera persona usando el componente Rigid Body como principal herramienta para crear el movimiento en primera persona. Realmente dio muchos problemas, pero finalmente se consiguió un mejor resultado que en SILENCE, por lo que a partir de este momento optamos por realizar los proyectos con movimientos en primera persona usando el Rigid Body en lugar del Character Controller.

Teniendo esto en cuenta, podemos comenzar con el desarrollo de nuestro movimiento, pero antes necesitaremos tener un par de cosas más en nuestro mundo para poder testear correctamente el movimiento cuando empiece a crearlo.

Necesitaremos una cuesta y una pared, ¿Por qué? Para crear como nuestro movimiento debe reaccionar ante colisiones con paredes y suelos con elevaciones. También pondremos un par de texturas del pack de la Asset Store [23] que nos ayudara para diferenciar mejor los entornos.

Una vez realizados los anteriores pasos se procede a realizar la implementación del movimiento en primera persona del jugador. El procedimiento empieza de manera muy similar al del juego realizado para tratar la claustrofobia, se necesitará conseguir la máxima inmersión del jugador posible en el juego. En la demostración, se han evitado cambios bruscos de valores en el movimiento de jugador, es decir, cuando el jugador pasa de andar a correr la velocidad de este no cambiará directamente, sino que una interpolación lineal.

Eso será aplicado a toda clase de transiciones entre estados que cambien la velocidad máxima o mínima que puede alcanzar el jugador.

Lo que queremos potenciar en esta demostración es la inmersión, en nuestro caso comenzaremos con la cámara puesto que es uno de los elementos que más trabajo ha requerido.

La cámara va a ser los ojos de nuestro jugador y sin duda va a ser uno de los puntos que más atención hay que prestar. Para juegos de terror, lo primero que hay que mentalizarse es que la cámara deberá tener un tratamiento más especial que en otros géneros como en Shooters generales ya que en este género más que en otros debemos darle al jugador la sensación de que él es el que está andando, corriendo, disparando etc.

Existen varias opciones para hacer que el jugador se sienta en el interior del juego y no como una cámara en un trípode. En Mirror's Edge (2008) [24], un juego de plataformas en primera persona basado en el parkour y el movimiento, hasta el simple hecho de estar quieto influye en la cámara del jugador, la protagonista respira y con esta respiración la cámara se mueve hacia arriba cuando inhala y hacia abajo cuando exhala.

Estamos hablando de ese nivel de detalles, que pueden parecer nimios e insignificantes al principio, pero sumados con otros crean una experiencia muchísimo más interesante. Mirror's Edge es un título del cual coger mucha inspiración ya que fue de los primeros juegos dónde realmente parecía que eras la protagonista y todo esto en gran parte fue por el genial manejo de la cámara, cada vez que Faith (su protagonista) realiza una acción la cámara se mueve en consonancia con esta, si por ejemplo Faith decide correr, la cámara simulara el movimiento de su cabeza moviéndose de una manera diferente a como lo hace cuando anda.

Estos efectos están presentes también en juegos de terror como Resident Evil VII [25], la mayor diferencia entre la cámara del anterior ejemplo y la de Resident Evil es que al ser de géneros diferentes la manera en la que muestran las acciones del jugador varían, los movimientos de la cámara de Resident Evil son mucho más lentos, también encontramos diferencia entre el movimiento que realiza la cámara al correr y al andar como en Mirror's Edge, pero al ser un juego menos centrado en los saltos y en el plataformeo, la cámara no necesita exagerar más de lo necesario las acciones del personaje principal ya que no va a ir tan rápido como Faith.

Estos efectos pueden lograrse de dos formas principalmente, podríamos conseguir estos efectos de respiración o movimiento mientras nos movemos mediante el uso de Animator. Pero realmente estos movimientos al ser realmente simples no merecen la pena ponerse a hacer animaciones para ello, en su lugar se "animara" cambiando la posición de la cámara mediante una función a partir de código.

La función usada puede verse en la siguiente figura:

```
5 referencias
public void HeadBobs(GameObject objectToBob, float pZ, float pXIntensity, float pYIntensity)
{
    objectToBob.transform.localPosition = new Vector3(Mathf.Cos(pZ) * pXIntensity, Mathf.Sin(pZ * 2) * pYIntensity , 0);
}
```

Figura 10: Función de movimiento de la cámara

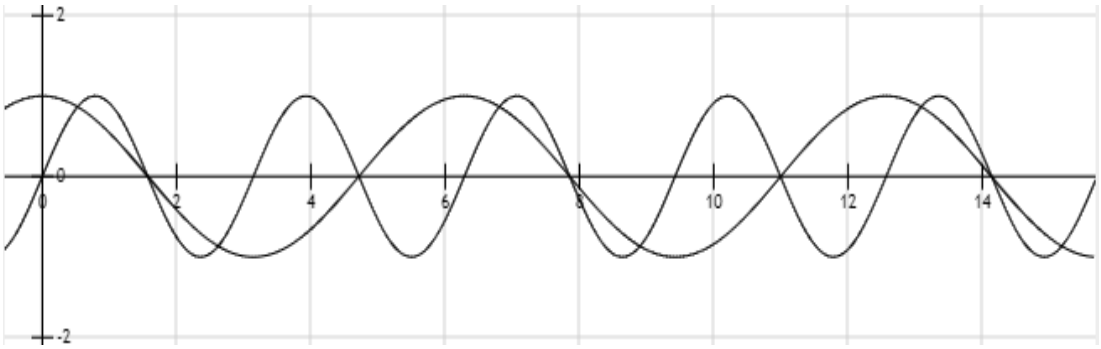


Figura 11: Representación del movimiento de la cámara.

Como podemos ver simplemente necesitamos alterar la posición local de la cámara (ya que está en el interior de una jerarquía de objetos bastante grande) y mover sus componentes x e y siguiendo un coseno y un seno respectivamente a la intensidad deseada (pXIntensity, pYIntensity). La posición en el eje Z de la cámara se decidió dejar quieta ya que realmente no se aprecia mucho y si a la gente con motion sickness [26] le suele afectar estos movimientos de cámara, mejor limitarse a lo estrictamente necesario, pero por supuesto con buenos resultados.

El Field Of View o Campo de Visión, es otro de los parámetros que nos será de gran ayuda para crear ciertas sensaciones en el jugador, más concretamente las relacionadas con la impresión de la velocidad.

El Field Of View es "the angular cone-of-vision a camera uses to display content on screen, and is subject to various characteristics and effects." [27], con este efecto lograremos no solo la sensación de velocidad ya mencionada al aumentar el ángulo de visión del jugador, también podremos hacer un efecto de zoom cuando apuntemos a través de la mirilla de nuestra pistola.

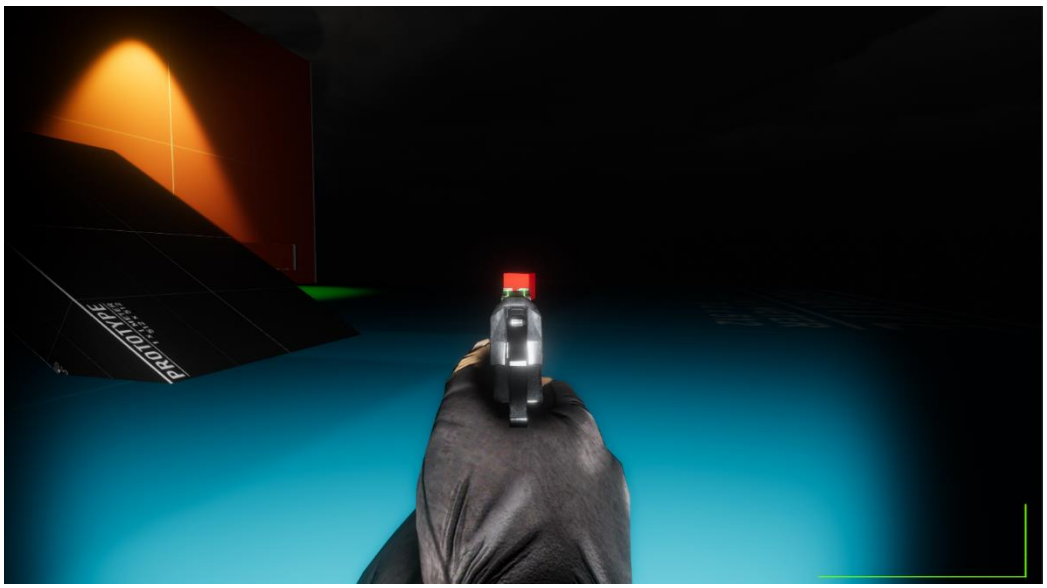


Figura 12: Apuntado con su FoV personalizado (Versión final del proyecto)

Figura 13: Apuntado con un FoV igual al usado para andar (Versión final del proyecto)

Como se puede ver, en la figura 12 el campo de visión es mucho más amplio con el Field Of View sin darle un valor especial al apuntado. El presente objetivo es simular que el foco de atención de nuestro avatar esta justo en frente y no en su visión periférica.

En la vida real para apuntar con una mirilla necesitaríamos cerrar un ojo, esto, al ser evidentemente imposible de realizar en un videojuego, intenta simularse con esta técnica para aparte de centrar al jugador en lo que tiene en frente, además, no puedes correr mientras apuntas por lo que estar con un *Field Of View* [8] y dar sensación de movilidad y un amplio campo de visión al jugador sería contrario a la experiencia que queremos lograr en una demostración de un juego de terror.

El Field Of View también ha sido usado en el cine creando efectos tan curiosos como el Dolly Zoom o Travelling Compensado, el cual consiste en acercar la cámara al actor u objeto deseado mientras se hace un Zoom Out generando así una sensación de vértigo, de hecho, la primera vez que se usó fue en la película Vértigo de Alfred Hitchcock. Este efecto sería muy útil si se usase en cinemáticas, pero, aunque estemos centrados en el gameplay el Dolly Zoom debía tener una mención especial puesto que su uso, no solo en videojuegos, sino en el género de terror está bastante expandido.

Otra característica que debe tener la cámara es reaccionar a la mecánica de disparos cuando el jugador la realice, para recrear como nuestro cuerpo reaccionaria al retroceso de un arma, en el caso de la demo, de una pistola.

El camera shake es uno de los elementos más famosos a la hora de añadir 'Juice' a un videojuego, en el caso de esta demo se añadirá no solo al disparo del arma, sino al recibir daño de un "Enemigo" y al recuperarse de una caída de una posición elevada.

Este efecto será efectuado en un contenedor de la cámara diferente para no crear problemas con los efectos anteriores que generan cambios en la posición de la cámara. El funcionamiento será el siguiente, efectuaremos un cambio en la rotación y en la posición en X e Y del contenedor de la cámara, este cambio una vez sea aplicado, en el Update estaremos continuamente actualizando la posición del contenedor para que vuelva a su posición original, todo esto a una velocidad y rangos de movimiento y rotación personalizables. En código luciría de esta manera:

```
void FixedUpdate()
{
    currentRotation = Vector3.Lerp(currentRotation,Vector3.zero,returnSpeed * Time.deltaTime);
    rotation = Vector3.Slerp(rotation, currentRotation, rotationSpeed * Time.deltaTime);
    transform.localRotation = Quaternion.Euler(rotation);
}

2 referencias
public void CreateCameraRecoilCustomized(Vector3 recoilRotation)
{
    if (movement.playeraCameraScript.aiming)
    {
        currentRotation += new Vector3(-recoilRotation.x, Random.Range(-recoilRotation.y, recoilRotation.y), Random.Range(-recoilRotation.z, recoilRotation.z));
    }
    else
    {
        currentRotation += new Vector3(-recoilRotation.x, Random.Range(-recoilRotation.y, recoilRotation.y), Random.Range(-recoilRotation.z, recoilRotation.z));
    }
}
```

Figura 14: Update y función del Recoil del arma en la cámara.

Como he mencionado antes, al haber hecho la función de manera que reciba los parámetros de intensidad que queramos, podemos crear diferentes tipos de retroceso de armas, esto sería muy útil si contásemos con diferentes tipos de armas, pero actualmente está implementado para simular el retroceso de la pistola, para cuando el jugador recibe daño o cuando cae de sitios elevados todo ello con valores diferentes.

El siguiente movimiento que hará nuestra cámara será el conocido como View Roll. Este movimiento crea una leve inclinación en la cámara cuando giremos hacia los lados, es un efecto muy peligroso y no muy usado ya que puede generar motion sicknes si es muy exagerado por lo que los valores usados serán bajos.

Lo que pretende simular este efecto es las leves inclinaciones de cabeza que se realizan cuando nos movemos hacia izquierda o derecha. Es un efecto realmente simple de añadir puesto que simplemente necesitamos rotar en el eje Z positiva o negativamente dependiendo de si giramos a izquierda o derecha. En la saga Half Life hubo varias personas quejándose debido a la implementación de este movimiento de cámara puesto que aparte de realizarse de manera muy intensa, la rotación de la cámara era demasiado elevada. Por suerte, se podía modificar la intensidad del efecto e incluso desactivarse completamente, al menos en la versión más reciente.

En la siguiente figura podemos observar cómo se vería en nuestra demo el apuntado de alguien moviéndose hacia la izquierda con este efecto activado y con este efecto sin activar:

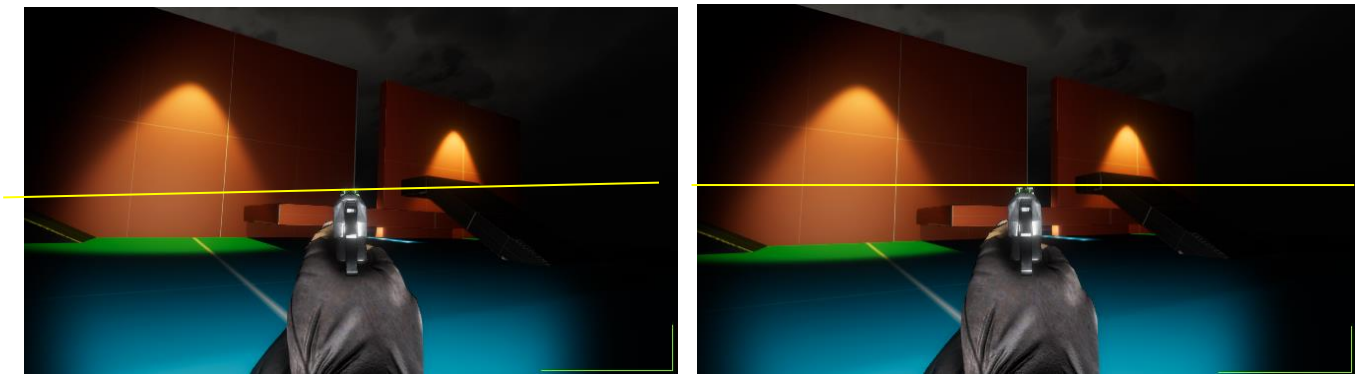


Figura 15: View Rolling vs No View Rolling (Versión final del proyecto)

Las figuras dobles que pretenden comparar datos se pueden ver completamente extendidas en el Anexo IV.

Tras haber visto los siguientes efectos de cámara, vamos a pasar a la siguiente problemática de nuestro proyecto y es que, actualmente como está el proyecto, tenemos un problema que rompe directamente toda la inmersión lograda ya que, evidentemente, en el mundo real cuando tocamos una pared no la atravesamos e incluso en el mundo del juego no tiene sentido. Nuestra demo pretende hacer hincapié sobre todo en la inmersión del jugador en el mundo (que, aunque de pruebas) debe ser consistente con las reglas que el jugador conoce, no existe ningún motivo por el cual esto deba suceder por lo que se debe arreglar.

Este problema se le conoce como *clipping* [9], en concreto, el arma atraviesa con cualquier pared o suelo que el jugador tenga en frente. Esto es debido a que el modelo 3D de nuestros brazos y arma quedan fuera de la capsula de colisión de nuestro personaje por lo que no colisionaran con paredes, sino que las atravesará. Este problema luce como en la siguiente figura y tiene distintas soluciones dependiendo de la render pipeline que usemos.

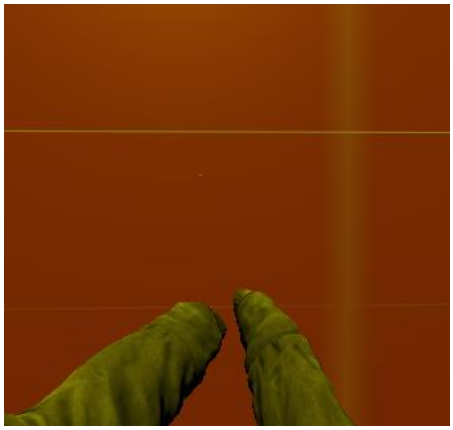


Figura 16: Modelo de nuestros brazos atravesando la pared

La solución en ambas render pipelines es usar una segunda cámara que renderice solamente los brazos y dejar a la anterior el renderizado del mundo. En la universal render pipeline contamos con un componente en las cámaras no disponible en la built-in render pipeline conocido como Camera Stacking el cual combina capas renderizadas en diferentes cámaras en una sola, y sería tan fácil como añadir las cámaras a la lista del componente como se puede ver en la web de Unity [28].

En cambio, cuando di el paso a la Built-In render pipeline este componente y facilidades no existían por lo que la manera de crear la segunda cámara es seleccionando las layers que cada una renderizará. Una vez hecho eso debemos poner esta cámara en la jerarquía del objeto player para que siga todos los movimientos que realiza la cámara que renderiza el mundo.

El resultado es inmediato y no solo contamos con el problema resuelto, sino que, al contar con una cámara nueva para renderizar exclusivamente los brazos, los valores de Field Of View de la cámara que renderiza el mundo y sus valores no afectaran a esta haciendo así que no se deformen los brazos por el efecto ojo de pez que se obtiene al tener un Field Of View demasiado elevado.

Figura 17: Modelo de nuestros brazos renderizados en otra cámara

El Field Of View se mantendrá siempre en la misma posición debido a que en este proyecto solo pude contar con el modelo de unos brazos y al aumentarlo puede verse dónde acaba el modelo, en cambio, si tuviese acceso a un modelado de un cuerpo entero, posiblemente se podría modificar para añadir incluso más sensación de velocidad, pero como ya hemos dicho no quiero generar una sensación de velocidad extrema en el jugador.

El post procesado será el último efecto que añadiremos a la cámara y fue el motivo por el cual cambié de render pipeline. Lo que se busca con los efectos de post procesado es generar un ambiente sombrío (color grading) con ciertos efectos que añadan un pulido extra a la imagen para que la demo no se vea tan mal y que, por ejemplo, los bordes de sierra no sean tan exagerados que saquen al jugador de la experiencia.

Cabe destacar que el post procesado variará entre las versiones de PC y móvil puesto que la potencia que se requiere para el post procesado en móvil no es la misma que en un portátil debido a los componentes que tengan.

Antes de empezar describiendo el porqué de los efectos usados, explicaré por qué se cambió de render pipeline por el post procesado. El post procesado en la Universal Render Pipeline funciona de manera similar a la Built In render pipeline pero hay un efecto que añade mucho a la imagen y a la opresión del ambiente y esa es la Ambient Occlusion.

Al haber trabajado ya con la Built In supe que debía volver a esta render pipeline lo antes posible solucionando todos los problemas que esto conllevó. Una vez resueltos al final pude contar con AO para la versión de PC y los restantes efectos para la versión de móvil.

Los efectos visuales usados fueron, vignette (viñeta), la ambient occlusion (oclusión ambiental), color grading (escalado de colores), bloom (resplandor) y chromatic aberration (aberración cromatica). Cada uno sirve un propósito diferente en la demostración:

Viñeta: Efecto que oscurece o ilumina los bordes de la pantalla, puede reflejar varias cosas en un juego, como que el jugador ha sufrido daño pintando de color rojo los bordes de la pantalla. En nuestro caso indicara al jugador que se encuentra agachado pintando los bordes de negro sutilmente.

Ambient Occlusion: La oclusión ambiental marcó una diferencia abismal en el juego que creamos para tratar la claustrofobia puesto que oscurecer las zonas donde se junta la geometría genera una sensación de oclusión que realmente realzó los espacios cerrados de aquel juego, en este proyecto se usará de la misma manera.

Color Grading: Básicamente altera el color de la imagen resultante, funciona como los filtros que puedes aplicar en las fotos de Instagram o cualquier otra red social. Sirve para dar personalidad a nuestro juego y un aura más personalizable pudiendo crear escenarios más cálidos o fríos, oscuros o iluminados etc., Dependiendo de la configuración de parámetros de color que usemos.

Bloom: Este efecto es algo difícil de definir, pero básicamente intenta simular los artefactos generados por objetos brillantes ante una cámara. Dependiendo de la intensidad y el radio de acción el aura formada alrededor de las luces pude ser exagerada hasta el punto de no ver nada. Evidentemente nosotros usaremos este ejemplo sin pasarnos ya que la mayoría de los espacios son oscuros y aunque queremos realzar las pocas luces que hay con este efecto, el objetivo es que no parezca que cada fuente de luz sea el sol.

Chromatic aberration: La aberración cromática simula un defecto de las lentes de las cámaras que son incapaces de converger todos los colores en un mismo lugar. Este efecto será añadido al camera shake cuando reciba el jugador daño.

En las siguientes figuras se pueden ver las virtudes de estos efectos cuando se comparan con una cámara sin post procesado. He elegido una escena en un interior bien iluminado para mostrar las virtudes del Bloom y la ambient occlusion, y un exterior sin iluminar para mostrar como el color grading puede cambiar completamente el tono de una escena.

Con esto se acabaron las diferentes características que he implementado en la cámara de la demostración, todas estas adiciones son parte del pilar del pulido del Game Feel el cual va a ser el que más nos diferencie de otros jugos en la competencia ya que la gran mayoría tienen entornos y controles en tiempo real pero la personalidad de cada juego se haya en este pilar.

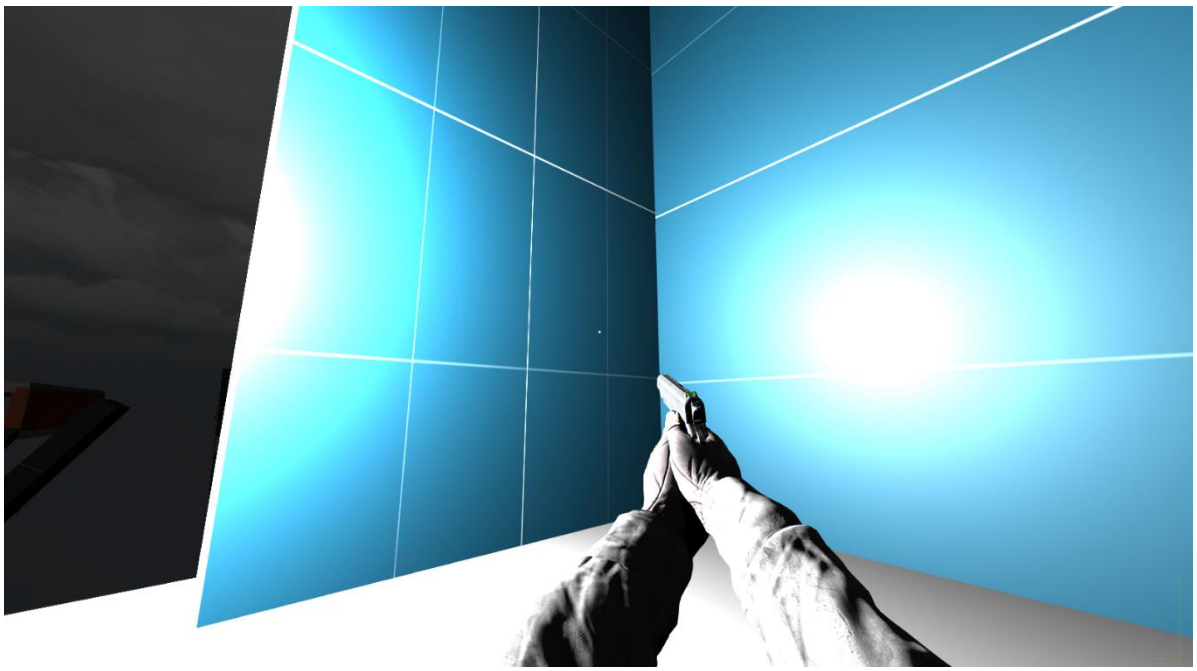


Figura 18: Interior iluminado sin post procesado

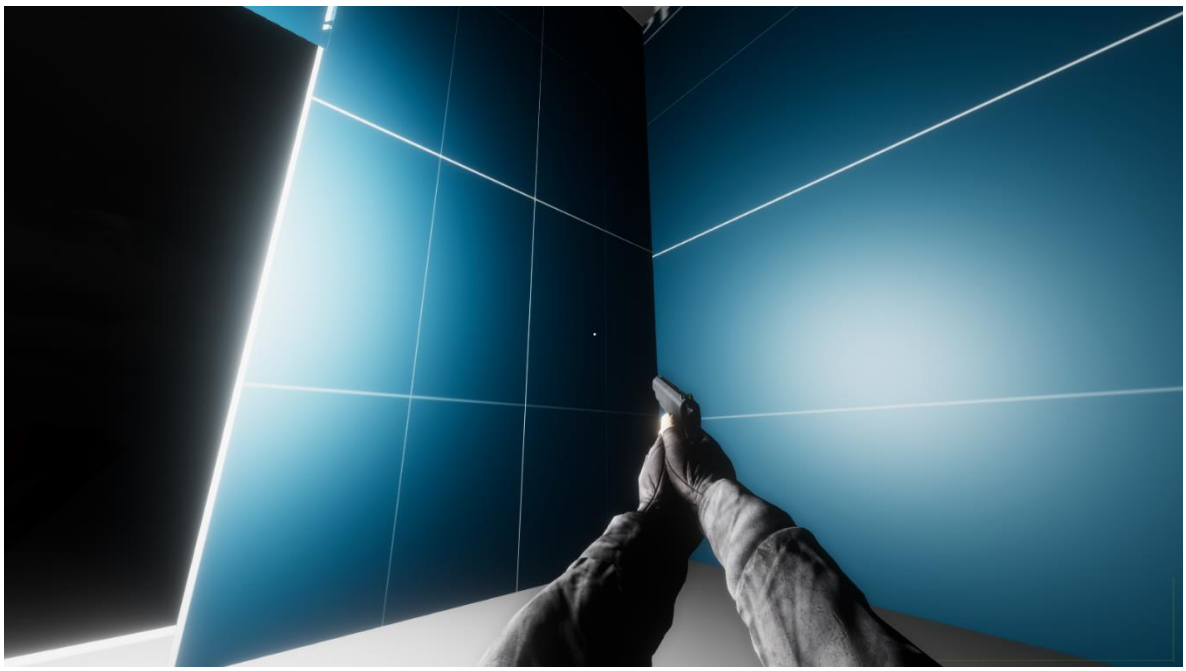


Figura 19: Interior iluminado con post procesado

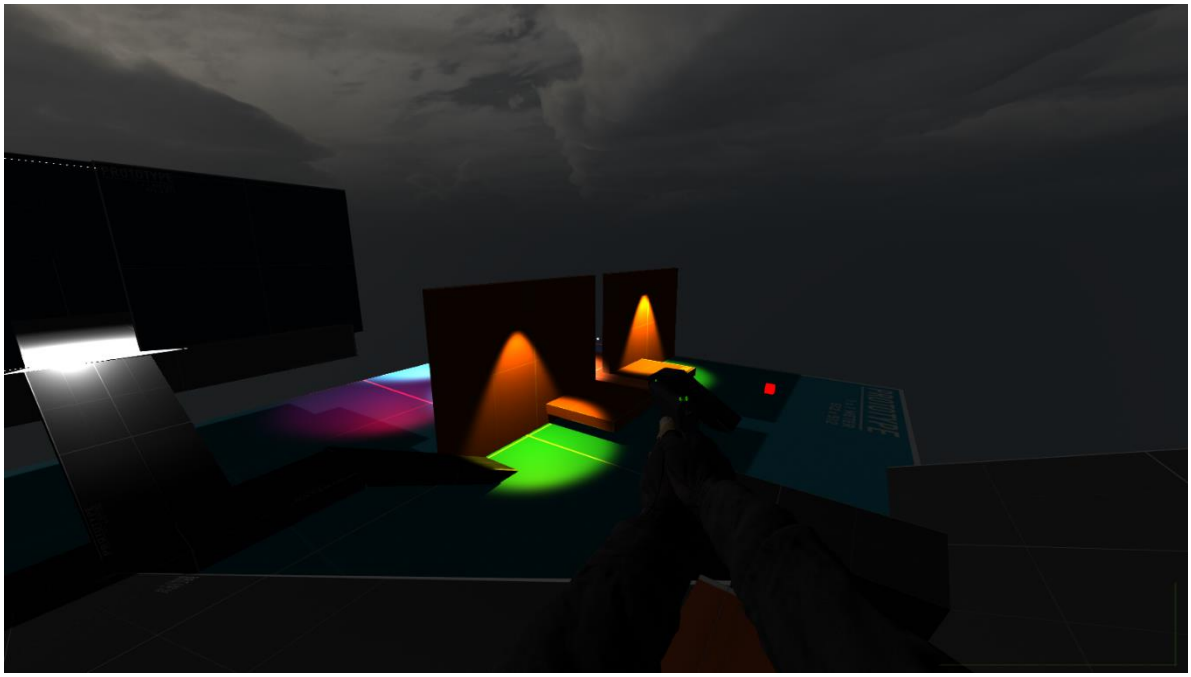


Figura 20: Exterior con poca luz sin post procesado

Figura 21: Exterior con poca luz con post procesado

Con esto vamos a comentar como fueron creadas las animaciones, seguimos en el pilar del pulido ya que sin estas el juego seria plenamente funcional.

La función principal de las animaciones será dar un claro feedback visual a cada una de las acciones del jugador y que simplemente con mirar a nuestro avatar el jugador sepa si estamos apuntando, corriendo, andando o cayendo entre otras acciones disponibles.

Las animaciones para videojuegos suelen hacerse en programas externos como Blender o Maya pero al ser programas mucho más enfocados en lo artístico y aprovechando que Unity da la opción de animar en el propio motor al final decidí hacer todo allí para evitar posibles problemas de compatibilidad.

Antes de empezar a animar debía decidir cuales iban a ser las acciones que el jugador iba a poder realizar en el juego. El jugador iba a ser capaz de: no hacer nada (idle), andar, correr, agacharse, apuntar, disparar, recargar y caerse. Para cada una de estas acciones necesitaremos una animación mínima y después ser capaz de relacionarlas todas entre sí.

El Animator de Unity es el lugar donde crear estas relaciones una vez ya están creadas las animaciones. El componente animator debe estar asociado al objeto que va a realizar las animaciones y una vez está asignado se crearán las animaciones.

A la hora de realizar las animaciones de un videojuego hay que tener en cuenta el contexto del mismo, si es un juego multijugador arcade como Call Of Duty tal vez rompa el flujo de juego rápido propio de la saga ya que las animaciones de recarga sean similares a las de un simulador de guerra como Arma 3 las cuales son largas y detalladas.

Al ser un juego de terror y querer hacer de nuestro avatar lo más expresivo posible las animaciones deberán ser detalladas, esto significa, crear animaciones que sean lo suficientemente lentas como para dar tensión, pero a la vez que sean realistas o que no sean lo suficientemente extravagantes como para romper la inmersión del jugador.

Las relaciones de las animaciones en el animador serían las que se observan en la siguiente figura:

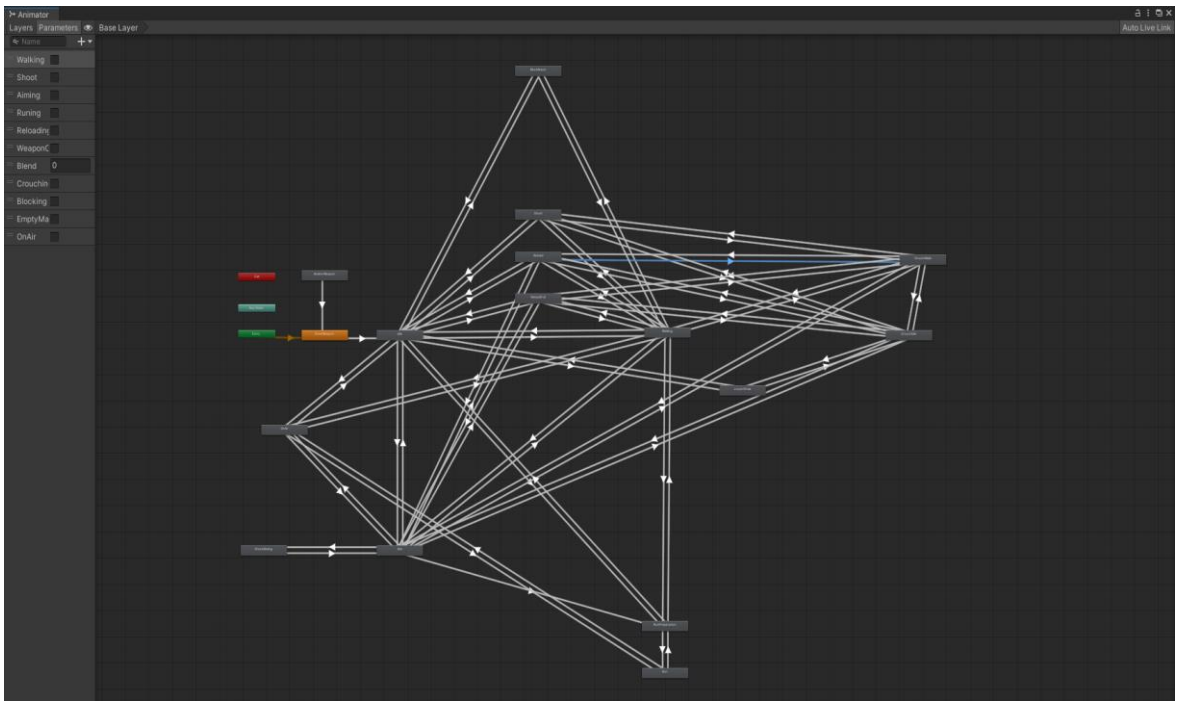


Figura 22: Relación de animaciones con todas las transiciones

Se han creado las mínimas relaciones necesarias para no añadir complejidad innecesaria a la relación de animaciones y aunque parezca realmente lioso representado gráficamente, cuando se está jugando se distinguen bastante bien las diferentes animaciones creadas y el estado en el que se encuentra el jugador.

Las animaciones pueden considerarse cortas teniendo como excepción las más largas que pueden ser las de recarga y la de no tocar nada. Aunque cortas, las animaciones tienen la duración exacta para generar en el jugador la sensación deseada. La animación de recarga dura bastante más de lo que puede durar en, como he dicho antes, Call Of Duty, pero no se aleja tanto de juegos del género como Resident Evil VII dónde el protagonista es una persona que no ha disparado armas en su vida.

De los siguientes apartados que debemos cubrir es el apartado sonoro. Debemos prestar especial atención al sonido de los pasos puesto que es un sonido que el jugador va a estar escuchando constantemente.

Debe estar lo suficientemente bien implementado como para que no rompa lo ya construido, es decir, habría que ejecutar un sonido de paso más o menos cuando la cámara va hacia abajo en el momento en el que simulamos dar un paso y acelerar la cadencia en la que se reproduce el sonido cuando pasemos a correr.

Otro sonido que tiene que encajar a la perfección es el de disparar la pistola, debe sentirse como una pistola de verdad por lo que se hizo bastante búsqueda hasta encontrar un sonido sin copyright para esto mismo. Como un único sonido de disparo se haría muy repetitivo se juntarán varios de armas de similar calibre y se modificara su pitch levemente para no causar fatiga si se dispara continuamente.

Con esto se ha concluido el desarrollo de la demo, ahora todas estas mecánicas hay que adaptarlas a las plataformas objetivo, PC y móvil.

Para PC contamos con un esquema de movimiento ya estandarizado entre los juegos en primera persona. Las teclas W A S D se encargan de manejar el movimiento del personaje. W y S se encargarán de recibir los inputs encargados de mover al personaje en el eje Z, en cambio la D y la A se encargarán de mover al personaje en el eje X. Por otro lado con la tecla shift mantenida y las anteriores teclas de movimiento correremos en vez de ir andando. El ratón manejará la cámara y la rotación del personaje en el escenario, aparte si apretamos el click izquierdo apuntaremos y con el derecho dispararemos, para recargar el arma usaremos la tecla R.

Realmente para PC los controles estaban bastante definidos desde el principio por esta estandarización, lo difícil llegó al intentar traducir todos estos inputs en la pantalla de un móvil. Necesitaré varios testers y recabar información de sus experiencias en la demo para ajustar los inputs de la manera más clara posible.

La demo en móvil comenzó con dos joysticks virtuales y un botón para cada acción. Pero ¿por qué dos joysticks? Por el simple tratamiento de las variables que manejan, al ser floats las variables que manejan el movimiento en el plano X Z, los botones no nos dejarían modificar a nuestra necesidad estos valores ya que muchas veces no querríamos un 1 absoluto sino un 0,25 para, por ejemplo, pasar por una zona cercana a una caída. En cambio, variables booleanas funcionan perfectamente con pulsaciones de botones para modificar si valor, por ejemplo, estar apuntando o no.

La distribución de estos joysticks en la primera versión de móvil fue que se ve en la siguiente figura:

Figura 23: Inputs en pantalla de la versión de móvil

Como podemos ver hay muchos botones y la barra de salud de abajo a la derecha no es del todo visible debido a que la mano derecha que maneja el joystick virtual derecho nos obstruye la visión.

Para poder correr es necesario pulsar el botón de arriba a la izquierda y para dejar de correr hay que volver a pulsarlo, es algo que varias personas de la muestra veían algo incomodo, por lo que una nueva versión para mejorar esta mecánica. La idea para esta segunda versión fue vincular la propia mecánica al joystick izquierdo, el de movimiento, para andar y correr directamente solo con ello.

Otra queja obvia es que no se especifica que hace cada botón por lo que unos iconos para cada botón serán necesarios para mejorar la interacción del jugador con la demostración. También deberían ser un poco más traslucidos para facilitar la visión a través de estos.

Se procede a hacer otra prueba con la gente para recabar información de como seguir mejorando la versión de móvil, a su vez se inicia la prueba en versiones de PC.

Por último, la modificación resulto lo bastante satisfactoria como para haber eliminado todos los problemas anteriores, aunque se hacían demasiado distantes los botones de acciones del arma por lo que se acabaron bajando para resultar más accesibles.

Por lo que la versión final de la demo en dispositivos móviles con respecto a sus inputs luce como en la siguiente figura:

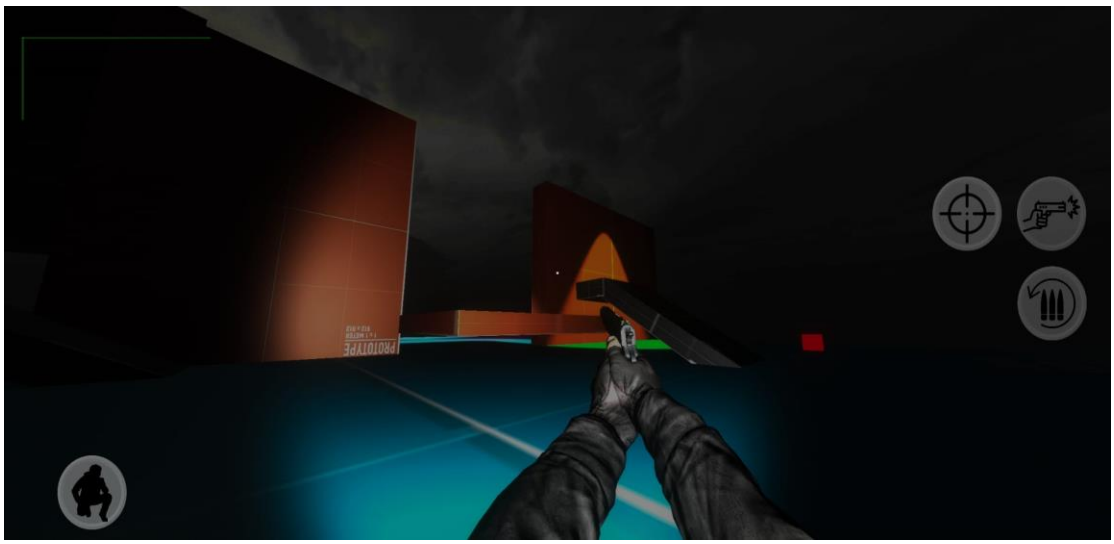


Figura 24: Nueva versión de Inputs en móvil.

Para hacer la experiencia del movimiento creado para móvil se crearon varias versiones del juego. La mayor problemática era ajustar la tasa de refresco adecuada dependiendo del dispositivo móvil usado. En mi caso, la pantalla de mi móvil tiene 90Hz por lo que tenía que activar la sincronización vertical a esa tasa de refresco, en cambio, para los testers este refresco podía variar por lo que para intentar mejorar el rendimiento en sus móviles implementé un sistema que es capaz de adaptar el límite de imágenes por segundo de la sincronización vertical a la tasa de refresco del propio dispositivo.

Dicho esto, vamos a comparar esta demo con otros juegos sus elementos escogidos para generar en el jugador ciertas sensaciones para contrastar el por qué lo hemos hecho en nuestra demo de esta manera.

5.3. Comparativa con otros juegos del mercado.

Para entender el porqué de ciertos elementos de la demo hay que viajar a uno de los juegos de terror más aclamados y disfrutados de principios de la década de 2010, me estoy refiriendo a Amnesia.

El Game Feel de Amnesia fue bastante innovativo para su tiempo ya que nuestro protagonista además de tener Amnesia puede ir perdiendo la cordura con el paso del tiempo debido a ciertas condiciones (permanecer en oscuridad por mucho tiempo y mirar fijamente a los monstruos principalmente). Para evitar esta cordura el jugador debía encender una lámpara antigua, en el caso de no tener material para hacerla funcionar poco a poco la visión se iba volviendo más y más borrosa y poco a poco ibas perdiendo capacidad de respuesta de la cámara ya que la sensibilidad de esta iba subiendo y bajando repentinamente. Por último, en los niveles más bajos de cordura se puede escuchar como el protagonista emite un ruido, este ruido no es otro que el de él mismo haciendo frotar sus propios dientes [29].

Como se puede ver no es algo realmente complejo de programar, un simple post procesado de la cámara, un cambio aleatorio en la sensibilidad de la cámara y un sonido, pero la gracia está en saber usarlo de la manera adecuada y Amnesia lo supo hacer de manera sobresaliente sobremodo para la época en la que salió.

En nuestro juego no existe este sistema de cordura puesto que el objetivo no era dificultar el control del jugador (sobre todo en móvil) sino que estos efectos pretenden crear sensaciones diferentes. El post procesado no indica nada al jugador en la demo creada excepto por un detalle, simplemente pretende añadir atmósfera, y crear un entorno más acorde con un escenario nocturno. La excepción la encontramos al recibir daño, en nuestro juego usamos el efecto de aberración cromática para saber que hemos sufrido daño, este efecto se puede ver en otros juegos en primera persona para simular el mismo efecto como en BlackMesa,

Tras el éxito de amnesia una segunda parte fue lanzada al mercado tres años más tarde. Esta secuela fue para muchos jugadores una decepción puesto que eliminó muchos de los elementos que caracterizaron a la primera entrega y que hicieron que muchas personas lo pasasen realmente mal jugándolo.

Sumado a que la empresa desarrolladora cambio, la mayor parte de las quejas fue la eliminación de este sistema de cordura quitando en gran parte esa tensión que generaba estar en ciertos lugares o presenciar sucesos extraños. En su lugar el juego que ese mismo año arrasó en el género de terror fue Outlast. Outlast fue el siguiente paso en lo que a inmersión en juegos de terror se refiere, poniendo al jugador en los pies de un periodista que debe investigar un psiquiátrico abandonado.

Nada más dar unos pasos en Outlast nos damos cuenta de algo que no existía en ninguno de los dos Amnesia, y ese es el Head Bobbing. Además, el personaje principal es realmente ágil, podemos esprintar saltar y girarnos hacia atrás mientras corremos para ver si estamos siendo perseguidos.

Y claro, con tanta agilidad y posibilidades puede que no se entienda dónde está la gracia de este juego y cómo pretende crear el sentimiento de terror o tensión si puedes huir de todo, bien, lo cierto es que uno de los mejores elementos de este juego son las persecuciones de la gente del psiquiátrico al protagonista por todo tipo de pasillos, el efecto de head bob genera una mayor inmersión en estas persecuciones puesto que nos ponen prácticamente en los ojos del protagonista.

Además de lo anterior expuesto contamos con una cámara con visión nocturna para poder ver en las zonas más oscuras, y tal y como lo haríamos en la vida real el protagonista de Outlast se acerca su cámara a los ojos para poder ver. Mecánicamente la cámara sirve como la lámpara de mano de Amnesia, solo que en vez de encontrar yesqueros para que funcione usamos baterías que encontraremos por el mapa.

Podemos apreciar también un genial uso del post procesado, nuestro personaje no pierde la cordura como en Amnesia, pero usa el truco de representar mediante post procesado el haber recibido daño. En este caso los efectos escogidos fueron un efecto viñeta rojo en el borde del cámara sumado a una sutil aberración cromática junto con un efecto de ojo de pez durante unos instantes.

Si lo comparamos con nuestro ejemplo, no quedan tan distantes puesto que no hay muchos ejemplos más parar representar visualmente que hemos recibido daño, además en Outlast no contamos con barra de vida por lo que exagerar estos efectos un poco es necesario para causar una diferencia notoria de cuando estamos en un estado normal a cuando estamos dañados.

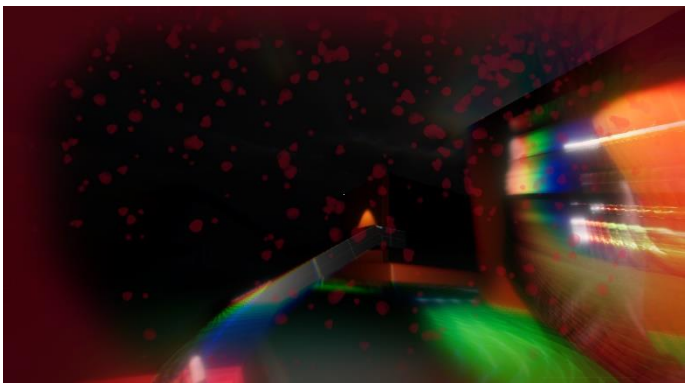


Figura 25: Indicador de daño de Outlast vs Indicador de daño de esta demo.

Como se puede observar en la demo de este trabajo de fin de grado tanto la viñeta como la aberración cromática estás mucho más acentuadas que en Outlast, esto es para magnificar lo peligroso que es recibir daño, además, he prescindido del efecto de ojo de pez puesto que sería demasiado caos visual para el jugador.

Como se puede ver, tanto en Outlast como en Amnesia el protagonista en ambos casos está completamente indefenso, en nuestra demo le hemos dado al jugador la capacidad de defenderse, como ya he comentado al tratarse de captar la inmersión que tiene un juego de terror debemos fijarnos en aquellos que tengan un sistema consistente sin llegar a convertirse plenamente en un shooter multijugador.

Tenemos varios ejemplos, unos más centrados en el shooting que en otros, para esta demostración pretende centrarse en aquellos que no exploran tanto esta faceta del gameplay. Como ejemplo principal tenemos al aclamado Resident Evil 7 el cual fue el primero en la franquicia en incorporar toda la jugabilidad en primera persona y con un nuevo personaje principal.

Hasta el Resident Evil 7 todos los anteriores habían sido en tercera persona, debido a la perdida de fans, pocas ventas y malas reseñas acusando a Capcom de la perdida de esencia de la saga. Capcom se propuso dar un toque nuevo a Resident Evil el cual lanzó su primera demo entre polémica con la similitud de lo que iba a ser el nuevo juego de Hideo Kojima [30].

El cambio de perspectiva no fue lo único que cambió en la saga, tanto en el 5 como en el 6 el terror y elementos survival horror habían desaparecido enfocándose más en elementos de juegos de disparos de acción convencionales.

Por eso en la demo vamos a ser capaces simplemente de llevar una pistola. En Resident Evil 7 tenemos un arsenal de armas bastante más amplio, pero nuestro protagonista no ha recibido instrucción militar como otros en la saga por lo que, cuando estamos usando una pistola, se siente lento y poco preciso ¿Cómo lograron esto?

Anteriormente hemos hablado de la duración de las animaciones y lo cierto es, que mucho de ese efecto es conseguido mediante estas, Ethan (el protagonista de Resident Evil 7) en ningún momento apunta a través de la mirilla de metal del arma, en su lugar, cuando presionamos el botón de apuntar, un efecto de zoom aparece para hacer énfasis a la zona central de la pantalla para favorecer el apuntado y la dispersión de bala (posibles lugares en la pantalla donde acabe la bala disparada) se minimiza. Este efecto llega a ser eliminado si Ethan permanece quieto durante un pequeño instante antes de disparar.

Como podemos observar, al tener que permanecer quietos para disparar exactamente dónde queremos estamos poniendo al jugador en un aprieto, ya que estamos añadiendo tensión a la mecánica que nos debería proporcionar más seguridad.

Aparte, las animaciones de recarga de Ethan no son nada profesionales, mueve de más las manos y, aunque no sean especialmente largas o lentas (puesto que es inexperto, pero entiende cómo funcionan) encaja perfectamente con el personaje y le dan personalidad.

Cuando llegó el primer DLC de Resident Evil 7 el protagonista cambió, ahora el jugador encarna a Chris Redfield, un antiguo protagonista de otros Resident Evil pero esta vez en primera persona. Chris, al ser un militar estadounidense tiene bastante más habilidades que Ethan en cuanto al uso de armas. La primera diferencia que podemos notar es que Chris si es capaz de apuntar y por lo tanto tiene precisión perfecta en el momento en el que el jugador decida apuntar con la mira.

Además, en el caso de jugar como Chris son mucho más rápidas y metódicas en comparación de las que podemos observar con Ethan. En la demostración jugable somos capaces de apuntar con la mira, pero si queremos disparar para tener una precisión buena deberemos permanecer quietos ya que en el momento en el que empecemos a andar nuestra pistola se moverá dificultando la precisión, sería algo así como una mezcla de estos dos aspectos de Resident Evil 7.

Figura 26: Diferencias entre apuntar con Ethan o Chris en Resident Evil 7.

Como hemos podido observar con el paso del tiempo y diversos lanzamientos de títulos de juegos de terror, se han ido añadiendo y conservando estos detalles que hacen que el jugador se sienta más inmerso en la experiencia. En nuestro caso hemos ido añadiendo las características esenciales para lograr que el movimiento de nuestro personaje se sienta como de un juego de terror, con obvias modificaciones para crear una experiencia completamente nueva.

6. Estudio económico

En este apartado ahondaremos en la repercusión económica que tendría este proyecto de ser plenamente convertido en un videojuego completo. Entre las cosas a tener en cuenta nos encontramos licencias de software usado, tiempo trascurrido hasta finalizar los objetivos de desarrollo y posible ayuda externa en el caso de que el desarrollo no fuese cien por ciento autónomo.

A continuación, se expone un gráfico con el porcentaje de tiempo en el que se ha invertido en este proyecto dividiéndose en 3. La implementación de la demostración jugable, la investigación realizada y la realización de la memoria o documento del trabajo de fin de grado.

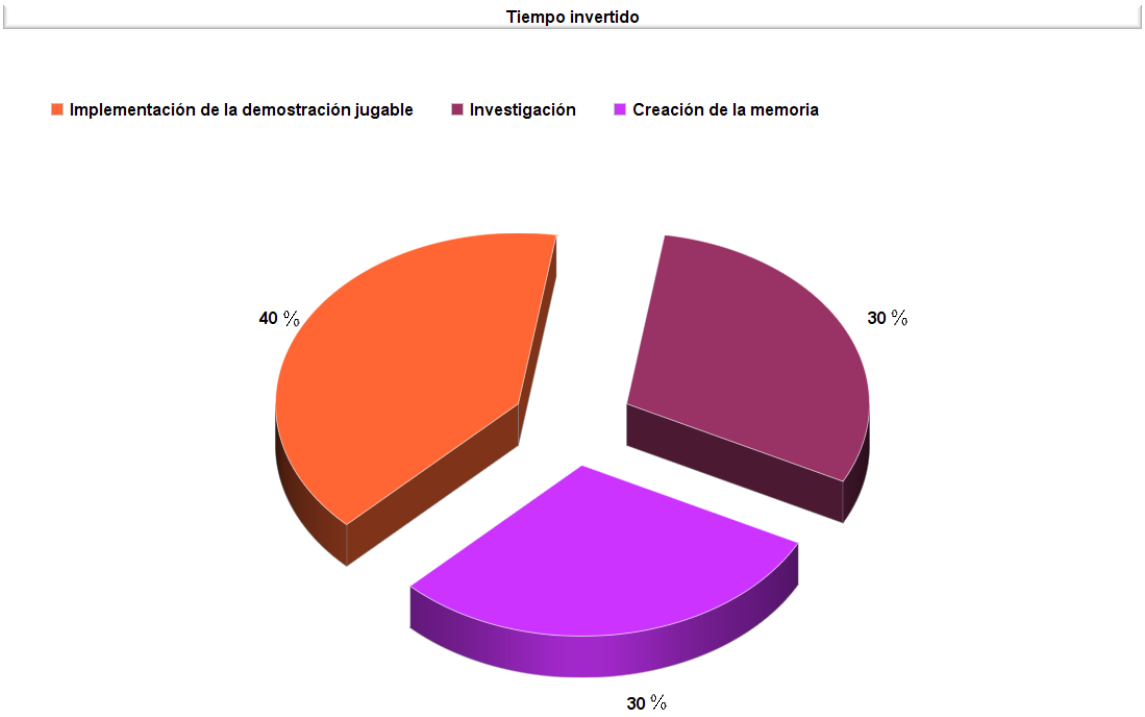


Figura 27: Gráfico del tiempo distribuido en este proyecto.

Contamos con 175 horas en total por lo que para la implementación de la demostración se invirtieron alrededor de 40 horas, en investigación de la materia se invirtieron unas 52,5 horas y para la creación de la memoria finalmente otras 52,5 horas.

Las herramientas utilizadas para el desarrollo de este trabajo han sido las que se pueden observar en la siguiente tabla en la que además se mostrará el precio de cada una de las licencias que correspondan.

Herramienta	Utilidad de la herramienta	Precio
Unity Pro	Motor de juego.	1.800 € anuales [31]
Blender	Modelado y rigging.	Gratis [32].
Photoshop	Elementos de la interfaz de usuario.	24,19 € mensuales [33].
Office	Realización de la memoria del documento	4,20 € al mes [34].

Tabla 1: Costes de software y herramientas.

Al haber estado desde comienzos del segundo cuatrimestre realizando la implementación de la demostración, pero centrándome cada mes o varios meses en distintos apartados del juego y volviendo a retomar algunos meses más tarde para mejorarlos o volver a trabajar en ellos los costes se dividirán en torno a los meses de marzo a agosto como se puede ver en la siguiente tabla de costes:

-	Marzo	Abril	Mayo	Junio	Julio	Agosto	TOTAL
Animaciones	-	-	-	-	-	-	-
Rigging	-	-	-	-	-	-	-
Modelado	-	-	-	-	-	-	-
Creación de interfaz y menús	-	24,19€	24,19€	24,19€	24,19€	-	96,76€
Programación	1800€	-	-	-	-	-	1800€
Memoria	4,20 €	4,20 €	4,20 €	4,20 €	4,20 €	4,20 €	25,2€
							1923,96 €

Tabla 2: Costes de software a lo largo del desarrollo.

Como la subscripción para Unity es anual, esos 1800 € en programación podrían haber sido incluidos en la parte de animación puesto que se usaría el mismo programa (Unity) pero lo he decidido poner en el apartado de programación puesto que como animar se puede conseguir con Blender el cual es gratuito nos dejaría solo disponible el apartado de programación para apuntar el dinero que cuesta la subscripción anual de Unity.

Siendo una demostración jugable y no un juego como tal, centrándonos específicamente en la locomoción del jugador y la mecánica de disparo 1923,96 es un precio bastante acertado para el trabajo realizado en las 175 horas que duró el completo desarrollo de todo el trabajo de fin de grado.

Si el trabajo hubiese sido expandido para crear un juego completo con niveles, historia etc. Deberíamos contratar a varias personas para realizar las tareas necesarias.

Habría que contar con los datos de los sueldos para pagar a los respectivos artistas y programadores, para la realización de este supuesto juego de terror necesitaríamos un modelador 3D, una persona encargada de hacer el esqueleto del modelo, un animador, un artista de HUD, un programador y un game designer.

Actualmente los salarios promedios de los empleos mencionados son:

TRABAJO	SALARIO MEDIO MENSUAL	SALARIO MEDIO ANUAL
Modelador 3D	1.939 €	23.272 €
Rigger	1.558 €	18.696 €
Animador	1.500 €	21.000 €
Diseñador de HUD	2.333 €	28.000 €
Programación	1.583 €	19.000 €
Game Designer	2634 €	31.615 €

Tabla 3: Salarios de los trabajadores.

Como poco un año de desarrollo sería lo que se podría tardar en tener un juego mínimamente preparado con un nivel detallado y con su versión porteada a móvil de la manera correcta.

Para los elementos que requieran más tiempo como programar todo el videojuego podemos crear contratos de trabajo fijo mientras que para elementos más puntuales podríamos contar con gente trabajando a tiempo parcial.

Como método de ingresos de este supuesto juego podríamos tener dos esquemas diferentes para las dos plataformas en las que estaría nuestro juego. Por un lado, tendríamos la versión de PC que tras pasar por diversas fases para conseguir que se pueda vender en Steam se podría comprar por un precio fijo de unos 5 euros ya que estamos hablando de un juego realizado en un año y que también se podría conseguir en móvil de otro modo.

Estamos hablando de que en versión móvil el juego se podría descargar gratuitamente y ser nuestra fuente de ingresos los anuncios que se reproducirían cada vez que el jugador muera (por ejemplo).

También podría ofrecerse una versión del juego en móvil libre de anuncios por una cantidad de dinero fija, tal vez más barata que la versión de Steam puesto que gráficamente es inferior, pero libraría a los jugadores de la versión gratuita de estar viendo anuncios cada vez que mueran.

7. Resultados

Los resultados obtenidos del playtest realizado fueron positivos en todos los sentidos y sobre todo esperables.

Se hicieron 2 playtests principales de varias sesiones alterando ligeramente varios parámetros dependiendo del tipo de playtest del que se tratase. Tras dejar a los testers moverse libremente por el entorno digital y hacer lo que quisieran se les presentó un formulario (Ver Anexo III) que rellenar para obtener datos de la sesión, el cuestionario que debían rellenar contenía dos preguntas simples y dependiendo del tipo de playtest del que estuviésemos hablando serían diferentes.

Estos playtest pretenden enfocarse en los pilares de la sensación de juego, expuestos al principio de este trabajo de fin de grado, pero, los pilares que realmente se pueden modificar de la demostración creada son los de control en tiempo real y pulido, por lo que los playtest serán únicamente de dichos pilares.

Finalmente dependiendo de las respuestas obtenidas por la gente se reajustarían los parámetros para mejorar y tratar de captar mejor las sensaciones de los jugadores para acercarlas lo máximo posible a la experiencia que queremos generar en ellos.

7.1 Playtest de pulido

Este será el gráfico donde ubicaremos cada uno de los resultados de cada uno de los diferentes playtest de pulido siendo un total de 10 los testers que lo realizaron:

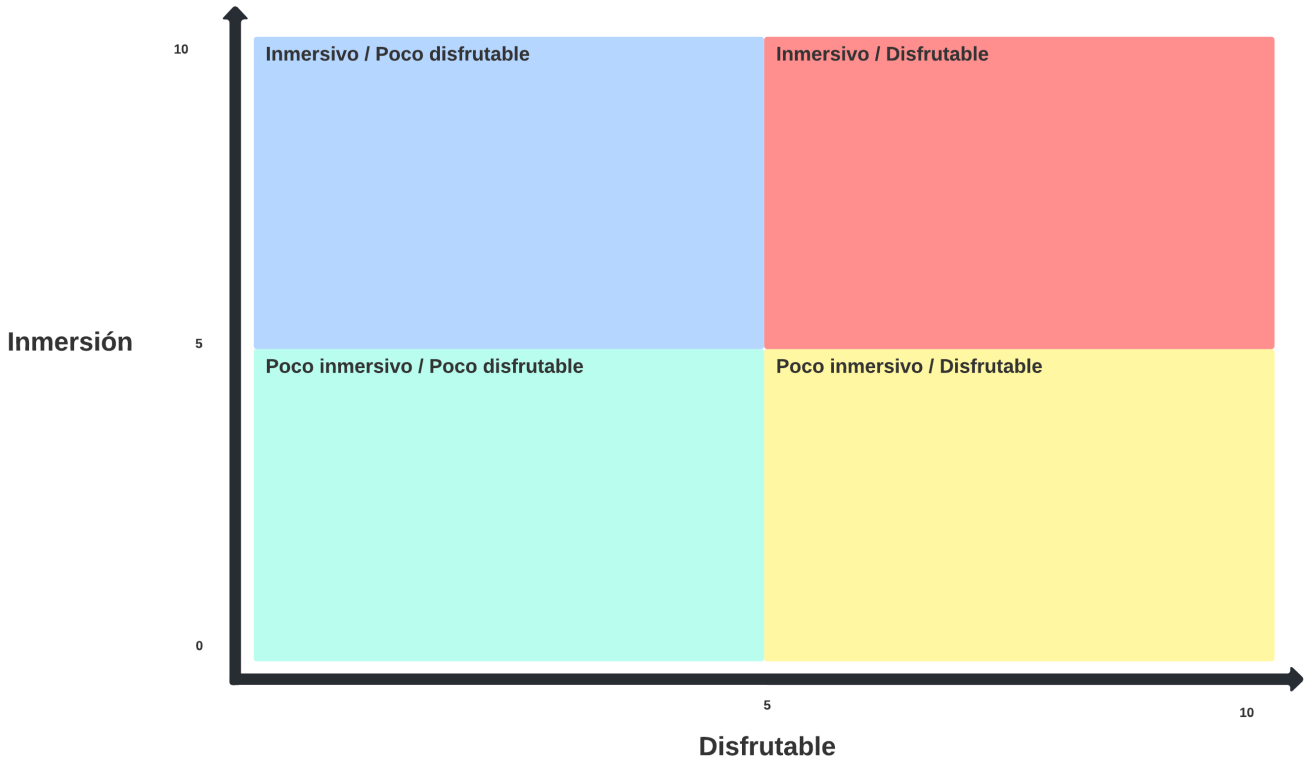


Figura 28: Ejemplo del gráfico de playtest (pulido).

En la primera ronda de playtest se les dará a los testers una versión del juego en la cual contamos con el menor pulido posible, es decir, sin animaciones, sin post procesado, sin movimientos de cámara y sin sonido, Les dejaremos 10 minutos para moverse por el entorno y 5 para realizar el cuestionario.

Una vez realizado el playtest se descansarán otros 5 minutos para entrar a la diferente versión más descansado puesto que hay muchas variantes diferentes que testear y muchos resultados que obtener.

Los resultados arrojados fueron los siguientes

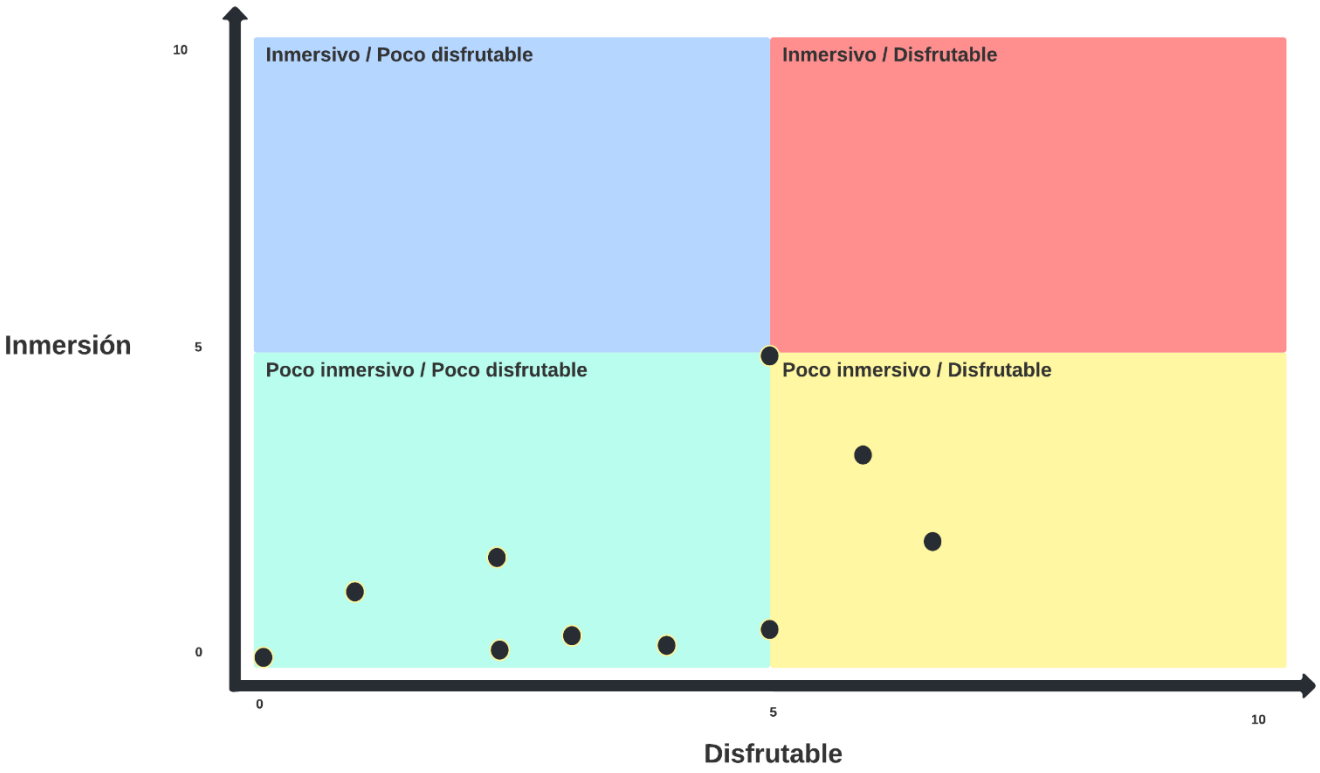


Figura 29: Resultados primer playtest (pulido).

Como se puede observar los resultados son malos, pero esto es esperable ya que sin ningún tipo de pulido apenas existe un feedback en el cual el jugador intuya que sus acciones influyen en la demo. La demo es funcional, pero sin todos los extras creados para generar inmersión en el jugador esta falla completamente en el intento.

Muy pocas personas de las escogidas para el playtest finalizaron pensando que la experiencia había resultado ser disfrutable a pesar de no tener a penas pulido, esto puede ser posible debido a que los perfiles de los jugadores escogidos no es el mismo y probablemente no estén familiarizadas con juegos en primera persona y los estándares actuales de la industria.

La segunda versión que probarán los playtesters se centra en dar el pulido necesario para hacer el movimiento por el escenario algo menos monótono y separarse de la idea de que estas controlando a una cámara y empezar a crear sensaciones, en este caso la de andar y correr. Por lo tanto, en esta versión se activarán los movimientos que realiza la cámara al movernos por el entorno de pruebas lo cual se espera que mejore en el eje de inmersión.

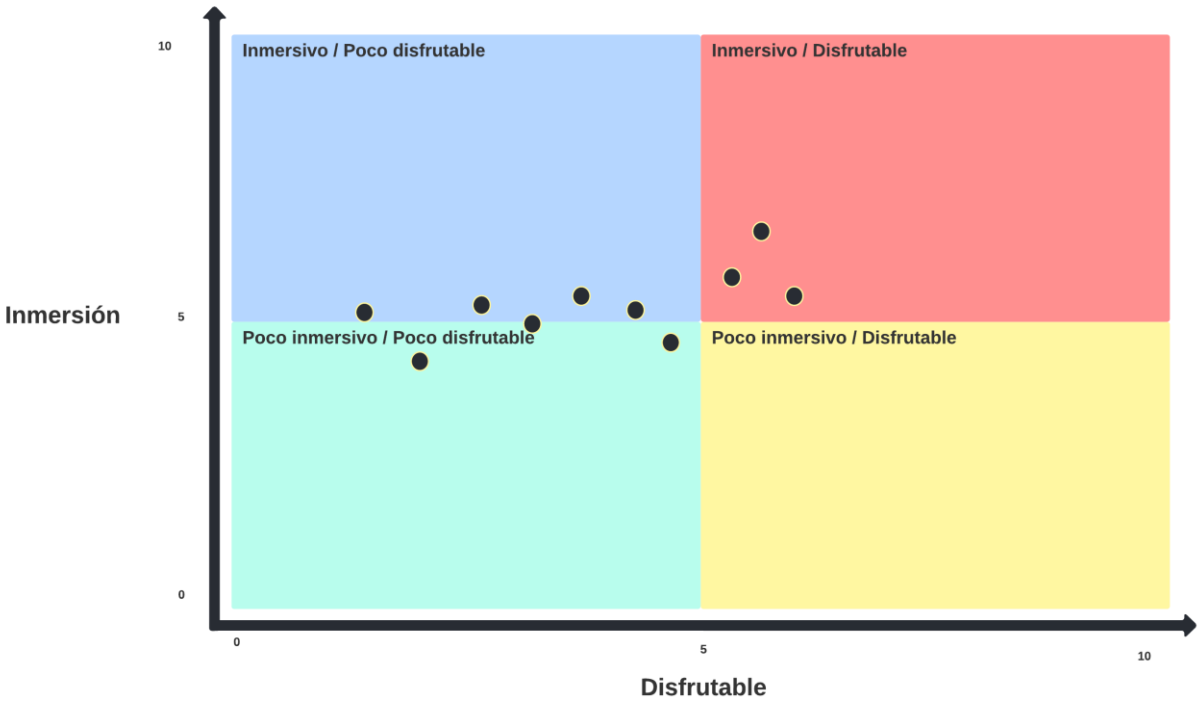


Figura 30: Resultados segundo playtest (pulido).

Como hemos comentado, hemos mejorado en inmersión, esta sigue siendo bastante pobre puesto que la demo sigue estando en un estado bastante poco pulido, pero con añadir los movimientos de la cámara realmente ayuda a sumergir a la gente en la experiencia, al menos más que en el primer playtest.

También podemos observar que la experiencia se ha hecho más disfrutable por el simple hecho de hacer del andar y correr por el escenario algo más visualmente dinámico para el jugador.

El tercer playtest añadirá el modelo de los brazos del avatar, pero sin ningún tipo de animaciones ni movimientos de cámara al realizar la acción de disparar (como en los anteriores playtest).

Se espera unos resultados similares al anterior playtest pero saber cómo es el arma que dispara y la apariencia de los bazos de nuestro avatar tal vez tenga más significancia de la que se cree en un principio.

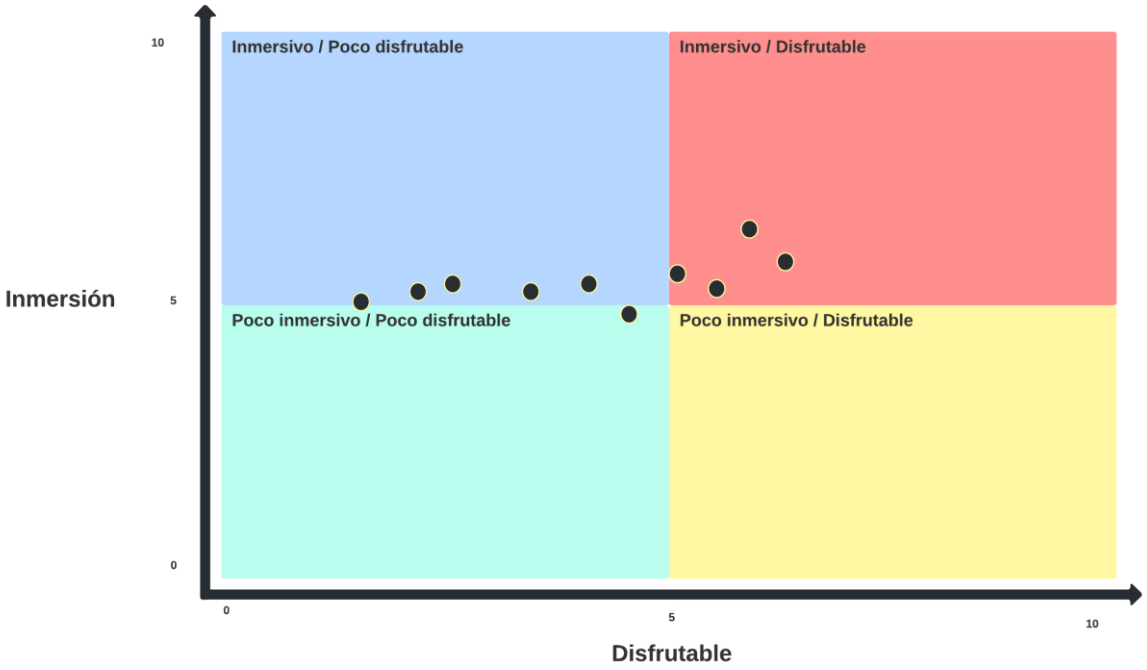


Figura 31: Resultados tercer playtest (pulido).

Los valores de inmersión como eran previsibles han mejorado ligeramente mientras que en cuestión de que la experiencia sea disfrutable hemos recibido resultados muy similares al anterior test. Ver los brazos del avatar no es suficiente para añadir más inmersión a la experiencia jugable que se está creando.

El siguiente playtest será el penúltimo y en el que activaremos las animaciones tanto para mecánicas de movimiento como en las de disparo, se espera un gran avance tanto en inmersión como en hacer la experiencia mucho más disfrutable.

Los resultados fueron los siguientes:

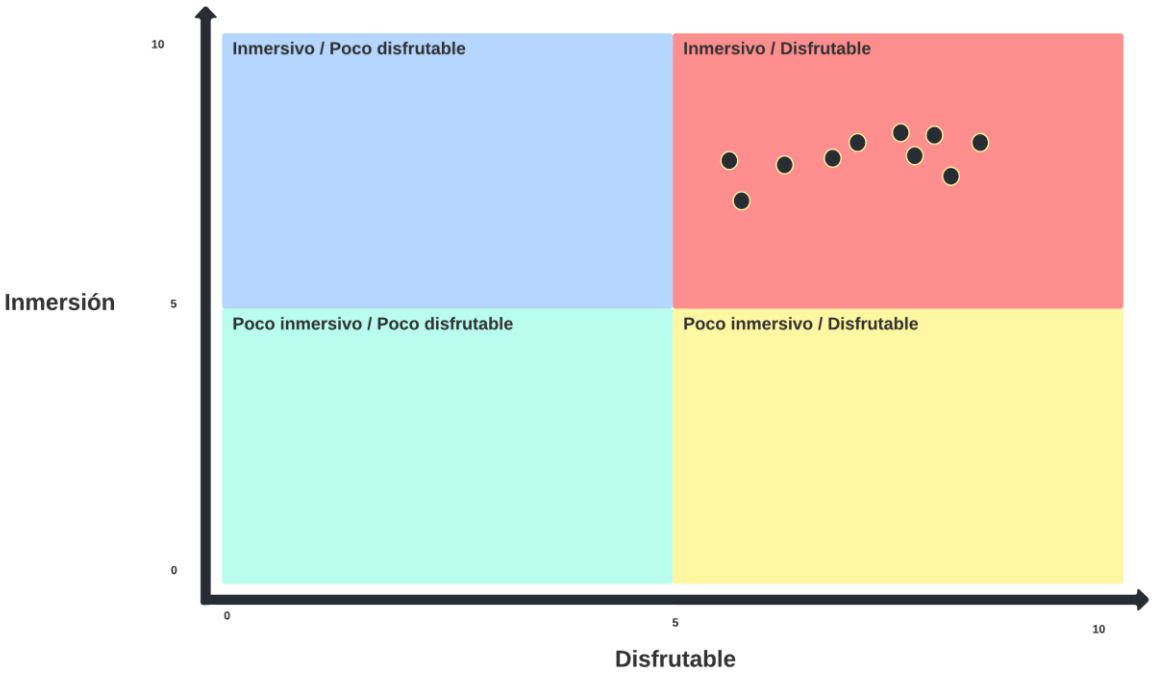


Figura 32: Resultados cuarto playtest (pulido).

Como se puede apreciar, las animaciones han generado un cambio muy abrupto en los resultados de este playtest, ver cómo reacciona el avatar a cada una de las acciones que realizan los testers les ha convencido lo suficiente como para ponerse todos en el rango que más nos favorece.

Era esperable un ascenso en ambos rangos (inmersión y experiencia disfrutable) pero se pensaba que alguno de los tester siguiera supendiendo la experiencia en al menos alguno de los dos rangos ya que aun han visto la versión final, la cual será el siguiente y último playtest a realizar por ellos.

En este último playtest todas las opciones creadas para la demo estarán activas, los resultados se esperan que mejoren incluso más los resultados del anterior playtest. Los resultados han sido los siguientes:

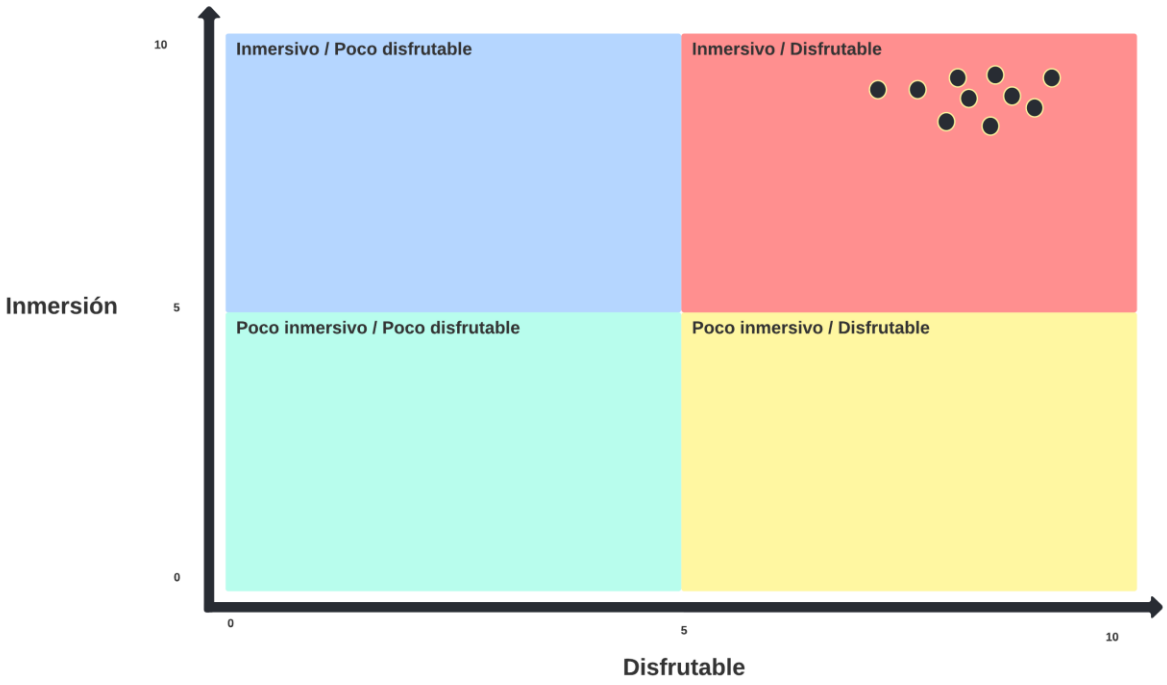


Figura 33: Resultados quinto playtest (pulido).

En este último playtest tanto la inmersión como el disfrute ha mejorado muchísimo, incluso teniendo valores bastante decentes en el anterior playtest.

7.2 Playtest de control en tiempo real

En este playtest se van a cambiar parámetros como la velocidad del jugador y como los inputs cambian los parámetros de las mecánicas a las que están asociadas.

Se realizarán dos playtest simplemente, en los cuales se pretende generar en el jugador la sensación de que, aunque animaciones y pulido sea siempre el mismo, estos valores que son imperceptibles a simple vista pueden cambiar el género de nuestro juego.

Esta será la tabla en la que se colocarán los resultados de los playtest, en este caso contamos con 5 playtesters:

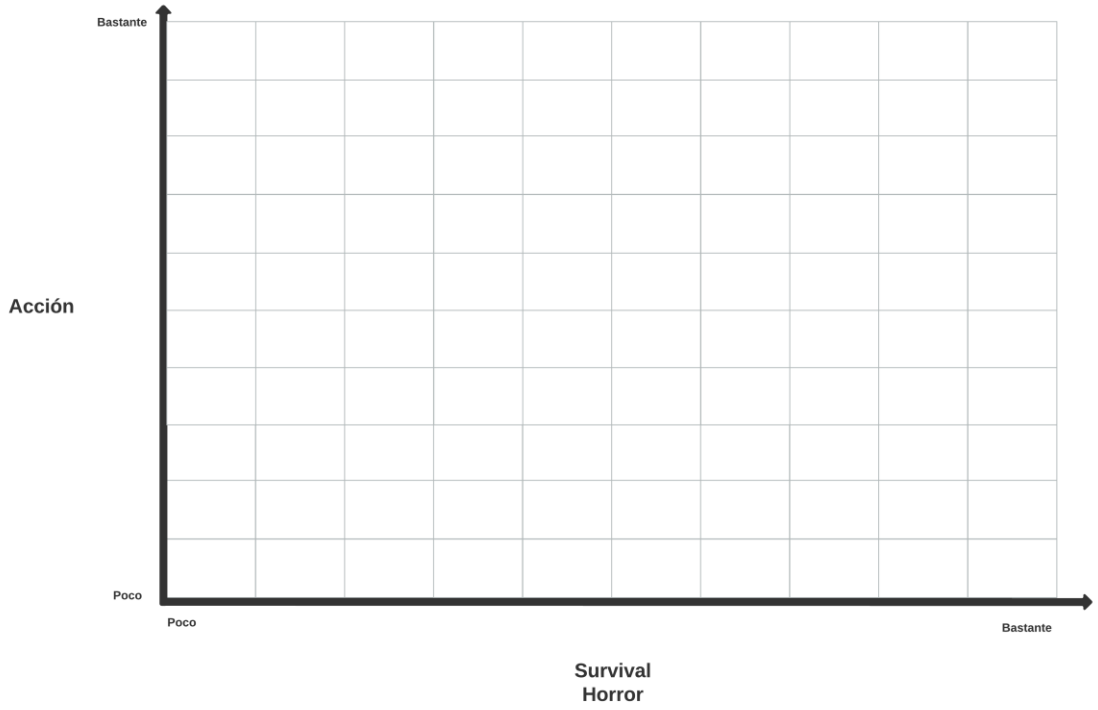


Figura 34: Ejemplo de gráfico de playtest de control en tiempo real.

El número de playtesters ha sido reducido ya que deben ser personas que puedan diferenciar con facilidad géneros de videojuegos y que al menos tengan un cierto historial de videojuegos a sus espaldas para poder comprender a que género nos referimos con survival horror o acción.

El primer playtest realizado será exactamente la misma demo que la del último playtest de pulido pues los valores escogidos para crear una experiencia survival horror con toques de acción, siendo esta un género secundario en la experiencia.

Los resultados del primer playtest fueron los siguientes:

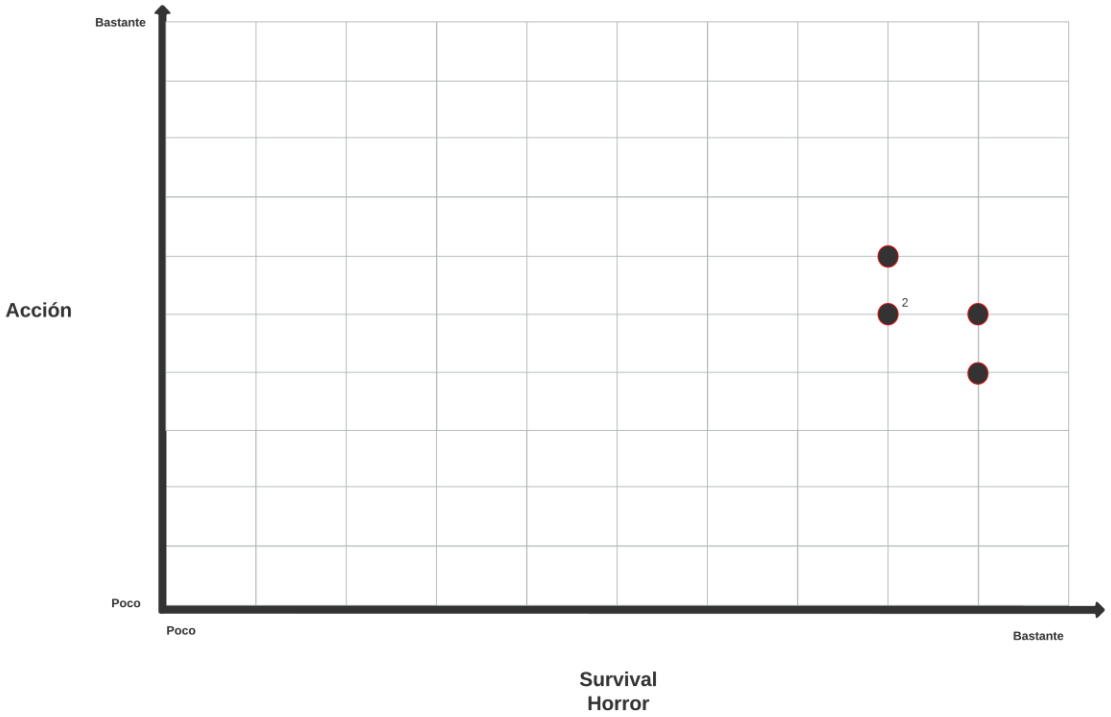


Figura 35: Resultados primer playtest (Control en tiempo real).

Como se puede observar y como era previsible, en este playtest debido a los parámetros usados para la demo, priorizamos un ritmo lento y sosegado, pero manteniendo algo de acción en la mezcla con un sistema de disparo no tan rápido como el de un juego centrado en mecánicas shooter y de acción, pero aun así fluido y responsivo.

En el siguiente playtest modificaremos los valores para que la experiencia sea mucho más ágil, tanto para parámetros relacionados con el movimiento como para las mecánicas de disparo. El objetivo es crear un flujo de juego mucho más rápido y se esperan resultados más enfocados al lado de la acción.

Los resultados son los siguientes:

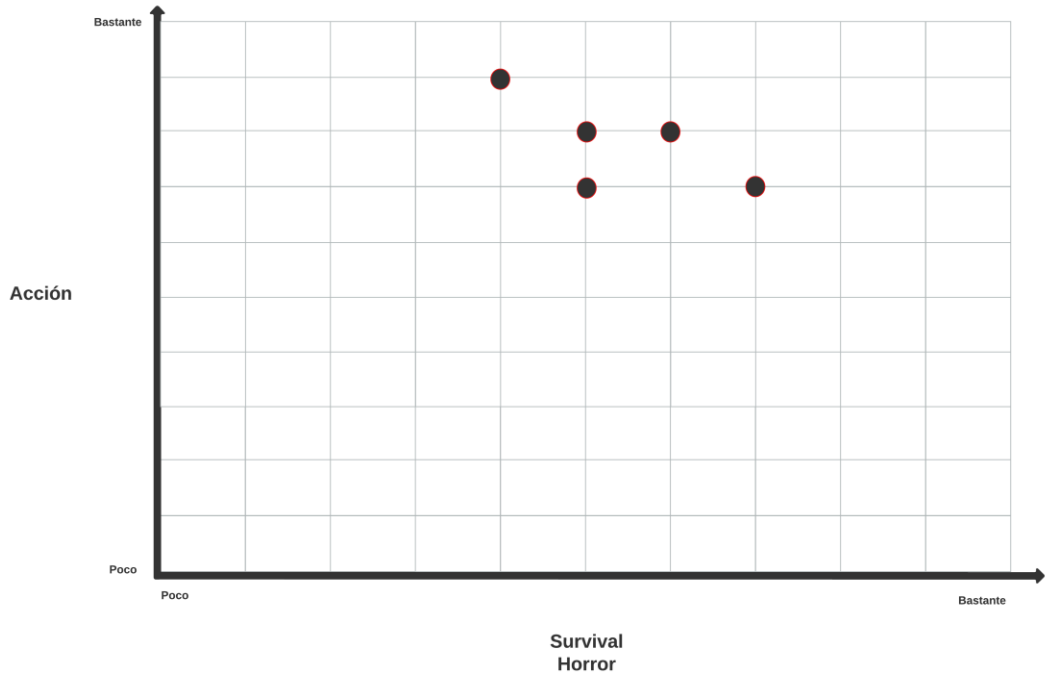


Figura 36: Resultados segundo playtest (Control en tiempo real).

Como podemos observar, la percepción de los testers ha cambiado. Ahora la acción toma un foco más principal a pesar de contar con un pulido que busca generar una sensación de juego del género de survival horror. La velocidad en la que se ejecutan todas las acciones, al ser aceleradas eliminan cierta tensión en las mecánicas del juego ya que limitan los momentos en los que es posible que el jugador este vulnerable, y aunque no haya amenazas puesto que es una demo, la diferencia es notoria.

En el último playtest que vamos a realizar para el control en tiempo real va a ser justo lo contrario al anterior. Vamos a hacer que todas las mecánicas sen mucho más lentas que incluso en el primer playtest de control en tiempo real. Con esto se espera obtener peores resultados en el apartado de acción, pero pronunciar hasta los niveles del primer playtest o incluso más los niveles de Survival Horror de la demo.

Los resultados han sido los siguientes:

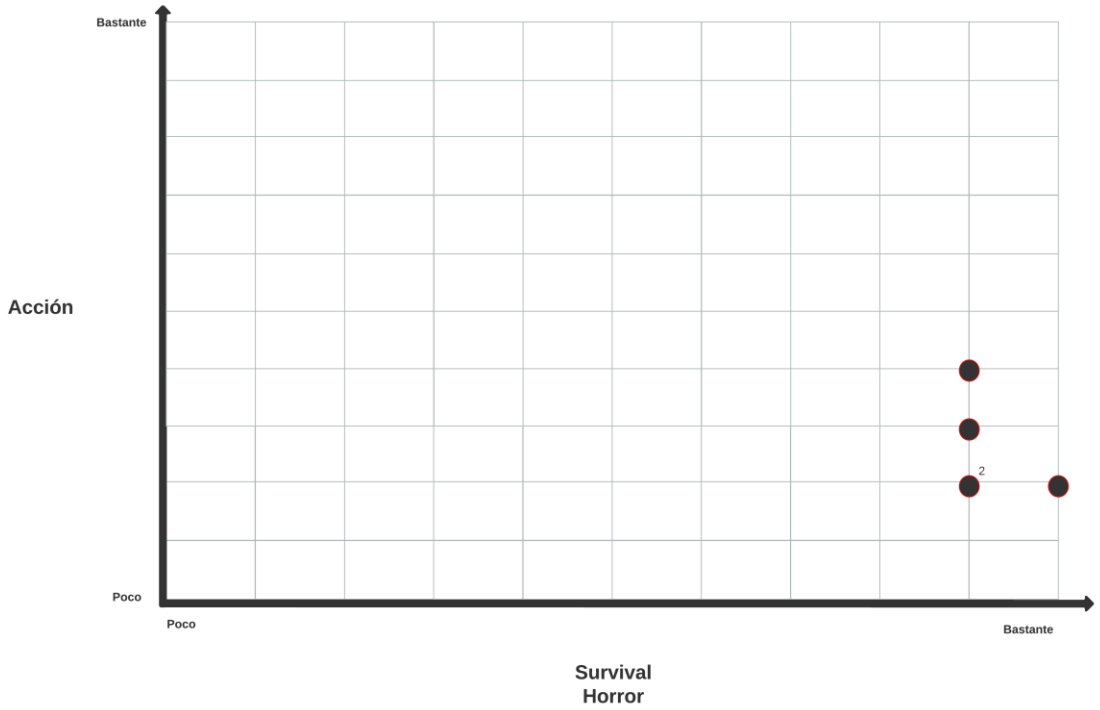


Figura 37: Resultados segundo playtest (Control en tiempo real).

Al modificar el flujo de jugo para crear una experiencia más pausada y con más tiempos dónde el jugador es vulnerable, la confrontación puede que no sea lo más viable como en el anterior playtest dónde el movimiento y disparar el arma es ágil. En esta demostración al movernos y disparar en espacios más largos de tiempo el posicionamiento será mucho más importante que vaciar el cargador.

8. Conclusiones

La Sensación de Juego o Game Feel es algo que todo desarrollador debe conocer en algún momento de su formación, puede que de manera inconsciente lo aplique desarrollando sus juegos, pero conocer en su plenitud las técnicas usadas por otros desarrolladores y su sentido e historia hace que los juegos creados aumenten su calidad de manera notoria.

Para cada tipo de videojuego existe una sensación que se pretende generar en el jugador, el cómo se consigue es algo realmente complicado de averiguar. Son 3 los pilares que conforman esta sensación de juego y sus componentes son 7 (como hemos visto en la introducción y estado del arte). Conocer estos detalles nos hará más fácil identificar los apartados de nuestro juego necesario para generar las sensaciones deseadas.

En la demo creada para este Trabajo de Fin de Grado se han tenido desde el comienzo en cuenta dichas características y se ha pretendido aplicarlas a una demo de un juego de terror o survival horror. Comparando esta demo con anteriores proyectos y juegos de la industria para dar un enfoque más realista y enfocado a juegos existente en el mercado, la realización de esta ha demostrado que conocer y saber aplicar los conocimientos de la sensación de juego nos ayudará para obtener no solo las sensaciones que buscamos de manera más concreta, organizada y en definitiva profesional.

Como se ha podido ver en los resultados, un enfoque claro desde el comienzo de desarrollo y tener en cuenta los elementos que conforman la sensación de juego nos permitirá llegar de manera satisfactoria a la sensación que queremos transmitir a los jugadores.

Anexo I – Propuesta de proyecto final:

1. TÍTULO DEL PROYECTO

Estudio del efecto de diferentes parámetros de diseño de videojuegos en el Game Feel o Sensación de Juego.

2. DESCRIPCIÓN Y JUSTIFICACIÓN DEL TEMA A TRATAR

Me parece un tema muy interesante e importante ya que normalmente no tenemos tiempo de prestar atención a los detalles en los proyectos de la universidad. Crear un buen game feel puede marcar la diferencia y hacer que nuestro juego sea mejor valorado que otro que no ha tenido el tiempo, conocimiento o las ganas de implementarlo correctamente.

3. OBJETIVOS DEL PROYECTO

Los objetivos de este proyecto son:

1. Creación de una demostración jugable de un juego de terror basándonos enteramente en dar una Sensación de Juego especifica al jugador.

2. Analizar juegos en el mercado del mismo género y contrastar con datos las decisiones tomadas.

3. Comparar con proyectos personales anteriores y por qué la nueva es mejor en cuestión de Sensación de juego

4. Analizar con estadísticas los resultados obtenidos y obtener una conclusión basada en experiencia de usuario.

4. METODOLOGÍA

La metodología se establecerá en las primeras fases del proyecto.

5. PLANIFICACIÓN DE TAREAS

Las tareas quedan predefinidas de manera global en los objetivos. Serán fijadas de forma concreta durante el desarrollo del proyecto.

6. OBSERVACIONES ADICIONALES

Anexo II - Reuniones

1. **Fecha:** 24 de septiembre de 2020. **Modo:** online. **Programa:** Microsoft Teams.

   Contenido: Definimos el tema a tratar en el trabajo, se realizó la solicitud pertinente y se indagó en posible documentación que podría resultar útil si la propuesta fuera aceptada.

   Una de estas fue la posibilidad de realizar la lectura del libro de Steve Swink Game Feel: A Game Designer's Guide to Virtual Sensation, el cual popularizó el término. También se sugirió buscar más artículos a través de internet que, aunque no tocasen plenamente el termino, pudiesen ser útiles para la realización del trabajo.

2. **Fecha:** 21 de enero de 2021. **Modo:** online. **Programa:** Microsoft Teams.

   Contenido: En esta reunión se buscó la manera en la que se podía ejecutar el trabajo de una manera satisfactoria. Se concluyó con que se realizaría una demostración jugable junto con la memoria para mostrar con datos de testers como afecta el game feel.

   La demostración jugable sería basada en un juego del género de terror puesto que ya se habían realizado proyectos similares en el pasado y supondrían un interesante contraste con los conocimientos adquiridos gracias a la labor de investigación realizada sumado a los juegos de terror que ya hay en la industria del videojuego y que fueron jugados.

3. **Fecha:** 9 de julio de 2021. **Modo:** online. **Programa:** Microsoft Teams.

   Contenido: Con la demostración casi finalizada esta reunión trató sobre la resolución de dudas en la implementación de ciertas mecánicas. También se me dio a conocer más artículos que me podrían resultar útiles para la realización de la memoria.

   Se propusieron ideas que fueron desechadas en el futuro debido a incompatibilidades del motor (más concretamente al render pipeline elegido) como usar el Shader-Graph de Unity para crear elementos ambientales como gotas de agua cayendo por las paredes etcétera.

4. **Fecha:** 20 de agosto de 2021. **Modo:** online. **Programa:** Microsoft Teams.

   Contenido: Se revisaron ciertos aspectos de la memoria como la división de apartados y posible uso de los Anexos para mostrar los cuestionarios realizados por los testers en las sesiones que realizaron. Se recomendó el uso de otros documentos de Trabajo de Final de Grado que usaron también testeos con gente y cuestionarios.

   También se realizó un último vistazo a la demostración jugable, la cual estaba prácticamente finalizada. Se sugirieron ciertos cambios para facilitar el uso de esta en la presentación oral del Trabajo de Fin de Grado como la posibilidad de cambiar en runtime todas las opciones de pulido posibles o crear una capa extra de interfaz visual para representar de manera más evidente los cambios realizados.

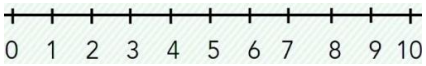
Anexo III - Cuestionarios

Primer cuestionario:

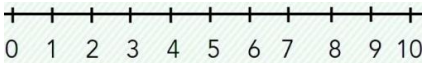
CUESTIONARIO DE PULIDO:

PRIMER PLAYTEST:

Sitúa en la siguiente recta cuanto has disfrutado del playtest:

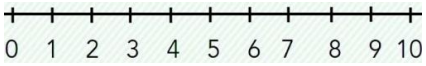


Sitúa en la siguiente recta cuan inmersiva crees que ha sido la experiencia:

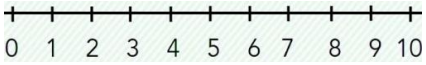


SEGUNDO PLAYTEST

Sitúa en la siguiente recta cuanto has disfrutado del playtest:

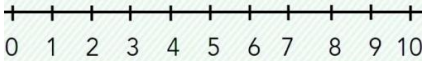


Sitúa en la siguiente recta cuan inmersiva crees que ha sido la experiencia:

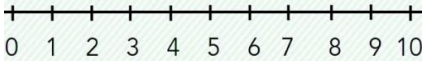


TERCER PLAYTEST

Sitúa en la siguiente recta cuanto has disfrutado del playtest:

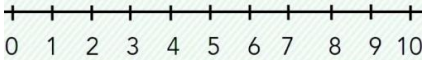


Sitúa en la siguiente recta cuan inmersiva crees que ha sido la experiencia:

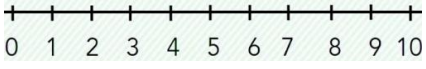


CUARTO PLAYTEST

Sitúa en la siguiente recta cuanto has disfrutado del playtest:

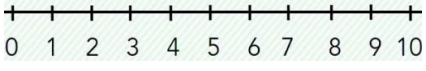


Sitúa en la siguiente recta cuan inmersiva crees que ha sido la experiencia:

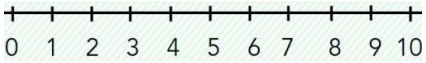


QUINTO PLAYTEST

Sitúa en la siguiente recta cuanto has disfrutado del playtest:



Sitúa en la siguiente recta cuan inmersiva crees que ha sido la experiencia:



CUESTIONARIO DE CONTROL EN TIEMPO REAL:

PIMER PLAYTEST

Del 1 al 10 cuan intensa ha sido la experiencia en términos acción:

1 2 3 4 5 6 7 8 9 10

Del 1 al 10 cuan intensa ha sido la experiencia en términos acción:

1 2 3 4 5 6 7 8 9 10

SEGUNDO PLAYTEST

Del 1 al 10 cuan intensa ha sido la experiencia en términos acción:

1 2 3 4 5 6 7 8 9 10

Del 1 al 10 cuan intensa ha sido la experiencia en términos acción:

1 2 3 4 5 6 7 8 9 10

TERCER PLAYTEST

Del 1 al 10 cuan intensa ha sido la experiencia en términos acción:

1 2 3 4 5 6 7 8 9 10

Del 1 al 10 cuan intensa ha sido la experiencia en términos acción:

1 2 3 4 5 6 7 8 9 10

Anexo IV – Figuras comparativas:

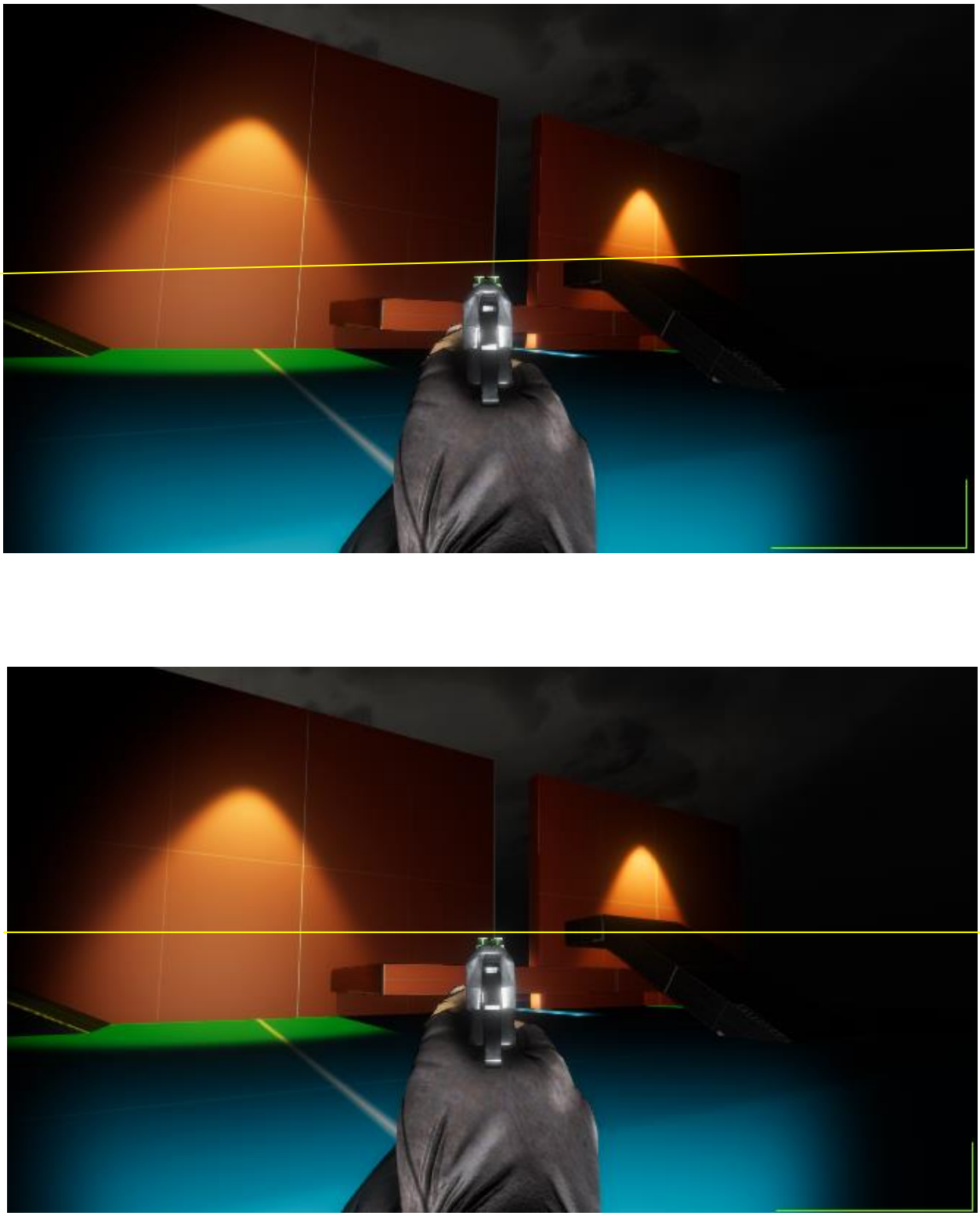


Figura 15: View Rolling vs No View Rolling (Versión final del proyecto)

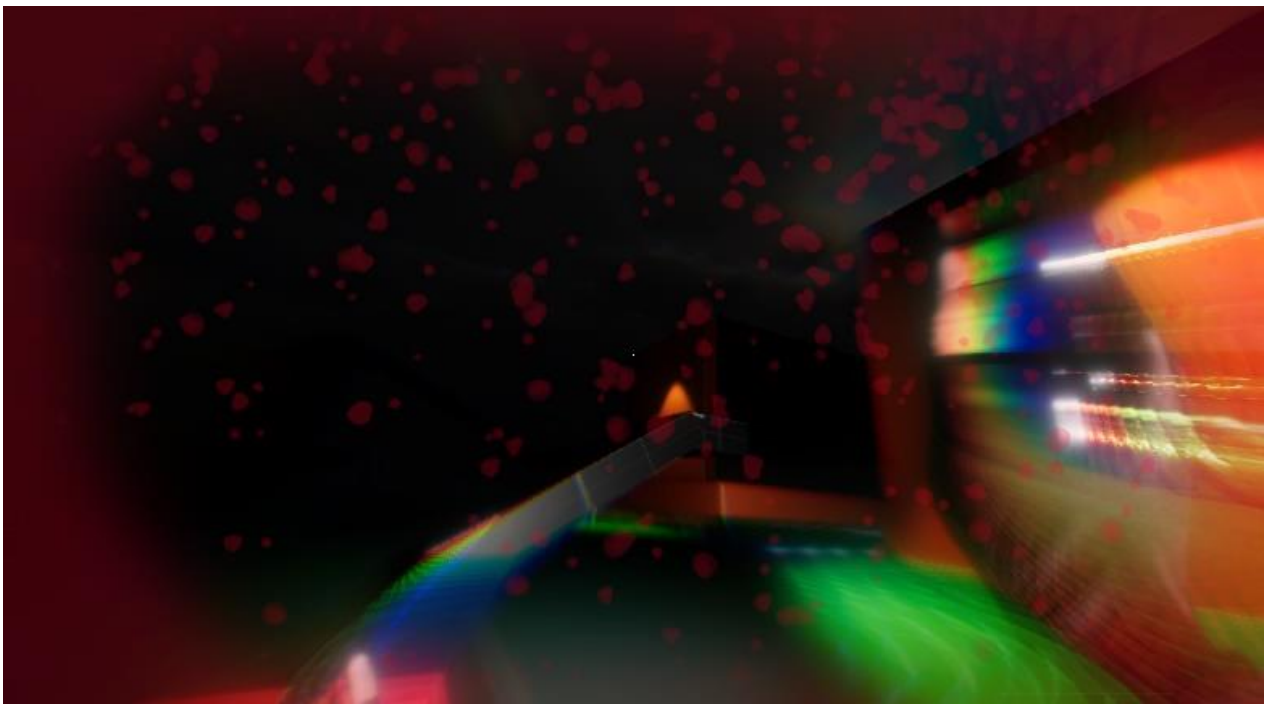


Figura 25: Indicador de daño de Outlast vs Indicador de daño de esta demo.

Figura 26: Diferencias entre apuntar con Ethan o Chris en Resident Evil 7.

9.Bibliografía

[1] SWINK, STEVE. `Game Feel: A Game Designer's Guide to Virtual Sensation' [En línea] Noviembre, 2008. Disponible en: https://gamifique.files.wordpress.com/2011/11/2-game-feel.pdf.

[2] SWINK, STEVE. `Game Feel: The Secret Ingredient´ [En línea] Noviembre, 2007. Disponible en: https://www.gamedeveloper.com/design/game-feel-the-secret-ingredient.

[3] A KIDWELL, BRANDOM. `A LOOK INTO GAME FEEL´ [En línea] Junio, 2019. Disponible en: kanashigd.com/kgdb/tag/game+feel.

[4] PERRY, DOUGH. `Super Mario 64 Review´ [En línea] Octubre 2020. Disponible en: https://www.ign.com/articles/1996/09/26/super-mario-64.

[5] NEWELL, ALLEN & CARD, STUART & MORAN, THOMAS. `The Psychology of Human-Computer Interaction´ [En línea] 1983. Disponible en: https://drive.google.com/file/d/13RR00DVv7GM987bMKDiq9oJxQpVZBu_i/view?usp=sharing.

[6] Instituto de Tecnología Carnegie Mellon. `How to Prototype a Game in Under 7 Days´ [En línea] Octubre, 2005. Disponible en: https://www.gamedeveloper.com/disciplines/how-to-prototype-a-game-in-under-7-days.

[7] Tadeusz Stach, T.C. Nicholas Graham, Matthew Brehmer, Andreas Hollatz. `Classifying Input for Active Games ´ [En línea] Octubre, 2009. Disponible en: https://mattbrehmer.ca/pubs/stach_ace09.pdf.

[8] MORRIS, CHRIS. 'Shigeru Miyamoto talks Nintendo's return to the movie world´ [En línea] Agosto, 2015. Disponible en: https://fortune.com/2015/08/21/nintendo-movie-partnerships/.

[9] SEEDHOUSE, ALEX. 'Yoshi's Woolly World Review´ [En línea] Junio, 2015. Disponible en: https://www.nintendo-insider.com/yoshis-woolly-world-review/.

[10] SAED, SHERIF. 'Battlefield 1 no HUD, colour-graded footage show a more immersive way to play´ [En línea] Noviembre, 2016. Disponible en: https://www.vg247.com/battlefield-1-no-hud-colour-graded-footage-show-a-more-immersive-way-to-play.

[11] YIN-POOLE, WESLEY. 'Call of Duty: Warzone gets perhaps its most significant weapon balance update yet´ [En línea] Abril, 2021. Disponible en: https://www.eurogamer.net/articles/2021-04-22-call-of-duty-warzone-gets-perhaps-its-most-significant-weapon-balance-update-yet.

[12] INNOVACIÓN EN FORMACIÓN PROFESIONAL. '¿Qué es Unity y para qué puedo utilizarlo?´ [En línea] Agosto, 2020. Disponible en: https://www.ifp.es/blog/que-es-unity-y-para-que-puedo-utilizarlo.

[13] UNITY TECHNOLOGIES. 'Animator Component'. [En línea] Publication 5.3-Q, 2016. Disponible en https://docs.unity3d.com/es/530/Manual/class-Animator.html.

[14] UNITY TECHNOLOGIES. 'Post-processing´ [En línea] Agosto, 2021. Disponible en: https://docs.unity3d.com/Manual/PostProcessingOverview.html.

[15] YOU TUBE 'TFG Lista de reproducción' [En línea] Septiembre, 2021. Disponible en: https://www.youtube.com/playlist?list=PLn2GpBv7aUDBGpKzbsgq1Y-UCHd7m3CdX.

[16] BERRAONDO, ALEJANDRO & LOSTAO, DANIEL & GARCÍA, ISMAEL & SOLANO, JORGE. 'Gameplay Silence' [En línea] Enero, 2021. Disponible en: https://www.youtube.com/watch?v=mPYR9YEG19s.

[17] BERRAONDO, ALEJANDRO & LOSTAO, DANIEL & GARCÍA, ISMAEL. 'Gameplay Secluded´ [En línea] Enero, 2021. Disponible en: https://www.youtube.com/watch?v=Lg7PdVN15xk.

[18] UNITY TECHNOLOGIES. 'Render pipelines introduction' [En línea] Agosto, 2021. Disponible en: https://docs.unity3d.com/Manual/render-pipelines-overview.html.

[19] UNITY TECHNOLOGIES. 'CharacterController´ [En línea] Agosto, 2021. Disponible en: https://docs.unity3d.com/ScriptReference/CharacterController.html.

[20] UNITY TECHNOLOGIES. 'Rigidbody´ [En línea] Abril, 2018. Disponible en: https://docs.unity3d.com/es/2018.4/Manual/class-Rigidbody.html.

[21] IRONEQUAL. 'Unity: CHARACTER CONTROLLER vs RIGIDBODY´ [En línea] Agosto, 2017. Disponible en: https://medium.com/ironequal/unity-character-controller-vs-rigidbody-a1e243591483.

[22] BERRAONDO, ALEJANDRO & LOSTAO, DANIEL & GARCÍA, ISMAEL. 'Gameplay Challenge Asylum´ [En línea] Junio, 2020. Disponible en: https://www.youtube.com/watch?v=RHoRnuUlML0.

[23] UNITY TECHNOLOGIES. 'Asset Store´ [En línea] Agosto, 2021. Disponible en: https://assetstore.unity.com/.

[24] ONYETT, CHARLES. 'Mirror's Edge Review' [En línea] Mayo, 2012. Disponible en: https://www.ign.com/articles/2009/01/13/mirrors-edge-review-2.

[25] GARCÍA, JUAN & SORIANO DAVID. 'Resident Evil 7 Análisis' Enero, 2017. Disponible en: https://es.ign.com/resident-evil-vii/113600/review/resident-evil-7-biohazard-analisis-para-ps4-xbox-one-y-pc.

[26] GUTIERREZ, SANDRA. 'Video games can cause motion sickness—here's how to fight it' [En línea] Julio, 2021. Disponible en: https://www.popsci.com/story/diy/how-to-fight-gaming-motion-sickness/.

[27] LJUNG,KENTH 'Effects of Field-of-View in First-Person Video Games A Study on Camera Field-of-View in Relation to Game Design' [En línea] Junio, 2015. Disponible en: http://www.diva-portal.org/smash/get/diva2:818863/FULLTEXT01.pdf.

[28] UNITY TECHNOLOGIES. 'Camera Stacking' Universal' [En línea] RP 7.2.1, 2020. Disponible en: https://docs.unity3d.com/Packages/com.unity.render-pipelines.universal@7.2/manual/camera-stacking.html.

[29] FANDOM OF AMNESIA, WIKIAMNESIA 'Sanity' [En línea] Disponible en: https://amnesia.fandom.com/wiki/Sanity.

[30] WILLIAMS MIKE. 'Resident Evil 7 Isn't Inspired By PT, But Capcom Did Consider a Supernatural Focus' [En línea] Julio, 2016. Disponible en: https://www.usgamer.net/articles/resident-evil-7-isnt-inspired-by-pt.

[31] UNITY TECHNOLOGIES. 'Unity pro' [En línea] 2021. Disponible en: https://store.unity.com/configure-plan/unity-pro?_gl=1*1ymdn8z*_gcl_aw*R0NMLjE2Mjk3NDE4MzUuQ2owS0NRandqbzJKQmhDUkFSSXNBRkc2NjdWOU5OR2pTekdRZ2RMR2JYVGtoZ2p3bURqS2JJdE5taUZTb1ZYcm5haVB2eElIRlFISXB5TWFBdWVjRUFMd193Y0I.*_ga*OTMyOTUyMTA0LjE2MjkyODY4ODI.*_ga_1S78EFL1W5*MTYyOTc0MTUxMi4zLjEuMTYyOTc0MTgzNC42MA..&_ga=2.49774950.110657303.1629741512-932952104.1629286882&_gac=1.118831483.1629741835.Cj0KCQjwjo2JBhCRARIsAFG667V9NNGjSzGQgdLGbXTkhgjwmDjKbItNmiFSoVXrnaiPvxIHFQHIpyMaAuecEALw_wcB.

[32] Blender 'Descarga y características' [En línea] Blender 2.93.3. Disponible en: https://www.blender.org/.

[33] Adobe Systems Incorporated PHOTOSHOP 'Descarga y precio' [En línea] Disponible en: https://www.adobe.com/es/creativecloud/plans.html?mv=search&mv=search&sdid=HCS3XL5Q&ef_id=Cj0KCQjwjo2JBhCRARIsAFG667V8iIh7K6AfHc2H3ZyUCHFjRUHsiJTs3oovjgrfiVuW8q9bg29lZXEaApM6EALw_wcB:G:s&s_kwcid=AL!3085!3!340807831155!e!!g!!precio%20de%20adobe%20photoshop!1445901510!56657233136&gclid=Cj0KCQjwjo2JBhCRARIsAFG667V8iIh7K6AfHc2H3ZyUCHFjRUHsiJTs3oovjgrfiVuW8q9bg29lZXEaApM6EALw_wcB.

[34] Microsoft 365 'Descargas y versiones' [En línea] Disponible en: https://www.microsoft.com/es-es/microsoft-365/business/compare-all-microsoft-365-business-products-b?&ef_id=Cj0KCQjwjo2JBhCRARIsAFG667W1hTJE2QzE19mgD-tF4MpBqpLxSCGJ72g-NiCgDygZJvGZbL6kEjgaAulWEALw_wcB:G:s&OCID=AID2200006_SEM_Cj0KCQjwjo2JBhCRARIsAFG667W1hTJE2QzE19mgD-tF4MpBqpLxSCGJ72g-NiCgDygZJvGZbL6kEjgaAulWEALw_wcB:G:s&lnkd=Google_O365SMB_Brand&gclid=Cj0KCQjwjo2JBhCRARIsAFG667W1hTJE2QzE19mgD-tF4MpBqpLxSCGJ72g-NiCgDygZJvGZbL6kEjgaAulWEALw_wcB.