UNIVERSIDAD DE CHILE
FACULTAD DE CIENCIAS FÍSICAS Y MATEMÁTICAS
DEPARTAMENTO DE CIENCIAS DE LA COMPUTACIÓN

# MECANISMOS DE INTERACCIÓN EN VIDEOJUEGOS CONTROLADOS POR EYETRACKING

MEMORIA PARA OPTAR AL TÍTULO DE
INGENIERO CIVIL EN COMPUTACIÓN

PATRICIO ANDRÉS ISBEJ CARRASCO

PROFESOR GUÍA:
FRANCISCO GUTIÉRREZ FIGUEROA

MIEMBROS DE LA COMISIÓN:
ÉRIC TANTER
PATRICIO INOSTROZA FAJARDIN

SANTIAGO DE CHILE
2020

# Resumen

El seguimiento ocular o *eyetracking* es una metodología cuyo uso todavía no se encuentra masificado en el mundo de los videojuegos, debido a que los usuarios deben incurrir en altos gastos en dispositivos extra. Esto está cambiando con la llegada de dispositivos de menor costo que permitirán su llegada a más hogares, tal como ocurrió para otros mecanismos de juego como lo son las pantallas táctiles y los sensores de movimiento. Más aún, los mecanismos de *eyetracking* en un futuro podrían venir integrados a computadores o consolas.

En este trabajo de título se busca explorar el uso de *eyetracking* en videojuegos, específicamente métodos alternativos para controlar la interacción con el usuario. Para esto, se decidió un estilo de juego y en base a éste se diseño y programó un nivel de videojuego en el que se permitió el control del personaje usando mecanismos tradicionales (un mando común y corriente), un *eyetracker* o una combinación de ambos. Además se desarrollaron métricas que el juego almacena sobre el rendimiento del jugador, que fueron daño recibido en el juego y tiempo que tomó el usuario en superarlo.

Para validar el juego desarrollado, se ejecutó una prueba de concepto con 24 usuarios, en donde debieron superar el nivel programado con y sin rastreo ocular. Los datos obtenidos de rendimiento se evaluaron junto con cuestionarios de experiencia de juego, para después comparar los resultados de las diferentes mezclas de controles. Los resultados muestran que, para jugar, los usuarios disfrutaron utilizar una combinación del mecanismo de seguimiento ocular con el mando tradicional, ya que les permitió mejorar en el rendimiento en el juego.

Con lo anterior se concluye que hay variantes en las que la gente prefiere utilizar *eyetracking* por sobre el mando tradicional para juegos de disparos 2D. Esta combinación corresponde a la combinación de controlar el movimiento del personaje del juego por medio del mando tradicional, mientras que apuntar se realiza observando al objetivo al que se desea disparar. Estos resultados se pueden tomar en consideración por desarrolladores de juegos, ya que les podría permitir crear juegos más llamativos y agradables para los jugadores.

Este trabajo presenta una base en la que se puede seguir explorando con mayor profundidad el uso de seguimiento ocular en videojuegos de ese género. Así pues, como trabajo futuro se propone realizar el mismo experimento con una audiencia mayor, como también el aplicar mecanismos similares a géneros de juego no estudiados anteriormente.

# Agradecimientos

A mi profesor guía por todo su apoyo en este proceso, a Francisca, Sofía, Matilde, Elisa, mi familia y amigos quienes me motivaron a continuar escribiendo. Y a quienes me recibieron alguna vez, en especial el depa y Gabriel por dar alojo y compañía todos estos años.

# Tabla de Contenido

# Índice de Tablas

# Índice de Figuras

# Capítulo 1

# Introducción

El seguimiento de mirada (en inglés, *eyetracking*) es el proceso de medir, por diferentes medios, la posición y movimiento de los ojos para determinar el punto que observa un sujeto [16]. Esta técnica es utilizada en el estudio del comportamiento de la mirada al realizar ciertas tareas, como la forma en que un usuario percibe una página o imagen/video, determinando dónde centra el usuario su atención [5]. También puede aplicarse como método de interacción entre un usuario y un software para mejorar la experiencia de uso; en particular, este trabajo se enfoca en su uso en los videojuegos [12].

Su uso se ha masificado en este área, en parte, gracias al reciente desarrollo de productos de seguimiento de ojos para uso comercial de menor costo y tamaño, permitiendo un alcance mayor de jugadores. Dentro de estos productos, Tobii[1] es una de las compañías más importantes en el área, desarrollando uno de los seguidores de mirada más populares tanto para pantallas normales como para lentes de realidad virtual. Otro de los factores que vuelve importante a esta compañía es su soporte al desarrollo de videojuegos, ya que sus productos ofrecen un kit de desarrollo para dos de los motores de videojuegos más utilizados: Unity[2] y Unreal Engine[3]. Gracias a este kit se han desarrollado más de 100 juegos, siendo muchos de éstos de alto presupuesto [2].

## 1.1. Problema abordado

Si bien existen muchos juegos que utilizan seguimiento de mirada, una revisión reciente de la literatura en el área [15] plantea que muchos de los estudios realizados no cuentan con sustento de validación empírica rigurosa. Esto se traduce en que el conocimiento nuevo, útil y reusable generado, no necesariamente se ve reflejado en pautas o guías específicas para apoyar el diseño y desarrollo de nuevas aplicaciones en este dominio.

Algunos de los estudios reportados revelan que el rendimiento de un jugador al utilizar

[1] https://www.tobii.com
[2] https://www.unity.com
[3] https://www.unrealengine.com

un seguidor de mirada no refleja necesariamente una mejora contra métodos de entrada convencionales (como utilizar un mando o *mouse* y teclado), pero sí se revela una mayor satisfacción al utilizarlos [3, 10, 11]. Esto último podría deberse a un factor de novedad al controlar el juego usando la mirada.

Como ha ocurrido con otras tecnologías, se estima que el uso de *eyetracking* se masificará con la llegada de hardware de mejor calidad y más barato. Esto ya ha ocurrido, por ejemplo, con pantallas táctiles y sensores de movimiento que hoy se usan masivamente en los juegos de teléfonos celulares y consolas. Un estudio utilizó la cámara de celular del usuario como método de entrada cuando éste guiñaba [9], demostrando que los teléfonos actuales tienen la capacidad de procesamiento y hardware suficiente para realizar acciones de seguimiento de mirada. Luego, resulta necesario estudiar en cuáles interacciones sirve el seguimiento de mirada y cuáles no.

## 1.2. Solución desarrollada

En este trabajo se diseñó e implementó un videojuego del género de disparos 2D de vista superior. En este juego se permitió utilizar la mirada con distintas aplicaciones interactivas, las cuales son mover al personaje, apuntar junto con disparar, y ambas mecánicas simultáneamente. Para el desarrollo de este juego se utilizó el motor de juegos Unity junto con el dispositivo de *eyetracking* Tobii 4C. Para determinar la aplicación del rastreo ocular en el videojuego se definieron métricas e implementó un cuestionario de experiencia de juegos, el cual fue aplicado a 24 usuarios. Posterior a las pruebas se recopiló y estudió los resultados, determinando que una combinación de uso de *eyetracking* con mando tradicional fue el preferido por los usuarios. En la figura 1.1 se puede observar el juego desarrollado.

## 1.3. Objetivos

A continuación, se presenta el objetivo general y los objetivos específicos a cumplir en este trabajo de título.

### Objetivo General

Diseñar y desarrollar un videojuego en el cual se incorporen mecánicas de seguimiento de mirada, con el fin de medir comparativamente el desempeño y grado de satisfacción de los usuarios en distintas configuraciones de juego.

Figura 1.1: Juego desarrollado para este trabajo.

### Objetivos Específicos

Para lograr el objetivo general expuesto, se propone cumplir con los siguientes objetivos específicos:

1. Definir mecanismos de interacción mediados por *eyetracking*, abarcando distintos tipos de configuración de su uso, ya sea desde usar únicamente un mando tradicional de juego, pasando por una mezcla híbrida de mando junto con seguimiento ocular, hasta utilizar únicamente *eyetracking* para todas las interacciones.

2. Desarrollar un videojuego básico controlable por *eyetracking*, de tal manera que permita adaptar sus controles en función de las configuraciones definidas.

3. Generar métricas de uso de manera automática que permitan describir el rendimiento y experiencia del usuario durante la sesión de juego.

## 1.4. Metodología

A continuación, se listan los pasos seguidos durante el desarrollo del videojuego, para así cumplir con los objetivos propuestos en el presente trabajo de título:

1. Se analizó y comparó diversos equipos comerciales de seguimiento de mirada para encontrar el mejor balance entre accesibilidad y precisión, optando finalmente por el más

barato, ya que las especificaciones ofrecidas cumplen con los requisitos para el estudio de su uso en videojuegos. Dicho *eyetracker* incluye un kit de desarrollo para los motores de juegos más utilizados actualmente. Además de esto, su precio lo convierte en el candidato más apto para el uso masivo.

2. Se llevó a cabo una búsqueda de artículos en el estado del arte para estudiar de forma detallada el uso actual de *eyetracking* en videojuegos, determinando, en primer lugar, las áreas en las que existe documentación para evitar repetición de trabajos y, en segundo lugar, qué problemas recurrentes en el campo de investigación se deben tener en consideración al diseñar e implementar el videojuego.

3. Se diseñó un videojuego teniendo como prioridad la implementación de mecánicas de juego que pudieran ser fácilmente adaptadas a controles basados en la mirada. Para esto, se puso especial énfasis en refinar los parámetros que pudieran afectar la precisión del *eyetracker*, como la separación de los enemigos, eliminación de elementos distractores, tamaño de los objetos en la pantalla, etc. Todo esto, evitando los géneros y estilos de juego previamente abordados en otros estudios de *eyetracking*.

4. Para el desarrollo del juego, se estudiaron y evaluaron los dos motores más populares con soporte del *eyetracker* utilizado, Unity y Unreal Engine. Se decidió optar por el primero, el cual cuenta con el mejor soporte para juegos 2D, además de que el kit de desarrollo de Tobii está mejor documentado y avanzado en ese motor.

5. Se implementó el videojuego diseñado previamente, asegurándose que todas las variantes puedan ser utilizadas de manera sencilla, para esto se realizó una prueba de concepto donde se obtuvo comentarios y sugerencias para mejorar la experiencia. También se desarrolló en el juego la recolección automática de distintas métricas para poder comparar resultados entre experiencias distintas.

6. Se probó el juego con 24 usuarios, quienes debieron completar la etapa desarrollada, una vez con un mando tradicional, y otra variante utilizando *eyetracking* para controlar el movimiento y/o disparo del personaje. Una vez completado el nivel de las 2 maneras distintas, cada usuario respondió un cuestionario de experiencia de juego, para obtener su apreciación en cada variante jugada y comparar posteriormente sus resultados.

7. Se recopiló los datos obtenidos de las pruebas a usuarios realizadas y se analizó, comparándolos entre ellos para encontrar tendencias como método de entrada preferido o el con mejor rendimiento, para concluir posteriormente si es factible utilizar mecanismos de *eyetracking* en juegos del género desarrollado.

## 1.5. Estructura del documento

El resto de este documento se estructura como sigue. En el capítulo 2 se presenta y discute la literatura relacionada que sustenta el trabajo en esta memoria de título. El capítulo 3 está dedicado a presentar la selección del dispositivo elegido para seguimiento de mirada, además de el diseño de los elementos del juego diseñado. En el capítulo 4 se expone el desarrollo del juego y las mecánicas a usar con *eyetracking*. Posteriormente, en el capítulo 5 se presentan los resultados de una prueba de concepto para verificar el cumplimiento de los objetivos propuestos al inicio de este trabajo. Finalmente, en el capítulo 6 se presentan las conclusiones y se ofrecen perspectivas de trabajo futuro.

# Capítulo 2

# Trabajo Relacionado

Son 6 músculos los responsables de realizar el movimiento de cada ojo, los que les dan una libertad de movimiento en 6 ejes; 3 de rotación, y 3 de traslación [4]. Los tipos de movimiento que puede realizar una persona con sus ojos son 4 y se describen a continuación [13]:

- **Movimientos sacádicos:** Movimientos rápidos para reposicionar la fóvea.
- **Movimientos de seguimiento:** Movimientos fluidos para seguir un objetivo en movimiento.
- **Movimientos de convergencia:** Pequeños movimientos para ajustar el enfoque de ambos ojos.
- **Reflejo vestíbulo-ocular:** Compensa al movimiento de la cabeza de la persona.
- **Reflejo optocinético** Movimientos que se realizan cuando un objeto observado activamente desaparece del campo de visión.

La visión humana tiene 2 componentes diferentes.

- **Percepción:** Se realiza un "barrido" del entorno para obtener información.
- **Cognición:** Involucra procesos como pensamiento, memoria y razonamiento.

Un problema que surge en el uso de *eyetracking* es que el componente de percepción empuja a una persona realizar movimientos oculares involuntarios para obtener información de los alrededores, por lo que, al utilizarse como método de entrada, existe la posibilidad de gatillar acciones involuntarias al ver el entorno. A esto se le conoce como *Midas Touch* [7] y es un detalle que debe tenerse en consideración si se quiere trabajar con *eyetracking*. Una de las soluciones conocidas a este problema es la implementación de una entrada extra para confirmar una acción, como un botón o gesto con la vista. También se puede definir un pequeño tiempo de espera con la mirada fija para asegurar que se está realizando una acción y no sólo observando el entorno. Al respecto, Velichkovsky et al. [14] propusieron un método en el que se diferencia *ambient fixation* y *focal fixation*, siendo lo primero la exploración inicial de la imagen, en la que se realizan recorridos rápidos con los ojos usando principalmente la vista periférica, y lo segundo una mirada fija en un punto específico donde se realizan menos movimientos oculares.

Velloso y Carter [15], en un mapeo sistemático reciente de la literatura en el área de videojuegos e interacción mediada por la mirada, (1) clasificaron los tipos de movimientos de ojos, (2) identificaron los tipos de entrada que capturan los seguidores de mirada, y (3) propusieron una taxonomía de las mecánicas de juego que se pueden utilizar en este dominio de aplicación. En particular, las mecánicas propuestas son: navegación, apuntar y disparar, selección y comandos, interacciones implícitas, y efectos visuales. Dado lo anterior, el estudio realizado por Velloso y Carter sienta las bases para futuros trabajos de investigación en el área.

De las mecánicas de juego mencionadas en el mapeo anterior, aquellas de interés para este trabajo son *navegación* junto con *apuntar y disparar*. En particular, estas mecánicas definen formas en que se pueden controlar las acciones de un personaje utilizando la mirada. Por otro lado, otra mecánica de interés son las *interacciones implícitas*, las que no controlan a un personaje directamente, sino que adaptan el juego dependiendo de dónde se encuentra observando el usuario en ciertos momentos. Por ejemplo, se puede hacer que un personaje salude al jugador cuando este lo mira a los ojos.

Complementando lo anterior, Navarro y Sundstedt [11] utilizaron interacciones implícitas en un juego del género *Shoot'em up*, en donde el jugador controla una nave y debe disparar a asteroides y naves enemigas. La nave del jugador tiene dos modos, ofensivo y defensivo. Cuando el jugador observa su propia nave se asume que está intentando esquivar, por lo que el juego cambiaba el estado de la nave a defensivo, mientras que al mirar a los enemigos se asumía que intentaba atacar, cambiando a ofensivo. Un trabajo futuro que proponía este estudio es utilizar interacciones implícitas en otros géneros donde este mecanismo aún no haya sido abordado.

Dechant et al. [3], realizaron una evaluación empírica de técnicas para seleccionar en un juego en primera persona, midiendo el tiempo que se demora un jugador en acertarle a blancos presentados en la pantalla. Para esto evaluaron el rendimiento de esta tarea con cinco métodos distintos, tres de estos utilizando *eyetracking* (ya sea de manera única o de forma híbrida con mando tradicional), el cuarto siendo el uso del mouse, y el quinto utilizar un mando. Las conclusiones de ese trabajo fueron que el mouse logró obtener mejores resultados en rendimiento, mientras que utilizar únicamente *eyetracking* fue el peor. También se encontró que *eyetracking* combinado con mandos tradicionales de juego fue el método preferido por lo usuarios y que este superó en rendimiento a utilizar solamente mando en ciertos casos.

Con lo anterior se destaca que quedan aún muchos tipos de generos de juego y de interacciones con *eyetracking* que no se han explorado, y de haberse realizado, su uso no ha sido registrado en la literatura. Esto significa que existen todavía alternativas por estudiar, de las cuales se seleccionará una para estudiar en este trabajo.

# Capítulo 3

# Diseño del Juego

En este capítulo se describen los elementos principales que guiaron el diseño del videojuego desarrollado en este trabajo de título.

## 3.1. Elección de mecanismo de *eyetracking*

Previo al diseño del videojuego se decidió el *eyetracker* que se utilizaría durante el transcurso de este trabajo. Existen diversos métodos para determinar en dónde se encuentra observando un sujeto, entre éstos se destacan los siguientes dos mécanismos [4]:

- **Cámara de video:** Es posible utilizar una cámara de video y procesar la posición de la pupila respecto al ojo para saber en cada momento la dirección de la mirada. De este mecanismo existen dos variantes que son:
  - **Cámara fijada en rostro:** Se fija la cámara a la cabeza para permitir el libre movimiento del sujeto. Este método requiere de una cámara adicional que grabe de frente al usuario para poder conocer lo que observa.
  - **Cámara fijada en mesa:** Se fija la cámara en un lugar cercano a lo que se quiera estudiar, pudiendo ser esto una imagen o pantalla. Para que esta configuración funcione se debe asegurar que el usuario no se desplace mucho del lugar.
- **Medidor de potencial eléctrico:** Esto mide la acción de cada músculo responsable del movimiento de los ojos, por lo que permite una precisión muy alta. Al igual que la cámara fijada en el rostro, este mecanismo requiere de otra cámara que grabe hacia dónde observa el usuario o que éste deje su cabeza estática al momento de observar.

Como en este trabajo de título se busca el desarrollo de un videojuego, un factor importante a tomar en cuenta es la accesibilidad y comodidad de todo el material necesario para poder jugar. Por lo anterior se descartó el uso de medidor de potencial eléctrico por tener un

precio elevado y requerir de muchos pasos previos e incómodos para comenzar a utilizarse, y se consideró las siguientes alternativas al momento de decidir el mecanismo de *eyetracking* con el cual trabajar:

- **Programa gratuito[1]:** Existe una variedad de software gratuito que permite utilizar cualquier webcam para realizar seguimiento ocular. Es la alternativa más barata pero la precisión que entregan es aún muy baja.

- **Tobii 4C[2]:** Corresponde a una cámara fija en la mesa, cuyo precio ronda los 200 dólares. Este dispositivo incluye un kit de desarrollo para los motores de juegos Unity y Unreal Engine.

- **Gazepoint GP3[3]:** Cámara fija en la mesa con una precisión un poco mayor que Tobii. Tiene kit de desarrollo incluido y su precio es de aproximadamente 700 dólares.

- **Pupil Core[4]:** Cámara fija en el rostro. En apariencia es similar a un marco de lentes donde se tiene una cámara que enfoca hacia los ojos del usuario y otra apuntando al frente para grabar el entorno. Provee una API para facilitar su uso, su precio es aproximadamente 2000 dólares.

La tabla 3.1 resume las diferentes características de cada mecanismo estudiado para este trabajo.

| Dispositivo | Tipo | Precio | Precisión | API |
|---|---|---|---|---|
| Software gratuito | Cámara fijada en mesa | Gratis | Mala | Poco soporte |
| Tobii 4C | Cámara fijada en mesa | $200 | Media | API y SDK juegos |
| Gazepoint GP3 | Cámara fijada en mesa | $700 | Buena | API |
| Pupil Core | Cámara fijada en rostro | $2000 | Buena | API |

Tabla 3.1: Resumen análisis de dispositivos de seguimiento ocular

El dispositivo seleccionado para utilizar durante este trabajo fue Tobii 4C ya que es el hardware más barato encontrado y tiene una precisión lo suficientemente alta como para ser utilizado en juegos. Además, este dispositivo fue creado con el propósito de ser una herramienta para videojuegos, por lo que no sólo proporciona un kit de desarrollo sino que además provee un kit para ser usado de manera fácil en motores de videojuegos.

Tobii 4C viene con un software que facilita su instalación y uso. Al respecto, las figuras 3.1 y 3.2 muestran el proceso a seguir para calibrar el Tobii 4C cuando va a ser utilizado por un usuario nuevo. Además, la compañía a cargo de este dispositivo es la que más se encuentra involucrada en el desarrollo de videojuegos utilizando seguimiento ocular, otorgando una

[1]https://imotions.com/blog/free-eye-tracking-software/
[2]https://gaming.tobii.com/tobii-eye-tracker-4c/
[3]https://www.gazept.com/product/gazepoint-gp3-eye-tracker/
[4]https://pupil-labs.com/products/core/

serie de material de aprendizaje y documentación para el uso del *eyetracker* en los motores de juego Unity y Unreal Engine[5].


Figura 3.1: Proceso calibración Tobii 4C: instrucciones y punto central

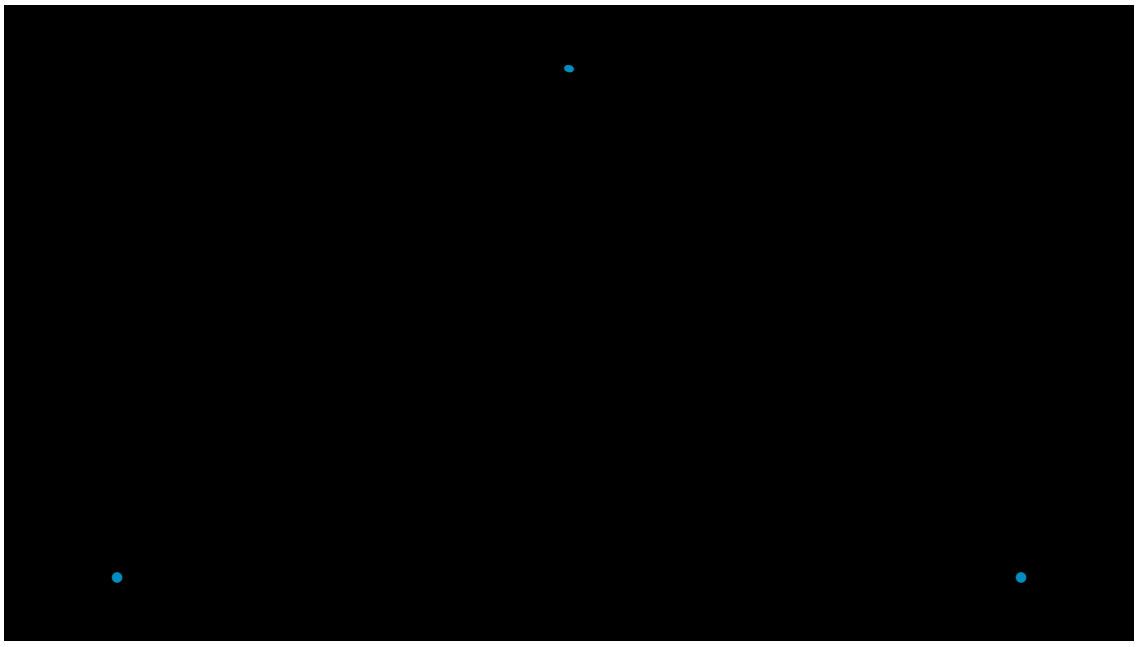

Figura 3.2: Proceso calibración Tobii 4C: puntos para reconocer los bordes del monitor

[5]https://gaming.tobii.com/developer/

## 3.2. Diseño

Un problema identificado es que el *eyetracker* que se va a utilizar en este trabajo de título tiene una precisión de, aproximadamente, una "uña de pulgar" en la pantalla [1], como se puede apreciar en la figura 3.3. Esto significa que si dos objetos son lo suficientemente pequeños y se encuentran próximos entre sí, al querer interactuar con uno de éstos se podría interpretar como una interacción con el otro. Para evitar esta situación se debe tener en cuenta que los objetos a interactuar no pueden ser muy pequeños y se deben encontrar apartados de los otros. Por ejemplo si se desea implementar un menú que utilice *eyetracking*, cada elemento de este debería ser grande y con una separación adecuada.

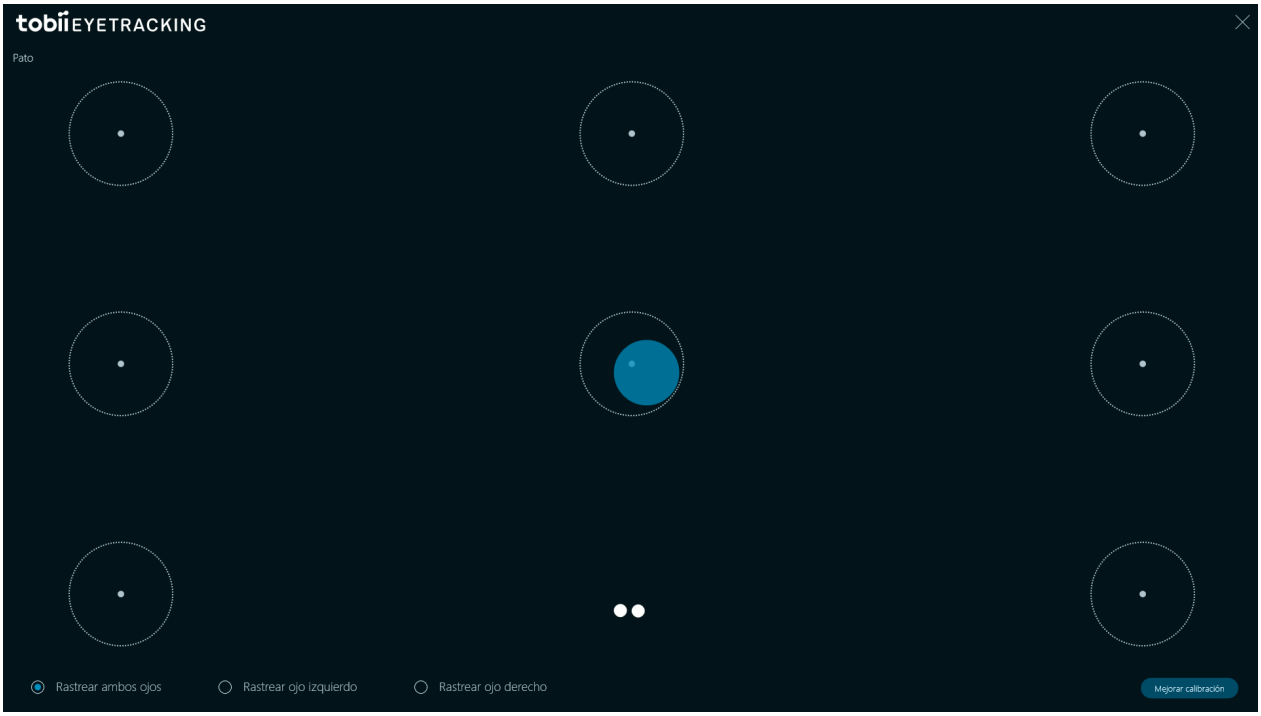


Figura 3.3: Precisión de eyetracker en monitor de 27 pulgadas, el circulo dentro de la circunferencia central muestra el área que percibe el *eyetracker* dónde se encuentra observando el usuario

El juego que se decidió diseñar para este trabajo es un juego de disparos en dos dimensiones (2D) con vista superior en donde el personaje podría moverse en dos ejes y al mismo tiempo apuntar y disparar a enemigos. Por ejemplo, un juego que sirve de inspiración para la definición de las mecánicas de interacción propuestas es *Zombs Royale*[6], cuya interfaz principal de acción se muestra en la Figura 3.4.

En el juego, el usuario debe superar un nivel con diferentes zonas en las que se debe ir eliminando los enemigos que se encuentren antes de poder proseguir. Para facilitar el uso de la vista y evitar al disparar que ocurra el problema de precisión del *eyetracker* mencionado en la parte anterior, se usarían enemigos estáticos separados unos de otros. Al respecto, un bosquejo que muestra la idea del juego a desarrollar en este trabajo de título se muestra en

[6] `https://zombsroyale.io/`

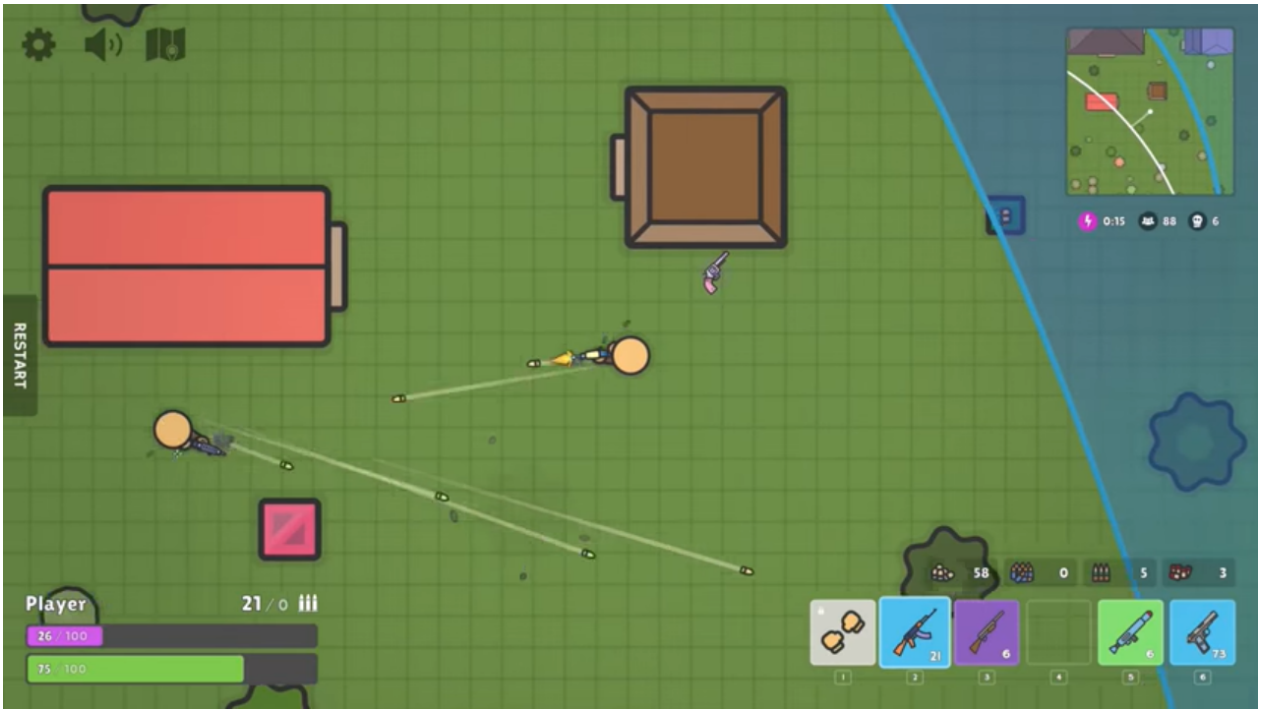


Figura 3.4: Juego de ejemplo: Zombs Royale

la Figura 3.5. Como se puede observar, se decidió representar a los enemigos como torres ya que son lo suficientemente grandes para poder ser observadas fácilmente por el jugador y son estructuras que no se mueven, evitando que se puedan encontrar 2 de ellas próximas entre ellas.

La manera en la que se pretende utilizar *eyetracking* para este trabajo es implementar distintas formas de controlar al personaje. Así pues, se podrá controlar su movimiento con mando de juegos o el *eyetracker*, y lo mismo para apuntar y disparar. Ambas mecánicas se podrán configurar por separado, con lo que se obtienen cuatro configuraciones distintas, estas configuraciones se pueden dividir en 3 grupos diferentes, como se muestra a continuación:

- **Sólo mando tradicional:** En esta configuración, tanto el movimiento como el disparar se hacen mediante el uso de un mando, sin la necesidad del *eyetracker*. Este modo de juego se utilizó para tener una medida base con la cual se compararon resultados con las otras configuraciones que si tienen mecánicas que usan la mirada.
- **Híbridos:** Consisten en dos configuraciones diferentes que utilizan tanto el mando de juego como el *eyetracker*. La primera es controlar el movimiento del personaje con el mando y disparar usando la vista, mientras que la segunda configuración es utilizar la mirada para mover el personaje y disparar utilizando el mando.
- **Sólo *eyetracker*:** Con esta configuración no se jugaría con el mando tradicional y se utilizaría únicamente la mirada para controlar el movimiento del personaje, apuntar y disparar. Para esta configuración, el jugador podrá fijar el blanco al que se encuentra disparando para así poder facilitar el moverse con disparar simultáneamente.

En la figura 3.6 se presentan las configuraciones posibles que se obtienen al combinar los

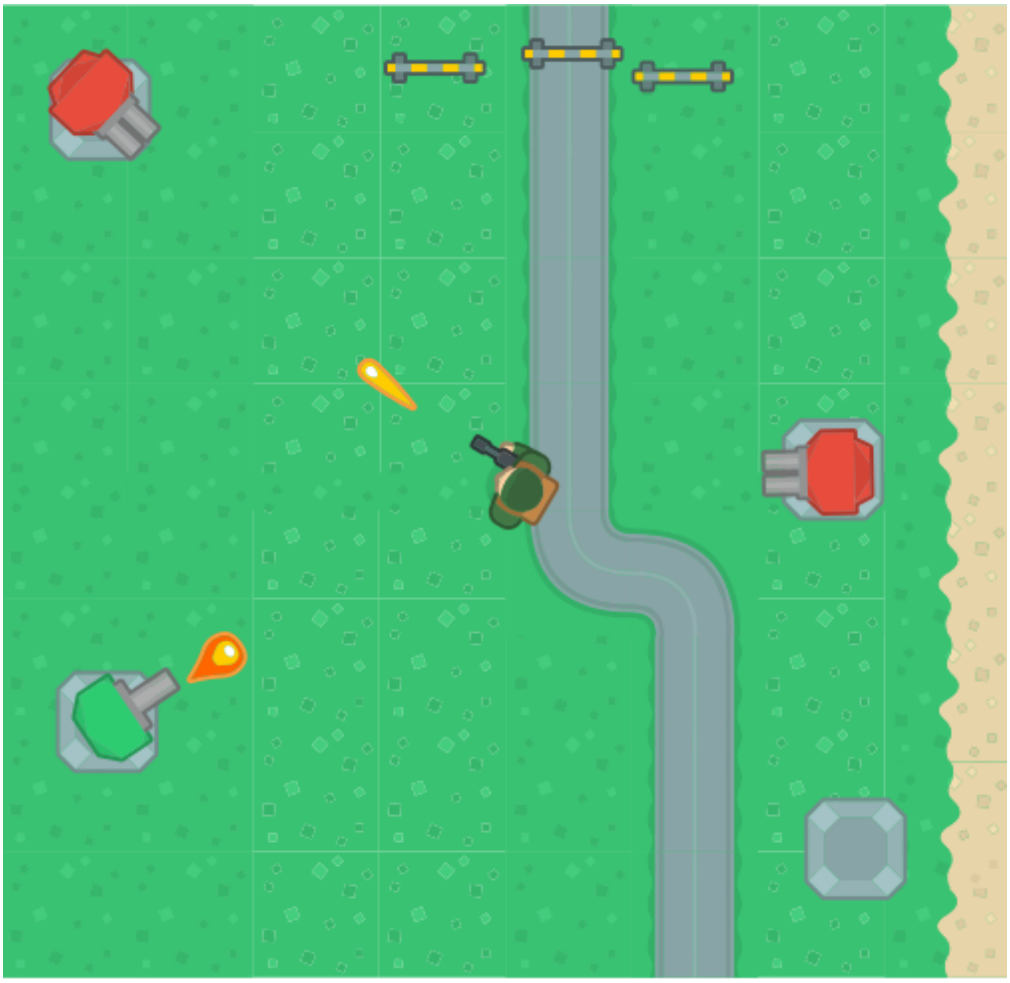

Figura 3.5: Bosquejo de juego propuesto

distintos métodos de entrada para cada mecánica.

| | | Apuntar/Disparar | |
|---|---|---|---|
| | | **Control** | ***Eyetracker*** |
| Movimiento | **Control** | Sólo Control | Híbrido |
| | ***Eyetracker*** | Híbrido | Sólo *eyetracker* |

Figura 3.6: Combinaciones de mecánicas

Aplicando lo aprendido en la revisión de literatura relacionada, para evitar el problema de *Midas touch* al disparar, se fijará un pequeño umbral de tiempo en el que el jugador debe dejar de mover la vista para poder apuntar al lugar observado. De lo contrario el personaje dispararía erráticamente cuando el jugador se encuentre observando los alrededores de la etapa.

### 3.2.1. Mecánicas del juego

Como se describió anteriormente, las 2 mecánicas principales que tiene el juego diseñado son:

- **Movimiento:** Para la variante sin *eyetracker* el movimiento se realizará utilizando el *stick* izquierdo del mando, mientras que en la variante con *eyetracker*, el personaje se moverá automáticamente al punto de la pantalla en donde se encuentre observando el usuario. Para facilitar el movimiento durante todo el juego se configurará la cámara de modo que siga al personaje en todo momento dejándolo siempre al centro de la pantalla.

- **Apuntar y disparar:** Para la variante sin *eyetracker*, se apuntará usando el *stick* derecho del mando y se disparará con el gatillo derecho. Para la variante con rastreo ocular el personaje apuntará donde se encuentre observando el usuario hasta que su vista se encuentre con un enemigo por un tiempo determinado, en cuyo caso el personaje comenzará a apuntar hasta que la vista se despegue del enemigo en caso de realizar el movimiento con mando de juegos, si el movimiento se realiza con la vista el personaje no dejará de apuntar al enemigo para facilitar el acto de esquivar las balas enemigas.

## 3.3. Etapa

El juego consiste en una etapa dividida en 4 zonas distintas, el paso entre cada parte se encontrará bloqueado por unos obstaculos que impedirán el paso, para eliminar los obstaculos y avanzar de una zona a la siguiente se debe eliminar a todos los enemigos que se encuentren ahí.

Las zonas se diseñaron para poner a prueba las habilidades de los jugadores con las mecánicas de moverse y apuntar. Estas son:

### 3.3.1. Zona para aprender

En esta zona el jugador no se encontrará con enemigos y tendrá un espacio reducido en el que se pueda mover, esto con el propósito de que tenga tiempo para aprender y acostumbrarse a los controles, sobre todo en los casos donde se utilice la mirada.

### 3.3.2. Zona amplia

En esta zona no se tendrá obstáculos y se contará con pocos enemigos, para acostumbrar al jugador a apuntar a las torres mientras esquiva los proyectiles que le sean disparados.

### 3.3.3. Zona angosta

Esta zona corresponde a un pasillo, mucho más angosto que la parte anterior y con más enemigos también. Aquí el jugador deberá esquivar de manera más hábil los proyectiles ya que sus movimientos se verán reducidos por la cercanía a las paredes.

### 3.3.4. Zona final

Esta es la última zona de la etapa diseñada y aquí se pone a prueba de forma más difícil las habilidades del jugador, con más obstáculos y enemigos más resistentes que en las zonas previamente mencionadas.

Un bosquejo de la etapa diseñada con sus distintas zonas se puede observar en la figura 3.7. Cada zona se representa como un rectángulo y los obstaculos que previene el paso entre zonas se representa como una linea más gruesa entre los rectángulos.

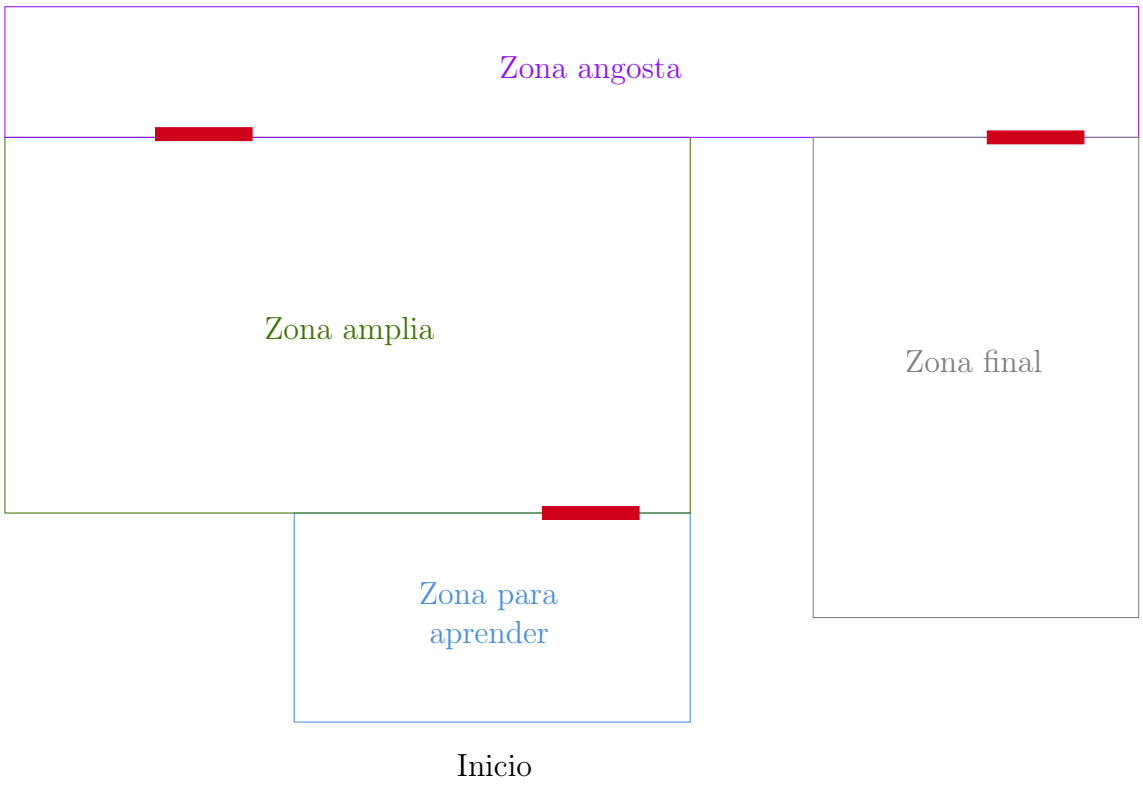


Figura 3.7: Bosquejo de la etapa diseñada con cada zona representada.

## 3.4. Assets

Para el audio y gráficos del juego se decidió utilizar los del sitio `https://www.kenney.nl/`, grupo dedicado a generar estos materiales para el dominio público. Los materiales de esta

página se encuentran creados bajo la licencia CC0[7], lo que permite su uso libre para todo tipo de proyectos. Entre estos se pueden encontrar gráficos de juegos de disparo de vista superior, como se puede observar en la figura 3.8


Figura 3.8: Assets del sitio `https://www.kenney.nl/`

[7] `https://creativecommons.org/publicdomain/zero/1.0/`

# Capítulo 4

# Implementación del Juego

En este capítulo se describen las principales decisiones de ingeniería y aspectos técnicos del desarrollo del videojuego implementado en esta memoria.

## 4.1. Elección motor

En primer lugar, se debe decidir respecto a qué motor de desarrollo de videojuegos utilizar para implementar la solución diseñada.

### 4.1.1. Pruebas de motores de desarrollo

Como existen dos motores de juegos que cuentan con un SDK (kit de desarrollo) provisto por Tobii, se debió decidir cuál de éstos se iba a utilizar durante el transcurso del desarrollo del trabajo de título. Para poder decidir esto, se especificaron diversos parámetros que resultan importantes para el desarrollo del juego a realizar. Luego se comparó cada motor respecto a cada parámetro, los cuales son los siguientes:

#### Precio

Un factor importante a considerar es el precio de crear juegos en el motor, ya que se desea evitar incurrir en gastos muy elevados. Se determina que ambos motores poseen versiones gratuitas si el juego realizado no es utilizado con fines comerciales.

### Soporte del kit de desarrollo de Tobii

Con el objetivo de saber qué motor de juego se encuentra más apto para trabajar con Tobii, se importó en cada uno su kit de desarrollo correspondiente[1] y se cargaron las demostraciones para conocer ejemplos de cómo utilizar el *eyetracker* en cada uno. Asimismo, se buscó además la documentación de la API de Tobii para saber qué información se puede obtener de la mirada. Al respecto, la figura 4.1 muestra uno de los tutoriales provistos por Tobii. De lo anterior se observa que la documentación de la API se encuentra únicamente para Unity, además de que el kit de Unreal Engine todavía está en beta. Asimismo, se identificó que la información que se puede obtener del *eyetracker* es la posición de la mirada del usuario en la pantalla, la inclinación de su cabeza y si se encuentra presente o no.

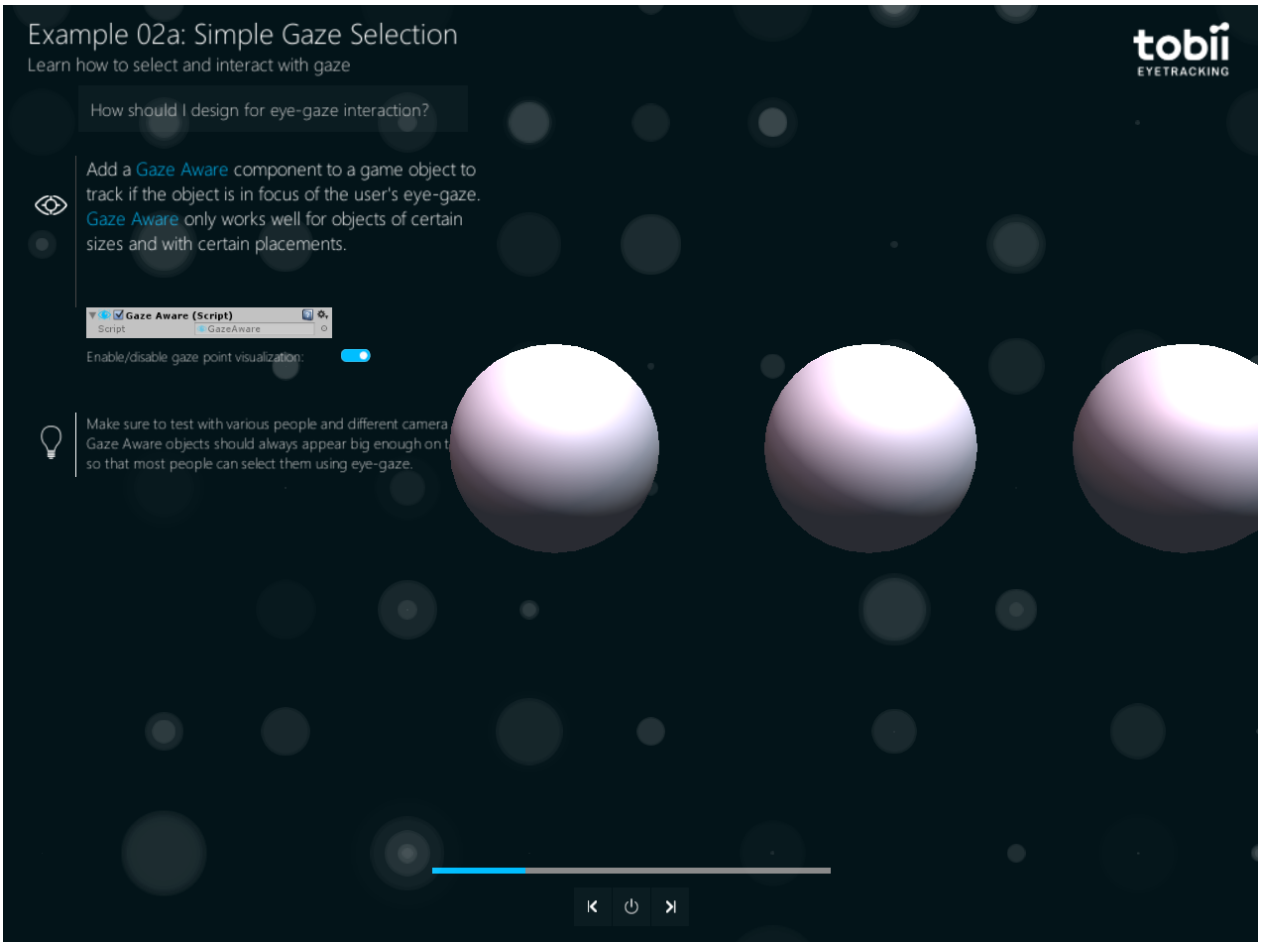


Figura 4.1: Demo de uso de *eyetracker*

### Soporte del motor para realizar juegos 2D

Como el juego propuesto es de tipo 2D, se desea saber cuál de los dos motores permite un desarrollo más expedito de juegos de este estilo. Para esto, se buscó material de referencia al respecto para cada uno. Además, se leyó la documentación sobre este tema para cada motor.

[1] `https://developer.tobii.com/pc-gaming/develop/`

Con esto se observó que existe una mayor cantidad de material sobre desarrollo de juegos 2D para Unity. Otro punto que se notó es que la opción para creación de juegos 2D en Unreal Engine (Paper2D[2]) se encuentra muy poco actualizada.

#### Familiaridad con el ambiente de desarrollo

Para contar con un desarrollo expedito durante el trabajo de título, se realizó para ambos motores un prototipo de juego básico, el cual consiste en un nivel de plataformas. Esto con la finalidad de decidir cuál de ellos resultaba más fácil de utilizar. De esta prueba se observa que Unity utiliza código escrito en `C#`, mientras que Unreal Engine usa `C++`, ninguno de estos presenta una dificultad mayor. También se nota que ambas interfaces son lo suficientemente robustas y ofrecen más de las funcionalidades necesarias para la realización del juego propuesto.

La tabla 4.1.1 resume las diferencias entre los motores comparados observadas con anterioridad.

| Característica | Unity | Unreal |
|---|---|---|
| Precio | Gratis | Gratis |
| Soporte Tobii 4C | SDK y Documentación | SDK |
| Soporte juegos 2D | Mucho | Obsoleto |
| Lenguaje de programación | `C#` | `C++` |
| Familiaridad con entorno de desarrollo | Muy poca | Nula |

Tabla 4.1: Resumen comparativo entre motores de desarrollo de juegos

Luego, con los parámetros evaluados anteriormente, ya que no existe preferencia por las interfaces de los motores, se decide realizar este trabajo utilizando Unity, porque este presentó ventajas importantes en los 2 primeros puntos, principalmente el hecho de que Unity se encuentre mejor diseñado para el desarrollo de juegos 2D y Paper2D se encuentre obsoleto.

### 4.1.2. Desarrollo de un prototipo de aplicación en Unity

Para familiarizarse con el uso de Unity en conjunto con el *eyetracker*, se desarrolló un prototipo básico en donde una figura deba moverse hacia donde se encuentre observando el usuario. Para poder obtener las coordenadas de la vista del usuario Tobii provee en su SDK[3] la función `GetGazePoint()`, la cual entrega un objeto `GazePoint` que contiene un vector de 2 dimensiones con la posición (x,y) de la vista del usuario. La Figura 4.2 muestra el entorno de desarrollo de Unity con el prototipo realizado.

Antes de utilizar el *eyetracker* junto con Unity fue necesario instalar primero un software de Tobii el cual permite reconocer y calibrar el *eyetracker*. Luego se descargo e importó en

[2] `https://docs.unrealengine.com/en-US/Engine/Paper2D/index.html`
[3] `https://developer.tobii.com/pc-gaming/unity-sdk/scripting-api/`

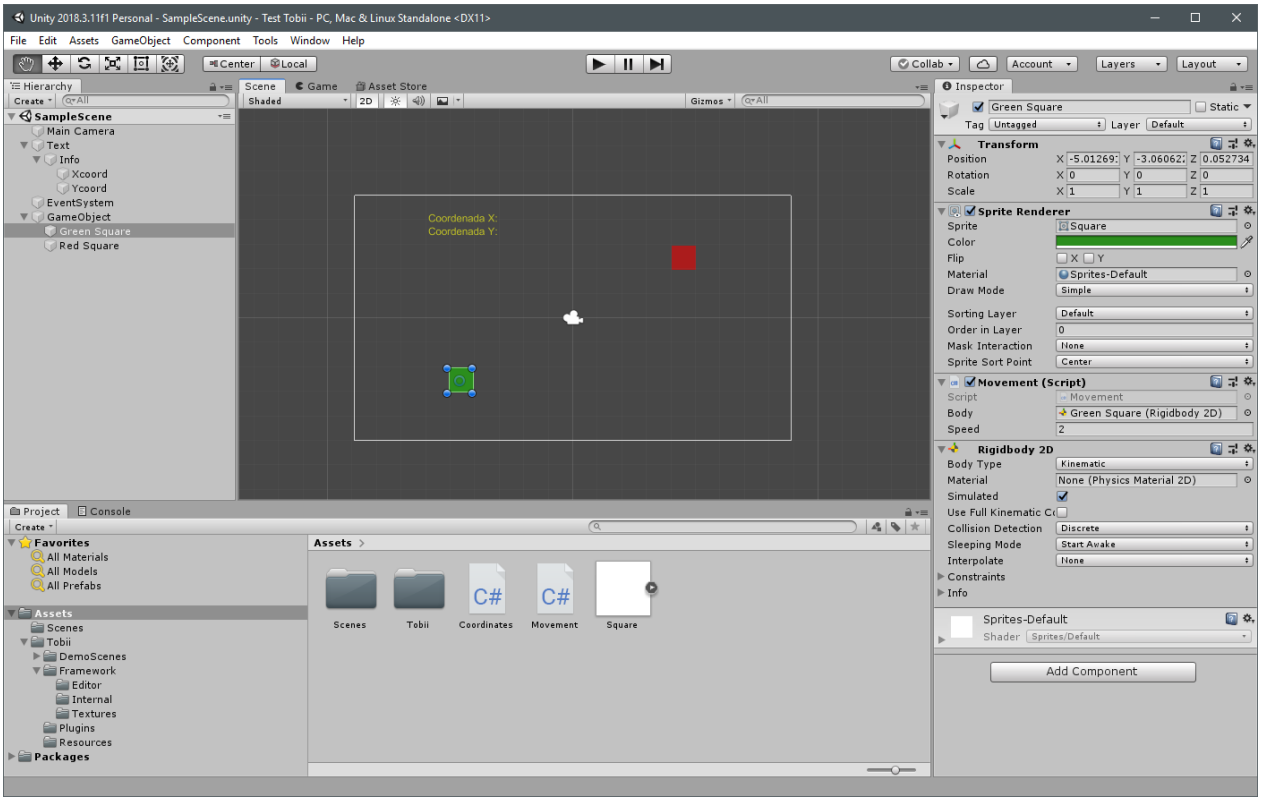


Figura 4.2: Prototipo de juego usando Unity en conjunto al *eyetracker*

el prototipo su SDK correspondiente. Durante el desarrollo de este prototipo se notó que el *eyetracker* funciona con una mayor tasa de error si recibe luz directamente, lo que se arregló cambiando el monitor de lugar en futuros trabajos.

Finalmente, y para mantener consistencia en la ejecución de este trabajo, se decidió utilizar el mismo hardware durante todo el transcurso del semestre, tanto para el desarrollo de la aplicación como para la evaluación de la misma. Al respecto, ya se cuenta con todo el hardware el cual consiste en:

- Tobii Eye Tracker 4C
- Mando de Xbox One
- Monitor LG 27MK400H-B

## 4.2. Desarrollo

En Unity, los juegos corresponden a un conjunto de escenas donde cada una posee una serie de *GameObjects*, estos objetos representan todos los elementos a utilizar para la escena en la que están y pueden ser modificados utilizando diversas componentes. Es posible añadir como componente a cada objetos scripts escritos en `C#` con código que controla y modifica su comportamiento.

A continuación se detallan los distintos elementos y scripts que se desarrolló para realizar el juego propuesto:

### 4.2.1. Escenas

Se crean dos escenas para manejar el juego, la primera corresponde al menú y contiene 2 selectores para decidir con qué se controlará las mecánicas tanto de apuntar y disparar, esta escena se puede observar en la figura 4.3. La segunda escena corresponde a la etapa a jugar y es cargada una vez se eligen las mecánicas, utilizando los parámetros seleccionados para el manejo del personaje.

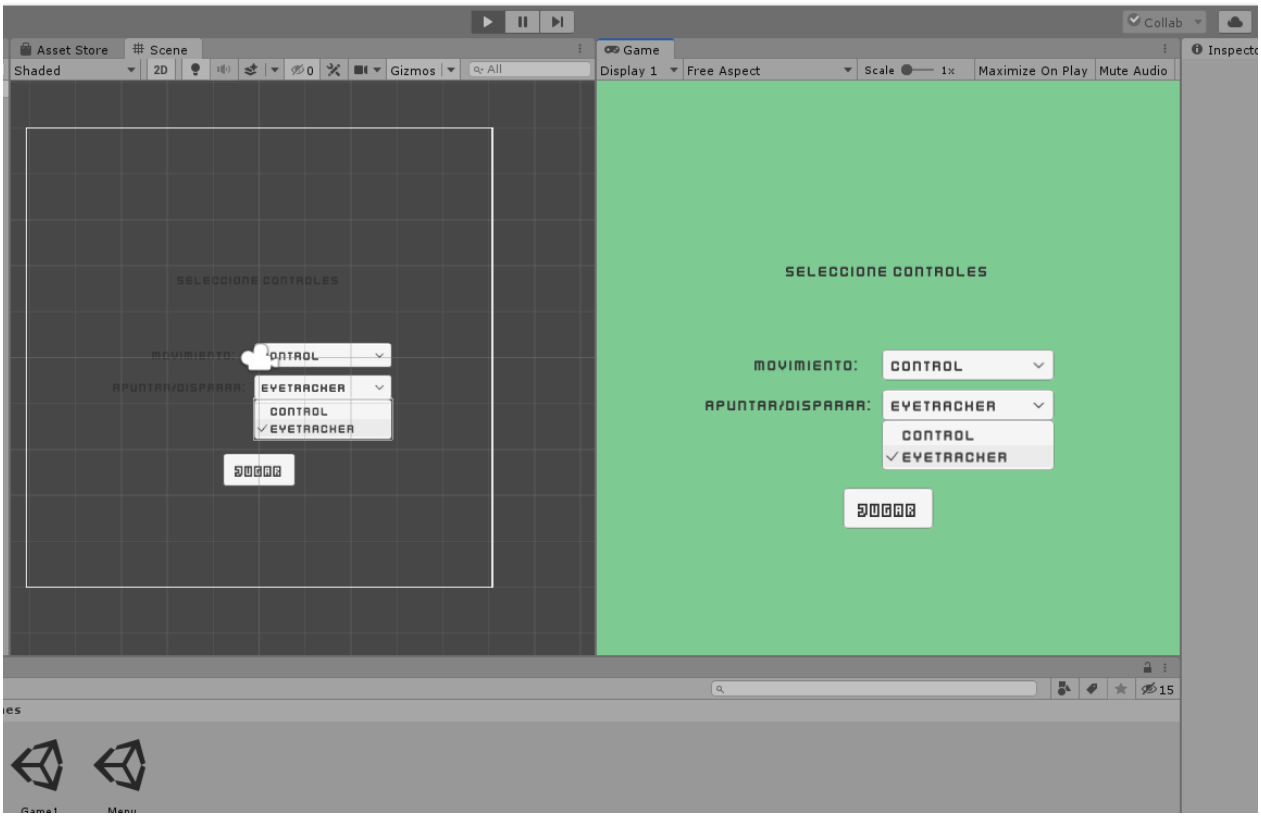


Figura 4.3: Menu inicial para seleccionar como controlar las mecánicas del juego

### 4.2.2. Personaje

Corresponde al personaje principal del juego, el cual es controlado por el jugador. Para representarlo en la pantalla se utilizó uno de los sprites descargados previamente. Para detectar si una bala colisiona con el personaje se le agregó una componente de unity llamada circle collider 2D, del tamaño del sprite colocado. Además a este collider se asoció a una componente llamada RigidBody 2D, que permite detener al jugador cuando se encuentre con enemigos o paredes. Una imagen del personaje con su collider asociado se muestra en la figura 4.4, ahí se puede notar que el collider es de un tamaño menor a la imagen del personaje,

esto con la finalidad de facilitar al jugador esquivar las balas mientras que enfoca su atención en otras partes y para prevenir una sensación de injusticia si el borde del personaje toca el borde de una bala.

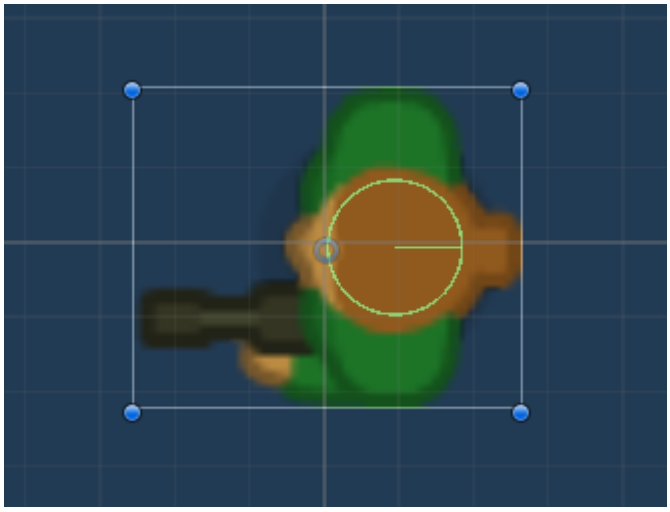

Figura 4.4: Personaje principal y su *hitbox*

Además de las componentes de Unity utilizadas se crearon 2 scripts que administran el control del personaje, los que son:

- `PlayerController`: Se encarga de manejar el movimiento del personaje y de actualizar las métricas a guardar en el transcurso del juego.

- `Weapon`: Controla los parámetros del arma del jugador y su disparo.

Estos scripts serán explicados con mayor detalle en la sección 4.4.

### 4.2.3. Cámara

Se utiliza la cámara base que viene por defecto en la escena, a la que se le crea como componente un script llamado `CameraController`, que le da dos comportamientos distintos dependiendo de cómo se esté controlando el movimiento del jugador. Estos comportamientos se pueden observar en la sección 4.4.

### 4.2.4. Bala

Se crean 2 *GameObject*, `PlayerBullet` y `EnemyBullet` que heredan de la misma clase `Bullet` la que define su comportamiento como velocidad y control de colisiones, la diferencia entre estas es el tipo de objetivo al que le hacen daño y la forma de verificar colisiones y color de las balas.

Para determinar el tipo de objetivo al que daña cada tipo de bala, al personaje principal se le otorga la etiqueta de player, mientras que a los enemigos se les otorga una etiqueta de enemy, por lo que cada tipo de bala al generarse una colisión verifica si el objeto que encuentra posee la etiqueta correspondiente. A cada bala se le asocia también una componente de tipo *Audio Source* que se encarga de realizar el sonido de cada disparo.

### 4.2.5. Enemigos

Para generar a los enemigos se creó primero un *GameObject* que representa a la base de la torre, la cual contiene de componente otro objeto que consiste en la torre en si, esta torre posee un componente *Circle Collider* que verifica si hace contacto con una bala del personaje principal.

En la figura 4.5 se puede observar los *sprites* elegidos para armar las torres enemigas, además de la torre armada en el juego junto a su collider, este es de un tamaño un poco que la imagen para facilitar al jugador dispararle en caso de la bala encontrarse tocando el borde de la torre. Se puede ver además que se cambió el color original, usando un color que contrasta más con los colores del suelo utilizados en el juego y así reconocerse con mayor facilidad.

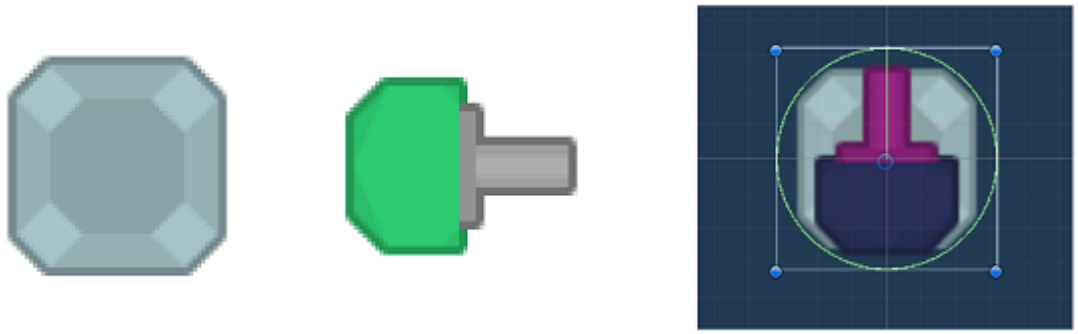

Figura 4.5: Base(izquierda) y torre(centro) usadas para armar a los enemigos(derecha)

Para este objeto se tuvo que crear también los siguientes scripts:

- `Enemy`: Se encuentra a cargo de manejar la vida de la torre y de rotarla para que se encuentre apuntando al enemigo, también se encarga de activar la torre cuando el jugador llega a la zona en la que se encuentra.

- `EnemyWeapon`: Controla la frecuencia de los disparos del enemigo.

- `FocusEnemy`: Se utiliza cuando el modo de apuntar se encuentra configurado en *eyetracker* y sirve para determinar la torre que observa el jugador actualmente y si ha pasado

el tiempo suficiente desde que lo observa como para que sea apuntado automáticamente.

### 4.2.6. Pared

Para delimitar cada área y el mapa del juego se crea un objeto que posee un *Box Collider 2D* y *RigidBody 2D* que evitan el avance del jugador y al mismo tiempo detiene las balas con las que hace contacto. Este objeto no posee *sprite* asociado ya que se usa en conjunto al mapa generado en la sección 4.3.

Para este objeto se escribió el script `Wall`, el que posee una función que se gatilla cuando un objeto se sobrepone a este y verifica si es una bala, y en caso de serlo la elimina.

### 4.2.7. Zonas

Se crea un conjunto de objetos correspondientes a las diferentes zonas, en las que se almacenan los enemigos que habitan cada una, además de los siguientes 2 objetos:

- **Vallas:** Previenen al jugador de avanzar a la siguiente zona, el script `Fence` las elimina una vez sean eliminados todos los enemigos de su zona correspondiente.

- **Gatillo:** Objeto invisible que posee un *Box Collider 2D* que activa a los enemigos de la zona para que apunten y disparen al personaje.

## 4.3. Mapa

Para desplegar la etapa en la pantalla se utilizó *tilemaps* de Unity, que separan la escena en una grilla de proporciones determinadas y en donde a cada celda de la grilla se le puede colocar un *sprite*. Estos tilemaps fueron generados con los assets descargados del sitio mencionado previamente en la sección 3.4 y utilizando el programa Tiled[4] para facilitar su fabricación, en la figura 4.6 se puede ver la fabricación del mapa en Tiled utilizando los assets descargados.

Una vez creados los mapas se importaron en Unity utilizando una librería llamada SuperTiled2Unity[5] que recibe el archivo generado por Tiled y lo convierte en un *Tilemap* que puede ser utilizado directo en la escena.

[4] `https://www.mapeditor.org/`
[5] `https://seanba.itch.io/supertiled2unity`

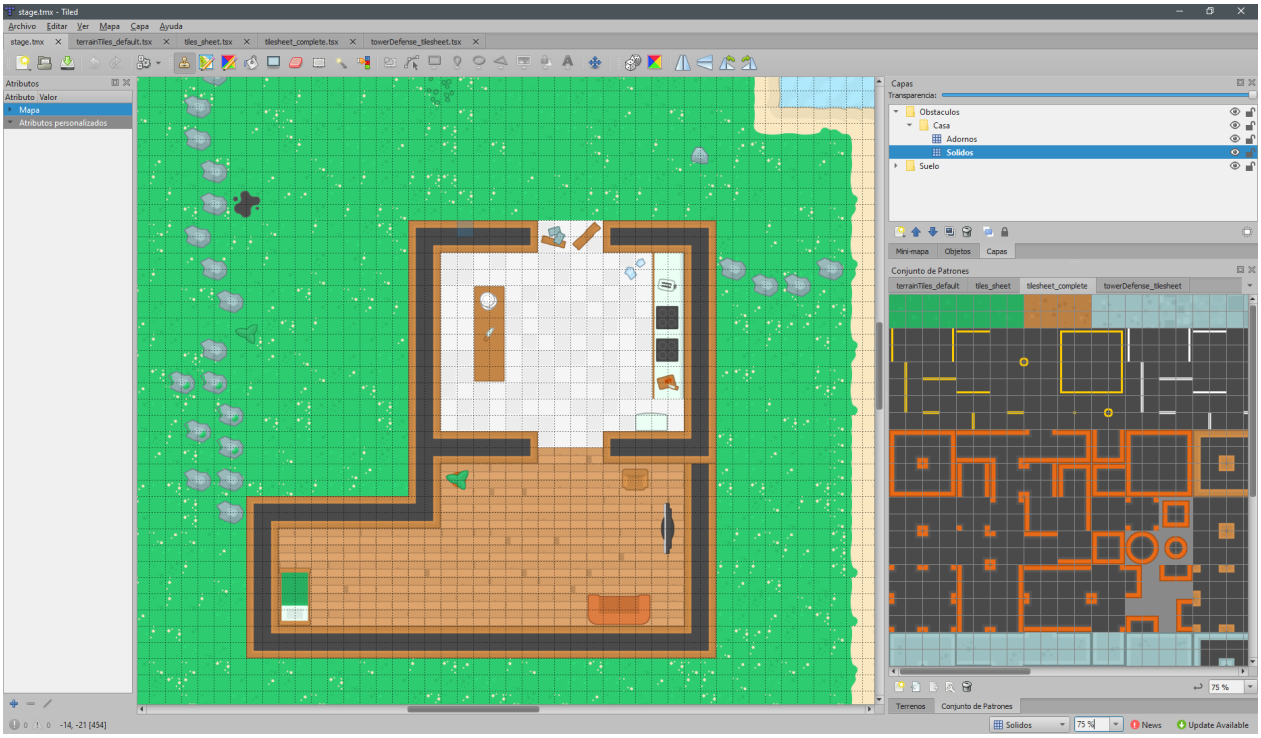

Figura 4.6: Creación de la etapa utilizando programa Tiled

## 4.4. Mecánicas

Como se ha mencionado anteriormente, lo importante para este trabajo de título es permitir que el juego sea controlable con la vista de distintas maneras. Esto es, se debe poder configurar de manera independiente las mecánicas de movimiento y de apuntar. En ambos casos se utilizó la función `GetGazePoint()` de la API de Tobii que retorna un elemento gazePoint que determina la posición de la vista en la pantalla.

Esta posición no necesariamente se encuentra ajustada a la cámara del juego, por lo que hay que realizar el cálculo de la posición de la vista respecto a lo que observa la cámara en la escena utilizando la función `Camera.main.ScreenToWorldPoint()`, obteniéndose así las coordenadas del juego en la que se encuentra mirando el jugador. Estas coordenadas serán utilizadas para controlar las mecánicas en caso de haberse configurado para utilizar el *eyetracker*.

### 4.4.1. Movimiento

En caso de utilizar mando de juego para moverse, se calcula la inclinación de la palanca izquierda tanto horizontal como verticalmente, obteniéndose un vector que luego se normaliza y pondera por un valor determinado correspondiente a la velocidad del personaje. Este cálculo se realiza de forma similar cuando se utiliza el *eyetracker*, con la diferencia que el vector es calculado en relación a la posición del personaje comparada a la posición observada.

Cuando el personaje se mueve, en caso de utilizarse mando tradicional, la cámara lo sigue de manera que siempre se muestre en el centro de la pantalla. En cambio, con *eyetracker* se define un radio desde el centro en el que el personaje puede moverse libremente sin intervenir la cámara. Cuando este radio es superado, la cámara se mueve en dirección al personaje hasta volver al límite.

### 4.4.2. Apuntar y disparar

Al igual que con el movimiento, cuando se configura el apuntar y disparar con mando, se calcula el vector de inclinación de la palanca derecha del mando, para rotar el personaje en aquella dirección y al presionar uno de los gatillos derechos se dispara una bala en la dirección de la rotación.

Para el caso de usar *eyetracker* para apuntar se debe determinar primero si los controles serán completamente con la mirada o si serán híbridos. En caso de ser híbridos el personaje apuntará en la dirección a la que está mirando el jugador, a menos que algún enemigo gatille el método `FocusEnemy`. Esto significa que el jugador está observando cerca de un enemigo y se apunta directamente al enemigo que corresponde para evitar fallar el disparo por culpa de la precisión del dispositivo. Al igual que con el caso del mando tradicional, el disparo se realiza cuando el jugador oprime uno de los gatillos derechos.

En caso de jugarse completamente con la mirada, el personaje apuntará a donde sea que esté observando el jugador hasta que mire a un enemigo. En este caso el personaje le apuntará y no dejara de hacerlo hasta derrotar al enemigo o hasta que el usuario observe a otro. Esto se hace con la finalidad de permitir al usuario esquivar los ataques hacia él, ya que en este caso para esquivar el jugador tendría que observar el lugar al que desea moverse, dejando de mirar al enemigo. Como en este caso se usa completamente la mirada se decidió no realizar los disparos con los gatillos del mando de juego y en cambio el personaje dispara automáticamente con una frecuencia dada.

## 4.5. Métricas

Para probar el rendimiento del jugador al finalizar la etapa con las diferentes mecánicas se almacenan distintos valores, los cuales son:

- **Controles:** Se almacena si se está utilizando mando tradicional o *eyetracker* para el movimiento del personaje, y para apuntar también.
- **Daño recibido:** Por cada bala enemiga recibida por el personaje se incrementa este valor en 1.
- **Tiempo en salir de la primera zona:** Para saber cuánto tiempo le toma al jugador familiarizarse con los controles se mide el tiempo que le toma llegar a la segunda zona de la etapa.
- **Tiempo en finalizar el juego:** Se calcula por último el tiempo que transcurre desde

que el jugador deja la primera zona hasta que llega al final de la última zona. Esto corresponde al tiempo que el jugador estuvo disparando a enemigos y esquivando sus ataques.

Todos estos valores son guardados en un archivo que se genera automáticamente al finalizar la etapa, para poder ser estudiados posteriormente junto a los resultados de todos los demás usuarios.

## 4.6. Validación Exploratoria

Antes de realizar la prueba final con usuarios se realizó una prueba de exploratoria del juego sin finalizar, para obtener retroalimentación y realizar ajustes necesarios respecto a la jugabilidad del software desarrollado. Para esta prueba asistieron 5 voluntarios.

La información que se obtuvo al realizar estas pruebas fue:

- El uso del eyetracker tuvo buena recepción, considerándose que mejoraba la experiencia del juego.
- En esta fase se recomendó la creación de una zona inicial sin enemigos para dar tiempo a los jugadores de acostumbrarse a los controles y mecánicas.
- Cuando se utilizaba *eyetracking* para mover al personaje muchos usuarios se encontraban con complicaciones dejando al personaje quieto ya que la etapa era muy homogénea. Esto se solucionó agregando pequeños puntos de interés como piedras o distintos colores de pasto para dar al jugador puntos en los que puedan enfocar su mirada en todo lugar.
- Fue sugerido cambiar la imagen del personaje principal ya que como se mostraba provocaba dificultad para reconocer distinguir entre su frente y espalda. Esto fue solucionado indicando con mayor claridad el arma del personaje para saber cual era su frente.
- A algunos usuarios no les gustó el gatillo elegido para indicar disparos, por lo que se permitió usar ambos gatillos de la derecha del mando.
- La sensibilidad de la palanca derecha era muy alta, por lo que al soltarse la dirección en la que apuntaba el personaje se cambiaba a una no deseada. Para solucionar ese error se fijó un umbral en el que la palanca derecha no reporta su posición.

# Capítulo 5

# Prueba de Concepto

Con el objetivo de determinar si el uso de *eyetracking* en el juego desarrollado resulta mejor a utilizar únicamente un mando tradicional en tanto rendimiento como en experiencia se realizó un estudio de usuarios con una muestra de participantes en donde debieron finalizar el juego de dos maneras distintas, una vez sin utilizar *eyetracker* y otra vez con alguna de las variantes que sí lo utiliza. Posterior a cada prueba se les aplicó un cuestionario de experiencia de juego.

## 5.1. Metodología

En esta sección se detallan los elementos principales del diseño empírico del estudio de usuarios ejecutado.

### 5.1.1. Participantes

Se reclutó a 24 personas para participar en un estudio de usuarios respecto a la interacción a través de *eyetracking* en videojuegos en bloques de 20 minutos repartidos en dos días. La muestra de usuarios que participó del estudio consistió en miembros de la Universidad de Chile entre 22 y 32 años, cuya experiencia en videojuegos variaba entre nula o mucha.

### 5.1.2. Materiales

Para las pruebas a usuarios se utilizó los mismos materiales usados en el transcurso del trabajo y consistió en:

- Tobii Eye Tracker 4C
- Mando de Xbox One

- Monitor LG 27MK400H-B

### 5.1.3. Procedimiento

Para prevenir que los resultados se vieran afectados por el orden de juego si todas las pruebas comenzaban con la variante de mando tradicional, se dividió a los voluntarios en 6 grupos diferentes los cuales son:

- Primera prueba mando, segunda prueba movimiento con *eyetracker*
- Primera prueba mando, segunda prueba apuntar con *eyetracker*
- Primera prueba mando, segunda prueba completamente *eyetracker*
- Primera prueba movimiento con *eyetracker*, segunda prueba mando
- Primera prueba apuntar con *eyetracker*, segunda prueba mando
- Primera prueba completamente *eyetracker*, segunda prueba mando

Para cada experimento realizado, se le indicó al usuario en qué consistía la prueba en la que se encontraban, mencionando que deberían superar una etapa de juego 2 veces, una de estas utilizando la mirada.

Posterior a la explicación se le solicitó al usuario sentarse en la silla ubicada frente al monitor con *eyetracker* y seguir unas instrucciones que aparecerían en pantalla. Esto con el fin de calibrar el dispositivo con los ojos del usuario.

Una vez finalizada la calibración del *eyetracker* se inicia el juego y se solicita al usuario que complete el nivel 2 veces, con las mecánicas correspondientes al grupo que fue repartido.

Finalizando cada partida los participantes debieron contestar un cuestionario de jugabilidad. El cuestionario aplicado es una versión traducida del *Game Experience Questionnaire* [6], el cual es considerado como un estándar de industria por la comunidad de investigadores en Interacción Humano-Computador [8] y mide cómo se sintió el jugador durante y posterior a la experiencia de juego. Al cuestionario se le agregó además al final la opción de dejar comentarios abiertos sobre la experiencia.

## 5.2. Resultados

Del estudio de usuarios ejecutado se recopiló la información tanto de rendimiento como del cuestionario de experiencia de juego. Los datos procesados se presentan a continuación:

### 5.2.1. Rendimiento

En promedio los jugadores tienen 25.17 años, se demoraron 164.89 segundos en superar el nivel del juego, recibiendo en promedio 32.17 balas. Los resultados de jugar utilizando

únicamente mando tradicional con alguna de las variantes se obtiene para el daño recibido y tiempo de juego, se muestran en figura 5.1. En ésta se puede notar que no hubo mucha diferencia entre ambas variantes, presentando promedios de daño recibido y tiempo muy similares.

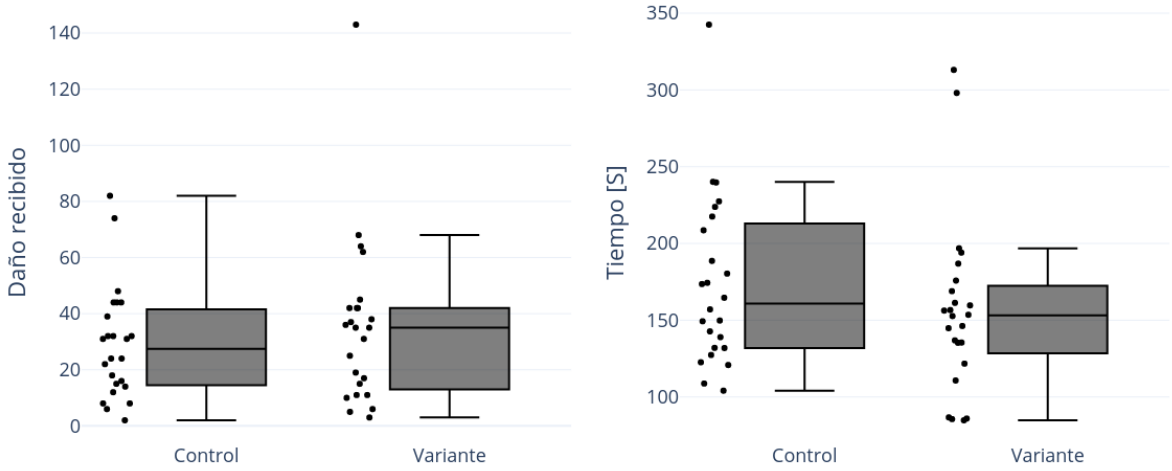


Figura 5.1: Diagrama de caja para comparar daño recibido y tiempo en completar nivel entre el uso de mando tradicional y variante con *eyetracking*

Las diferencias se hacen notar cuando se separa cada grupo por su tipo de control alternativo, en la figura 5.2 se muestra la diferencia cuando el jugador utiliza la mirada para controlar el apuntar como variante alternativa, en la figura 5.3 se presenta la diferencia entre completamente mando y movimiento con la vista, y finalmente la comparación entre solo mando y solo *eyetracker* se enseña en la figura 5.4.

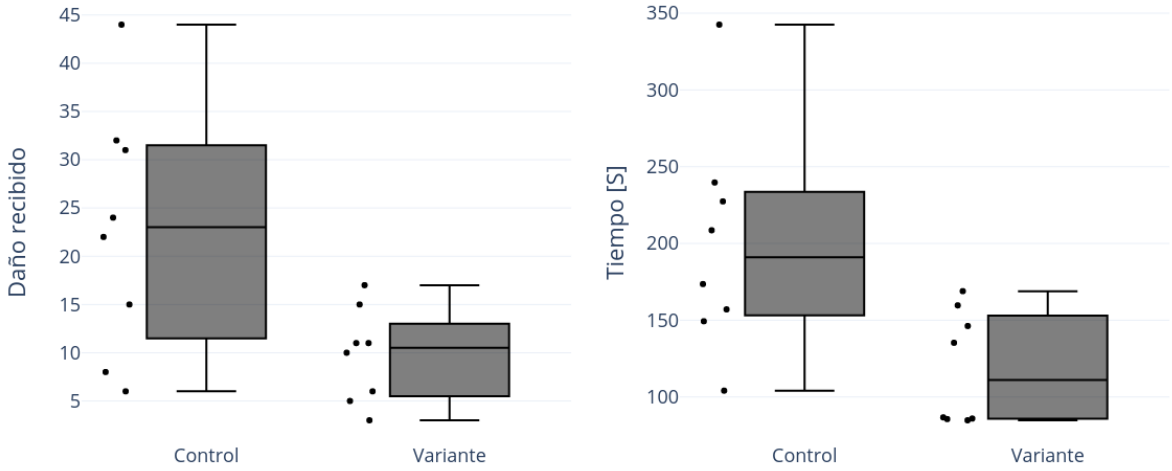


Figura 5.2: Diagrama de caja para comparar daño recibido y tiempo en completar nivel entre el uso de mando tradicional y variante de apuntar con *eyetracking*

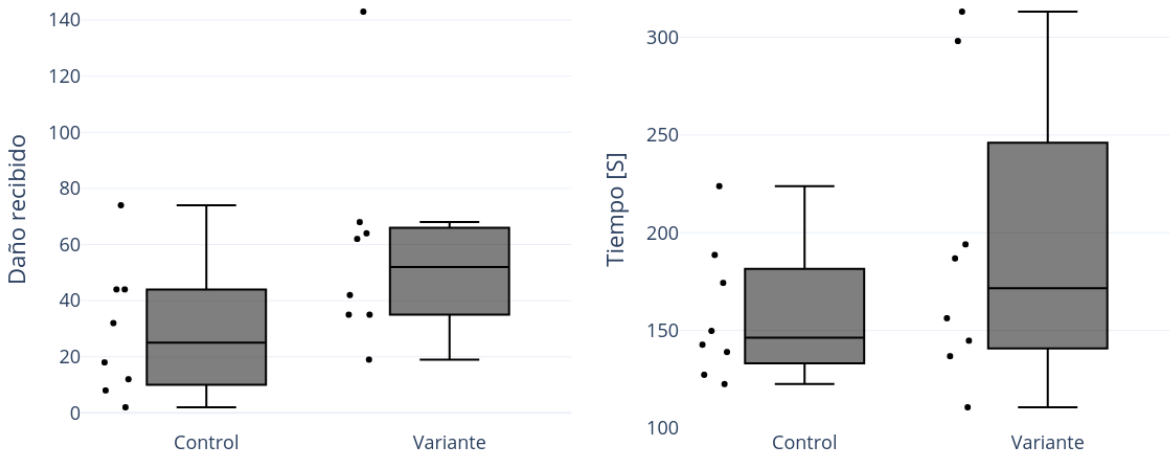


Figura 5.3: Diagrama de caja para comparar daño recibido y tiempo en completar nivel entre el uso de mando tradicional y variante de movimiento con *eyetracking*

Con esos gráficos se puede notar que la variante en la que se controla el apuntar del personaje con la mirada fue la que tuvo un mayor rendimiento, viéndose tanto el daño recibido como el tiempo en completar reducidos en promedio. Mientras que en ambas variantes en las que se controla el movimiento con la vista se vieron perjudicados, siendo el caso híbrido de estos el peor de todos.

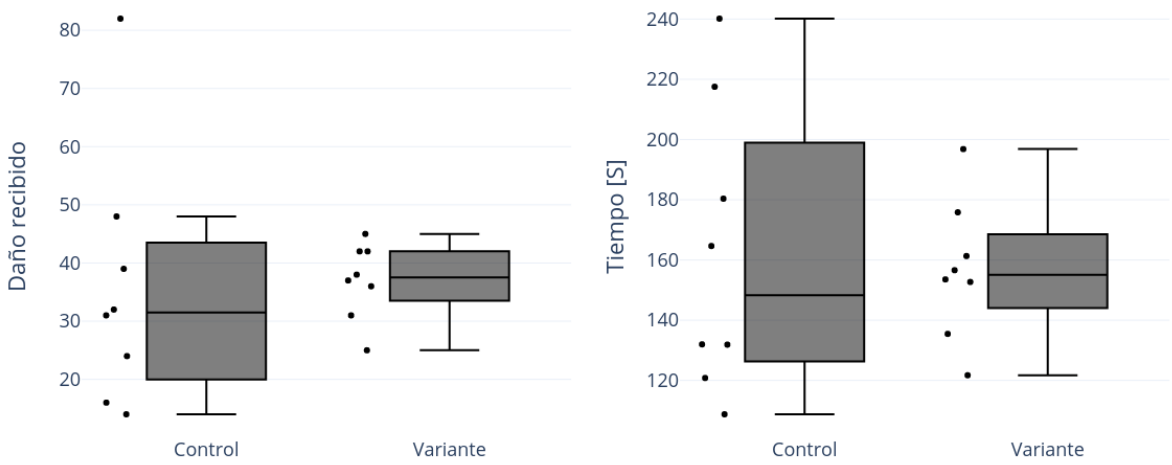


Figura 5.4: Diagrama de caja para comparar daño recibido y tiempo en completar nivel entre el uso de mando tradicional y variante completamente con *eyetracking*

### 5.2.2. Experiencia

Se utilizó dos módulos del cuestionario de experiencia de juego. El primero es el modulo base que mide la experiencia del jugador durante el transcurso de la partida. Este módulo mide las siguientes dimensiones:

- **Habilidad:** Sensación positiva que indica que tan hábil se siente el jugador con el juego. Se relaciona con sentimientos de logro y orgullo y motiva al jugador a continuar con el juego.
- **Inmersión:** Positivo, indica que tan "dentro del juego"se encuentra el usuario.
- **Flujo de juego:** Dimensión positiva, buen flujo significa que el juego mantuvo la atención del jugador, ignorando estímulos externos.
- **Fastidio:** Sensación negativa, un juego que provoque fastidio significa que será dejado de lado con mucha probabilidad.
- **Desafío:** Depende del usuario, un juego percibido como desafiante puede motivar a algunos jugadores a continuar mientras que otros podrían querer no continuar.
- **Percepción Positiva**
- **Percepción Negativa**

El segundo módulo del cuestionario utilizado corresponde al la sensación del jugador posterior a la experiencia de juego. Con este módulo se miden las siguientes dimensiones:

- **Experiencia positiva**
- **Experiencia negativa**
- **Cansancio**

- **Vuelta a realidad:** Si la vuelta a la realidad resulta muy fácil quiere decir que el jugador no se encontraba lo suficientemente inmerso en la experiencia.

En las tablas 5.1 y 5.2 se presentan los resultados del cuestionario base y posterior al juego respectivamente. Se presenta la diferencia de los promedios de cada dimensión entre la variante con *eyetracker* y solo con mando de juego, un número positivo en las tablas significa que el resultado fue mayor para el juego utilizando la mirada, mientras que un valor negativo indica que la dimensión obtuvo mayor valor al jugarse con mando únicamente. Los valores que pueden tener cada dimensión era entre 1 y 5, por lo que una diferencia de 1 punto en la tabla es bastante alta.

| Dimensión | Todos | Apuntar vista | Movimiento vista | Completamente vista |
|---|---|---|---|---|
| Habilidad | -0.23 | 1.68 | -1.65 | -0.70 |
| Inmersión | 0.56 | 1.10 | 0.25 | 0.33 |
| Flujo de juego | 0.62 | 1.03 | 0.48 | 0.35 |
| Fastidio | 0.10 | -0.33 | 0.54 | 0.08 |
| Desafío | 0.39 | -0.75 | 1.45 | 0.48 |
| Afecto Positivo | -0.17 | -0.41 | 0.22 | -0.31 |
| Afecto Negativo | 0.32 | 0.92 | -0.14 | 0.19 |

Tabla 5.1: Diferencia dimensiones de análisis entre la variante utilizando mirada y solamente mando tradicional durante el juego

Los resultados de la tabla 5.1 que más llaman la atención fue en el caso de sensación de habilidad, en donde la variante de movimiento con la mirada obtuvo 1.68 puntos más que su contraparte con mando tradicional (2.78 contra 4.45), mientras que en esa misma dimensión la variante de apuntar con la vista tuvo 1.65 puntos menos que su contraparte de únicamente mando.

El uso de la vista para apuntar en su mayoría tuvo mejor puntuación que el mando para las dimensiones positivas, siendo la excepción el afecto positivo que obtuvo solo 0.41 puntos menos. Aparte de la habilidad en el caso de movimiento con la vista y su dimensión de desafío, las variantes en que el movimiento se controla con la vista no tuvieron una diferencia muy significativa respecto a sus contrapartes con mando de juego tradicional.

| Dimensión | Todos | Apuntar vista | Movimiento vista | Completamente vista |
|---|---|---|---|---|
| Experiencia positiva | 0.43 | 1.38 | -0.45 | 0.35 |
| Experiencia negativa | -0.09 | -0.40 | 0.10 | 0.02 |
| Cansancio | 0.31 | -0.50 | 0.63 | 0.81 |
| Vuelta a realidad | 0.26 | 0.00 | 0.42 | 0.38 |

Tabla 5.2: Diferencia dimensiones de análisis entre la variante utilizando mirada y solamente mando tradicional después del juego

En la tabla correspondiente a la experiencia posterior al juego se puede notar nuevamente que la mayor diferencia fue obtenida en la variante en la que se apunta con el *eyetracker*, con 1.38 puntos por sobre a su contraparte, teniendo también una sensación de cansancio

menor a diferencia de las variantes de movimiento con la vista y completamente vista que reportaron un cansancio mayor a la experiencia con tan solo el mando.

### 5.2.3. Comentarios

De los comentarios obtenidos al final del cuestionario de obtuvo la siguiente información:

- En general la percepción de los usuarios fue bastante positiva, considerando la experiencia como novedosa.
- Todos los comentarios negativos fue en casos en que el usuario debía moverse al usar la mirada, comentando que lo encontraban muy difícil o complicado y prefirieron el mando tradicional. Tres de estos usuarios reportaron una leve molestia en la mirada durante la experiencia.
- Algunos usuarios hubiesen deseado jugar un poco más o explorar otras variantes ofrecidas en el juego, Además de querer conocer sobre la existencia de otros juegos que utilicen esta tecnología.
- Los usuarios con menor experiencia en juegos fueron quienes encontraron más entretenido el uso de *eyetracking* para jugar.

## 5.3. Discusión

De los resultados se puede determinar que tanto el rendimiento de los usuarios, como sus respuestas al cuestionario, e incluso sus comentarios y opiniones concuerdan en que la variante en la que se combina el mando para el movimiento, con el *eyetracker* para apuntar es la mejor alternativa de las 4 propuestas en este trabajo, superando incluso al uso del mando tradicional, esto puede ser debido a que su uso no resulta antinatural a diferencia del movimiento con *eyetracking* puesto que es necesario observar a los enemigos para poder apuntar, entonces esta variante resulta ser un apoyo al jugador mientras que las otras dos variantes presentaban un desafío más que un apoyo.

Esto podría llevar a suponer que todos los controles híbridos son de las mejores alternativas, pero no es el caso ya que el uso de solamente la mirada superó a la variante híbrida en la que se apunta con mando y se mueve con la mirada. Lo que significa nuevamente que apuntar con la vista es la alternativa preferida en juegos del género seleccionado y que el movimiento con la vista resulta perjudicial para el jugador.

Sobre el cuestionario de experiencia de juego se puede notar que la experiencia de un jugador se correlaciona a su rendimiento en el juego. Los casos que los que se obtuvo peores puntajes de experiencia fueron también los casos en que los jugadores rindieron peor, habiendo recibido más daño o demorándose más de lo necesario. Se puede concluir de este punto que si un juego presenta mucha dificultad a quien lo está jugando, superando lo que está dispuesto a soportar, este sentirá una mayor frustración y menor deseo de continuar con el juego.

Si bien hubo casos en que se consideraba complicado el juego utilizando la mirada, en general se notó que la gente aprecia métodos de juegos novedosos y antes no vistos, lo que significa que es importante explorar y encontrar nuevas maneras de jugar, ya sea por medio de mecanismos de entrada novedosos o primicias nuevas de juegos.

Recomendaciones que se pueden obtener en base a los resultados obtenidos son:

- Para juegos que ocurren en tiempo real no es recomendable utilizar *eyetracking* para movimientos muy finos ni de reacción rápida.

- Dar bastante holgura al registrar la vista ya que esta por naturaleza posee movimientos erráticos

- Para juegos de acción resulta mejor utilizar *eyetracking* como apoyo a un método de entrada diferente y no como método principal.

# Capítulo 6

# Conclusión y Trabajo Futuro

Para juegos de disparos 2D con vista superior es posible encontrar formas de jugar que conllevan a una mejor experiencia de juego que utilizar únicamente mandos de juego tradicionales. Una forma encontrada en este trabajo es realizar movimientos por medio del mando mientras que apuntar a enemigos se realice observando a los objetivos.

A continuación se retoman los objetivos específicos propuestos, además de el cumplimiento para cada uno de estos:

1. **Definir mecanismos de interacción mediados por *eyetracking*, abarcando distintos tipos de configuración de su uso, ya sea utilizando un mando tradicional para controles específicos junto al *eyetracker* para interacciones específicas, o utilizando únicamente *eyetracking* para todas las interacciones:** Se determinó un género de juego y sus mecánicas que se utilizarían con y sin *eyetracking*

2. **Desarrollar un videojuego básico controlable por *eyetracking*, de tal manera que permita adaptar sus controles en función de las configuraciones definidas:** Se diseñó e implementó un juego del género determinado y se programó su uso en conjunto a un *eyetracker*

3. **Generar métricas de uso de manera automática que permitan describir el rendimiento y experiencia del usuario durante la sesión de juego:** Se generaron estas métricas y se utilizaron para realizar pruebas a usuarios en conjunto con cuestionarios de experiencia de juego.

Con los objetivos anteriores alcanzados se da por cumplido el objetivo general de este trabajo de título, puesto que se diseñó e implementó un videojuego en el cual se incorporaron mecánicas de seguimiento de mirada, con el fin de ser medido comparativamente el rendimiento y grado de satisfacción de los usuarios en distintas configuraciones de juego.

En retrospectiva se debió haber indagado más sobre el motor en el que se va a desarrollar el videojuego, puesto que el no haberlo hecho a tiempo atrasó el avance del trabajo e incluso lo retrocedió al haber tenido que reescribir partes de proyecto para poder avanzar mejor

posterior a las correcciones.

Posibilidades de trabajos futuros incluyen el estudio de juegos con seguimiento ocular de distintos géneros y mecánicas, como también explorar distintas maneras de interactuar con el juego usando la mirada. También se puede retomar el trabajo realizado con un mayor número de usuarios para obtener resultados más determinantes.

Usos en los que queda todavía mucho por estudiar es el uso de *eyetracking* para experiencias de realidad virtual o realidad aumentada, ya que el surgimiento de hardware para esto de menor coste hace que vaya a ser más accesible a las masas y más personas adquirirán estos aparatos para su uso lúdico.

# Bibliografía


[1] Let's talk accuracy and precision – tobii eye tracking support. `https://help.tobii.com/hc/en-us/articles/213534825-Let-s-talk-accuracy-and-precision`. Última visita en 1 de julio de 2019.

[2] Tobii gaming. `https://gaming.tobii.com/games`. Última visita en 18 de abril de 2019.

[3] Dechant, M., Stavness, I., Mairena, A., and Mandryk, R. L. Empirical evaluation of hybrid gaze-controller selection techniques in a gaming context. In *Proceedings of the 2018 Annual Symposium on Computer-Human Interaction in Play* (New York, NY, USA, 2018), CHI PLAY '18, ACM, pp. 73–85.

[4] Duchowski, A. *Eye Tracking Methodology*. Springer London, 2007.

[5] Duchowski, A. T. A breadth-first survey of eye-tracking applications. *Behavior Research Methods, Instruments, & Computers 34*, 4 (Nov 2002), 455–470.

[6] IJsselsteijn, W. A., de Kort, Y. A. W., and Poels, K. *The Game Experience Questionnaire*. Technische Universiteit Eindhoven, 2013.

[7] Jacob, R. J. K. What you look at is what you get: Eye movement-based interaction techniques. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (New York, NY, USA, 1990), CHI '90, ACM, pp. 11–18.

[8] Law, E. L.-C., Brühlmann, F., and Mekler, E. D. Systematic review and validation of the game experience questionnaire (geq) - implications for citation and reporting practice. In *Proceedings of the 2018 Annual Symposium on Computer-Human Interaction in Play* (New York, NY, USA, 2018), CHI PLAY '18, ACM, pp. 257–270.

[9] Miluzzo, E., Wang, T., and Campbell, A. Eyephone: Activating mobile phones with your eyes. pp. 15–20.

[10] Nacke, L. E., Stellmach, S., Sasse, D., and Lindley, C. A. Gameplay experience in a gaze interaction game. *arXiv e-prints* (Apr 2010), arXiv:1004.0259.

[11] Navarro, D., and Sundstedt, V. Simplifying game mechanics: Gaze as an implicit interaction method. In *SIGGRAPH Asia 2017 Technical Briefs* (New York, NY, USA, 2017), SA '17, ACM, pp. 4:1–4:4.

[12] Sundstedt, V. Gazing at games: Using eye tracking to control virtual characters. In *ACM SIGGRAPH 2010 Courses* (New York, NY, USA, 2010), SIGGRAPH '10, ACM, pp. 5:1–5:160.

[13] Sundstedt, V. Gazing at games: An introduction to eye tracking control. *Synthesis Lectures on Computer Graphics and Animation 5*, 1 (Mar. 2012), 1–113.

[14] Velichkovsky, B., A. Rumyantsev, M., and A. Morozov, M. New solution to the midas touch problem: Identification of visual commands via extraction of focal fixations. *Procedia Computer Science 39* (12 2014).

[15] Velloso, E., and Carter, M. The emergence of eyeplay: A survey of eye interaction in games. In *Proceedings of the 2016 Annual Symposium on Computer-Human Interaction in Play* (New York, NY, USA, 2016), CHI PLAY '16, ACM, pp. 171–185.

[16] Young, L. R., and Sheena, D. Survey of eye movement recording methods. *Behavior Research Methods & Instrumentation 7*, 5 (Sep 1975), 397–429.

# Apéndices

## Apéndice : Cuestionario de experiencia de juego

Sólo se presentan los módulos del cuestionario [6] utilizados dentro de la encuesta diseñada para esta memoria.

### Módulo base

Para cada uno de los siguiente puntos indique cómo se sintió **DURANTE** el juego, usando la siguiente escala:

| Para nada | Un poco | Más o menos | Bastante | Totalmente |
|---|---|---|---|---|
| 0 | 1 | 2 | 3 | 4 |

1. Me sentí contento
2. Me sentí hábil
3. Estuve interesado en su historia
4. Pensé que fue divertido
5. Tuve toda mi atención puesta mientras jugaba
6. Me sentí feliz
7. Me puso de mal humor
8. Pensé en otras cosas mientras jugaba
9. Me pareció tedioso
10. Me sentí competente
11. Pensé que era difícil
12. Era estéticamente agradable
13. Olvidé todo a mi alrededor
14. Me sentí bien
15. Fui bueno en el juego
16. Me sentí aburrido

17. Me sentí exitoso
18. Me sentí imaginativo
19. Sentí que podía explorar las cosas
20. Lo disfruté
21. Fui rápido en alcanzar los objetivos del juego
22. Me sentí molesto
23. Me sentí presionado
24. Me sentí irritable
25. Perdí la noción del tiempo
26. Me sentí desafiado
27. Me pareció impresionante
28. Estuve profundamente concentrado en el juego
29. Me sentí frustrado
30. Se sintió como una experiencia enriquecedora
31. Perdí la conexión con el mundo exterior
32. Sentí la presión del tiempo
33. Tuve que esforzarme mucho

## Módulo Post Juego

Para cada uno de los siguiente puntos indique cómo se sintió **DESPUÉS** del juego, usando la siguiente escala:

| Para nada | Un poco | Más o menos | Bastante | Totalmente |
|---|---|---|---|---|
| 0 | 1 | 2 | 3 | 4 |

1. Me sentí revitalizado
2. Me sentí mal
3. Me costó mucho volver a la realidad
4. Me sentí culpable
5. Se sintió como una victoria
6. Me pareció una pérdida de tiempo
7. Me sentí lleno de energía
8. Me sentí satisfecho
9. Me sentí desorientado
10. Me sentí exhausto
11. Sentí que podía haber hecho cosas más útiles
12. Me sentí empoderado
13. Me sentí cansado

14. Me sentí arrepentido
15. Me sentí avergonzado
16. Me sentí orgulloso
17. Tuve la sensación de haber regresado de un viaje