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

VIDEOJUEGO EN REALIDAD VIRTUAL PARA LA REGULACIÓN DE ESTADOS DE SOLEDAD EN ESTUDIANTES UNIVERSITARIOS

MEMORIA PARA OPTAR AL TÍTULO DE
INGENIERA CIVIL EN COMPUTACIÓN

VALENTINA SOFÍA LIBERONA ZÚÑIGA

PROFESOR GUÍA:
FRANCISCO GUTIÉRREZ FIGUEROA

MIEMBROS DE LA COMISIÓN:
BENJAMÍN BUSTOS CÁRDENAS
SANDRA DE LA FUENTE GONZÁLEZ

SANTIAGO DE CHILE
2021

Resumen

La realidad virtual (VR) es un mercado en constante crecimiento que proporciona una experiencia inmersiva e interactiva. Por otro lado, puede proporcionar sensación de compañía de agentes virtuales a quienes viven en entornos confinados o aislados y con sensación de soledad, lo cual se cree que puede reducir la percepción de soledad, el estrés y mejorar el estado de ánimo.

El presente trabajo establece una base para el futuro desarrollo de videojuegos en realidad virtual con diseño de interacciones dinámicas centradas en mejorar la salud mental de los usuarios. En primer lugar, se realizó una revisión de literatura que integra los resultados de la investigación sobre el área de la *Computación Centrada en las Personas* en el ámbito de la salud mental y disminución de soledad, y por otro lado, en videojuegos de simulación de vida y de cuidado de mascotas virtuales. Por otra parte, se mencionó algunos de los diversos usos que se han explorado con respecto a la tecnología emergente de VR.

En el presente trabajo de título se desarrolló un juego en realidad virtual inmersivo dedicado a la simulación de vida en un ambiente aislado donde el principal objetivo consiste en el cuidado y acompañamiento de una mascota virtual. Para lograr esto, se escogió un visor de VR, el cual se consideró que tiene las mejores características para el contexto de este trabajo y un motor de desarrollo de videojuegos que fue compatible con el visor escogido. En cuanto al desarrollo, se escogieron los *assets* del juego, los cuales algunos fueron obtenidos de fuentes externas y otros fueron creados a través de software de modelado 3D y texturizado. Posteriormente, se implementaron las interacciones básicas en VR como lo son navegar por el ambiente, manipular objetos y usar la interfaz de usuario. Luego, se implementó la inteligencia artificial de la mascota virtual, lo cual incluyó una máquina de estados, el controlador de las animaciones y el guardado de parámetros internos. Luego, se implementó el funcionamiento del mundo dentro del juego, incluyendo el paso del tiempo, los cambios en el ambiente y el sistema de guardado de datos de la partida. Por otro lado, se realizaron las optimizaciones necesarias en cuanto al rendimiento de la aplicación para cumplir con los estándares de VR proporcionados por la guía de desarrollo del visor escogido. Además, se realizó una prueba de concepto con usuarios a baja escala, para poner a prueba el juego desarrollado.

Por último, se discutió sobre los comentarios obtenidos por los usuarios en las pruebas y lo que podría haberse hecho mejor, y qué otras mecánicas podrían agregarse al videojuego a futuro. Finalmente, se concluye con respecto al trabajo presentado que se pudo cumplir con todos los objetivos propuestos. En particular, se implementó un videojuego para un dispositivo VR cuya funcionalidad es cuidar y proteger a una mascota virtual.

Agradecimientos

En primer lugar quiero agradecer a mi profesor guía, quien con sus conocimientos y apoyo me guió a través de cada una de las etapas de esta memoria.

Asimismo, agradezco a mis compañeros y amigos del DCC, especialmente a Alejandro Q., Elías Z. y Gabriel C., por su apoyo y ayuda incondicional y sus valiosas sugerencias en momentos de duda.

Finalmente, a mi familia, en particular mi madre y padre no biológico por ser mi pilar fundamental y haberme apoyado incondicionalmente en las diferentes etapas de este proceso universitario.

Tabla de Contenido

Índice de Ilustraciones

Capítulo 1

Introducción

La soledad es un factor de riesgo único conocido por contribuir al desarrollo de trastornos en la salud mental como la depresión o la ansiedad [22, 24, 25]. La conciencia pública y gran parte de la investigación científica sobre la soledad relacionada a la salud mental se encuentra principalmente enfocada en personas mayores de 55 años. Sin embargo, al encuestar a todos los grupos de edad, las personas más jóvenes (de 16 a 24 años) aparecen como de especial riesgo [13, 32].

Cabe destacar que este período de la vida es el de mayor riesgo para la aparición de problemas de salud mental. En particular, esto incluye enfrentar mayor autonomía, asumir mayores exigencias y responsabilidades académicas, y responder a expectativas personales y familiares junto con la necesidad de adaptación a un nuevo contexto de vida [18]. En el caso de nuestro país, la soledad aún no es tomada como un índice para determinar la salud mental en gente joven. Sin embargo, ha habido estudios con respecto a la sintomatología depresiva como lo son los datos presentados en la Novena Encuesta Nacional de Juventud 2018 [28]. Además, se han realizado estudios de salud mental en muestras de estudiantes universitarios [4, 36], revelando que un 27 % de los estudiantes cumple con los criterios diagnósticos para una depresión, un 10,4 % estaría cursando con un trastorno bipolar y un 5,3 % de los estudiantes presenta un riesgo desde moderado a severo de cometer suicidio.

Sin embargo, para casi la mitad de los jóvenes del país es “Nada” o “Poco posible” costear por un periodo prolongado de tiempo consultas médicas con un psicólogo o psiquiatra (43,7 %), medicamentos para un tratamiento psicológico o psiquiátrico (45,2 %), o exámenes solicitados por un psiquiatra (46 %) [28]. Es por esta razón que se han explorado diversos tratamientos para prevenir o tratar dichos problemas de salud mental. A modo de ejemplo, se encuentran los animales como soportes emocionales [35]. Existen fuertes vínculos emocionales entre los dueños y sus mascotas, y el afecto compartido entre una persona y su mascota indica ser beneficioso para su salud percibida y su estado psicológico. Las intervenciones asistidas por animales, como interacción con animales de terapia, pueden complementar los tratamientos médicos tradicionales de los problemas de salud mental [31].

1.1. Problema abordado

La aplicación de las tecnologías emergentes es prometedora para abordar potencialmente diferentes dimensiones del desafío mundial de la salud mental [8]. Sin embargo, entre dichas tecnologías existe una vasta gama de posibilidades que se pueden estudiar en pos de ayudar a sus usuarios a sentirse menos solos, y en consecuencia, reducir sus niveles de estrés, ansiedad y/o depresión. Entre estas opciones se encuentra el uso de los videojuegos, los cuales proporcionan varios efectos positivos y beneficiosos para la salud mental [21]. Los beneficios de los videojuegos han sido estudiados en cuatro ámbitos principales: cognitivo, motivacional, emocional y social, a través de características tales como la interactividad, la estética y la narrativa [21]. Sobre todo los juegos de simulación de vida, ya que estos no solo acercan a los usuarios a la tecnología conocida, sino que también les ofrecen una inmersión total en un mundo sintético. Entre los muchos juegos de simulación que han tenido éxito, el género de simulación de mascotas digitales ha demostrado su popularidad en este segmento de población.

La comunidad de Interacción Humano-Computador (HCI, por sus siglas en inglés) recientemente ha manifestado un gran interés en el estudio y diseño de experiencias interactivas para promover mejores estados de salud mental. Al respecto, Birk et al. [10] discuten acerca de las oportunidades y desafíos de utilizar videojuegos como mecanismos para una mejor salud mental. En dicho artículo los autores proponen una serie de propuestas basadas en el uso de tecnologías e ideas innovadoras como la aplicación de un visor de Realidad Virtual (VR, por sus siglas en inglés) en adultos mayores, o el ajuste dinámico de dificultad de juegos basado en modelos de usuario.

En las tecnologías propuestas por Birk et al. [10] se encuentra la realidad virtual, la cual es relativamente reciente y novedosa, por lo que aún no se ha explorado extensamente sus posibles impactos y aplicaciones en el campo de la salud mental. Además, esta tecnología es cada vez más accesible, inmersiva y fácil de usar [11, 2], lo cual hace posible explorar las contribuciones que puede ofrecer en otros campos como lo son su aplicación en los videojuegos para la salud mental. La reciente tecnología emergente con aspecto de VR está muy orientada a la investigación, así como al éxito en dispositivos comerciales. La cualidad inmersiva e interactiva que ofrecen los visores de realidad virtual tienen el potencial de generar un impacto y beneficio en la disminución de percepción de soledad en el usuario, ofreciendo un contacto directo y estrecho con el entorno del juego y sus posibles entes [12].

Dado lo anterior, el desarrollo de un videojuego de VR que permita disminuir la percepción de soledad no es una tarea directa, debido a la escasa literatura actualmente disponible y aplicación en estos campos.

1.2. Solución desarrollada

En el presente trabajo de título se diseñó y desarrolló un prototipo de videojuego de simulación de mascota en realidad virtual interactiva, que permite a sus usuarios involucrarse

en un entorno virtual que responda a su estado emocional, basándose en el uso de estímulos visuales y auditivos. Esto se realizó tomando a la población de estudiantes universitarios de entre 20 y 25 años como usuarios objetivo con el fin de ayudar en la regulación de su estado de salud mental.

Se decidió implementar dicho videojuego en VR dado que al usarse en combinación la capacidad de seguimiento espacial y de orientación, es posible recrear un mundo virtual que parece rodear al usuario en las tres dimensiones. Esto le da al usuario una sensación inmersiva de espacio presencial y las expectativas de que pueden interactuar con el mundo, respondiendo este último de una manera percibida como realista por el jugador.

La realidad virtual consiste en simular un mundo físico mediante objetos virtuales y entornos sintéticos que pueden ser interactivos. El propósito de su construcción es aislar a los usuarios del mundo real y hacerles creer en el mundo virtual mediante la inmersión. Para mejorar la inmersión de la realidad virtual, se utiliza un visor montado en la cabeza (HMD, del inglés *head-mounted display*) el cual proporciona una perspectiva en primera persona. Los HMD simulan la visión 3D generando una secuencia de imágenes del entorno virtual siguiendo el movimiento de la cabeza del usuario. Otras características de inmersión son la navegación y la interacción con el entorno virtual.

Se propuso analizar dos alternativas para desarrollar la mecánica principal del juego. Una opción era realizar un trabajo con un enfoque de tomar un rol dentro de una comunidad, permitiendo al usuario interrelacionarse con otros que utilicen el mismo medio, como es el ejemplo de VRChat (VRChat, 2017). La otra era la de relacionarse con una mascota virtual, como indica el trabajo de Young et al. [52], utilizando aspectos de simulación de cría de mascotas digitales. Finalmente, se optó por trabajar esta última opción.

De este modo, la mascota virtual, actuando como personaje principal del juego, requiere de la ayuda del usuario para cuidarla, nutrirla y entretenerla, lo cual apela a que el usuario sienta que es requerido y necesitado, disminuyendo así su percepción de soledad. Además, se aplicaron aspectos del género de videojuegos de simulación de vida, incentivando a que el usuario también interactúe con su entorno, a través de interacciones con su entorno tanto del espacio virtual como de la mascota.

El ambiente del juego fue uno de los principales aspectos considerados en el diseño para hacer sentir al usuario acogido, con tal de proporcionarle de entretención y un espacio de distracción de sus problemas externos y sentimiento de soledad. Para esto, el nivel se conformó de dos elementos principales: una casa en donde viven el jugador junto a su mascota y una pequeña terraza exterior desde la cual se puede apreciar el paisaje.

Los aspectos más importantes que definen la atmósfera por la que se ve rodeada el jugador son la música, el clima y el aspecto visual del espacio físico en donde se encuentra. Para proporcionar un entorno realista, el entorno virtual desarrollado cuenta con un sistema de control de ciclo de día y noche el cual modifica la luz ambiental y el color del cielo. En la Figura 1.1 se presenta una parte del entorno exterior durante un atardecer dentro del videojuego diseñado en este trabajo de título.

Por otro lado, se utilizó un modelo de usuario afectivo, el cual incluye información sobre el

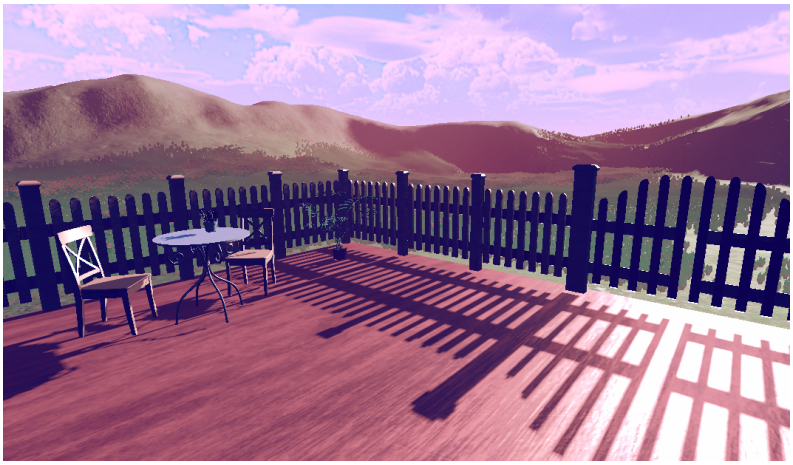


Figura 1.1: Ambiente exterior, atardecer

estado emocional actual del jugador. Esto es, para que tanto el ambiente del entorno virtual como el comportamiento de la mascota se adapten al estado de ánimo registrado por el jugador, y así este dinamismo en los elementos del juego ayuden a mejorar su estado anímico y aliviar su sensación de soledad. Para lograr esto se utilizó un sistema de auto reporte del usuario dentro del juego, y un sistema de estados que regula las transiciones del entorno y del comportamiento de la mascota virtual.

La mayoría de los dispositivos de VR existentes cuentan con soporte en varios motores de desarrollo de videojuegos, los cuales a menudo contienen un *SDK* (kit de desarrollo de software) de realidad virtual que permite a los desarrolladores diseñar, construir y probar sus juegos. Estos motores ofrecen herramientas para crear y editar experiencias en 3D totalmente inmersivas. Los motores de juegos de VR se engloban dentro de los motores de juegos tradicionales, pero además admiten sistemas operativos y hardware de VR, ya sea directamente o a través de una API. Entre ellos se encuentra el motor de videojuegos Unity, el cual se eligió para el desarrollo de este trabajo de título en conjunto con un visor de VR Oculus Quest que ofrece compatibilidad con el motor.

En términos generales, un visor de realidad virtual o HMD es un dispositivo montado en la cabeza que proporciona realidad virtual al usuario. El HMD esta compuesto por una pantalla estereoscópica montada en la cabeza (que proporciona imágenes separadas para cada ojo), sonido estéreo y sensores de seguimiento del movimiento de la cabeza. Algunos cascos de VR también tienen sensores de seguimiento ocular y mandos de juego.

Con el objetivo de escoger el HMD para el desarrollo del videojuego en el contexto del presente trabajo de título fue necesario evaluar detenidamente el tipo de visor a utilizar. Así, se tuvo en consideración el que no debe ser considerado como una molestia por el jugador, uno que debe tener soporte con el motor de juegos y que pueda ser accesible en el mercado de masas. Para ello, se revisaron las siguientes opciones: *Oculus Go*, *Oculus Rift*, *Oculus Quest*, *HTC Vive*, *Valve*, entre otros. Sin embargo, las dos opciones más importantes y que más inmersión ofrecen de todas las experiencias VR dentro de la industria del videojuego son los HMD de *HTC Vive* y *Oculus*.

1.3. Objetivos

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

Objetivo General

Diseñar e implementar un prototipo de videojuego de VR interactivo que permita al usuario involucrarse emocionalmente en un entorno virtual basándose en múltiples sentidos, a través de imágenes, texturas, audio y tacto.

Objetivos Específicos

Para lograr el objetivo general expuesto en este trabajo de título, se definieron los siguientes objetivos específicos:

1. Seleccionar las herramientas de desarrollo para VR más apropiadas en el contexto del presente trabajo. Esto es, escoger un visor de VR que funcione de manera independiente de un computador para ser configurado fácilmente y que tenga sensores de movimiento integrados junto con controles que simulen el movimiento de las manos de manera realista para lograr la sensación de inmersión dentro del juego. Además, debe ser asequible dentro del mercado de dispositivos VR. Los costos dependen de las marcas y los productos, dando un rango de precios de entre $199 dólares con el *Oculus Go* y $799 dólares con el *HTC Vive* [40]. Se logró cumplir con dichos requerimientos escogiendo el visor *Oculus Quest*.
2. Diseñar una experiencia de juego inmersiva que sea considerada como entretenida y aceptable en términos de experiencia de usuarios. Se logró lo anterior implementando de manera correcta tanto las interacciones como el entorno en VR.
3. Dar respuesta a las interacciones del jugador con su mascota y a las necesidades que esta tiene. Esto se logró utilizando un sistema de iconos emergentes de la mascota los cuales muestran el estado en que se encuentran sus necesidades en conjunto con un icono de corazón el cual aumenta su contenido a medida que el usuario interactúa con esta.
4. Proporcionar un entorno realista y un ambiente agradable para el jugador. Este objetivo se logró utilizando modelos 3D de objetos de decoración de casa y materiales PBR (Materiales de renderizado basados en la física), lo cual permite calcular la luz de una escena en 3D de manera realista. Además, se utilizaron técnicas de iluminación de escena para proporcionar reflejos realistas en materiales como vidrios y metales. Por otro lado, se implementó un ciclo de tiempo que cambia los colores del ambiente del día a la noche.
5. Desarrollar un prototipo de juego que cumpla con los criterios de jugabilidad y satisfacción del usuario que asegure la calidad de software. Para validar esto se realizó una

prueba de concepto con una muestra de potenciales usuarios finales.

1.4. Metodología

El proyecto se elaboró mediante la metodología de desarrollo ágil, la cual se aplicó presentando un nuevo prototipo cada semana durante las reuniones de avance y validación con el profesor guía, seguido de comentarios y planificación para la semana siguiente. El breve ciclo de desarrollo y retroalimentación tenía como propósito permitir cambios rápidos a la aplicación durante toda la fase de desarrollo. El proyecto se llevó a cabo de esta forma dada la naturaleza del proyecto y la falta de conocimientos previos. Es por esta razón que, a medida que se adquiría más experiencia en el desarrollo con esta tecnología, los requisitos iban mutando hasta llegar a configurar un prototipo que pudiese ser considerado como producto mínimo viable para poder ser evaluado por usuarios.

A continuación se detalla un listado de los pasos que se debieron seguir en pos de concretar el desarrollo del proyecto del presente trabajo de título:

1. Se realizó un análisis comparativo entre los distintos dispositivos de VR existentes y para escoger el más adecuado para el videojuego expuesto. Este análisis consideró la viabilidad de las distintas alternativas basándose en las características mencionadas anteriormente. Además, se definieron las herramientas de desarrollo de software a utilizar considerando que el motor debía tener soporte para la tecnología escogida. Esta tarea apuntó a cumplir el objetivo específico(1).
2. Se realizó una investigación previa de los distintos motores de desarrollo de videojuego 3D y modelado 3D existentes y compatibles con el hardware de VR escogidos para determinar las herramientas de desarrollo a utilizar en el presente trabajo.
3. Se habilitó el soporte de VR *Oculus Quest* en el motor de desarrollo de videojuegos *Unity* a través del framework *XR Plugin Management*. Esto permitió realizar la conexión con el visor de Oculus y proporcionó herramientas para facilitar la implementación de las interacciones básicas en VR.
4. Para la fase de diseño del videojuego se realizaron las siguientes tareas:
    - Se diseño el flujo del juego, el cual describe el funcionamiento estático y dinámico, la interacción con los usuarios y los diferentes estados que atraviesa el videojuego como software.
    - Se definieron las actividades *core* del jugador, es decir, sus restricciones de movimiento, y actividades y acciones que puede realizar durante el juego.
    - Se definió el sistema de navegación que usa el jugador para explorar el espacio virtual. Para esto, se exploró dentro de los distintos tipos de sistemas de *locomotion* frecuentemente utilizados en VR y el sistema de inputs.
    - Se definió la ubicación de la interfaz de usuario determinando cómo se posiciona dentro del mundo virtual, lo cual incluyó determinar la distancia a la que se encuentra del jugador y su resolución. Además, se definió el tamaño de los distintos elementos que proporcionan la interfaz de usuario (UI).

- Se plantearon los aspectos fundamentales que conforman el videojuego, lo cual implicó diseñar el nivel y las mecánicas de jugabilidad e interacción con la mascota y el ambiente.

5. Se realizó un estudio previo para familiarizarse con las herramientas de desarrollo. Para esto se utilizó la documentación oficial de Unity y de su integración con el hardware de VR en dicho programa.
6. Se configuró e implementó un sistema en VR básico dentro de Unity, lo cual incluyó recepción de inputs, presencia de manos, mecánica de movimiento, interacciones con objetos e interacciones con la UI.
7. Se creó el modelo 3D e implementaron las mecánicas de la mascota virtual, incluyendo el desarrollo de:
   - La creación del esqueleto y animaciones del modelo 3D de la mascota.
   - La implementación de un sistema que controle sus niveles de nutrición, energía y felicidad.
   - Las mecánicas de interacción. Esto es, que el usuario pueda tocar e interactuar con la mascota.
8. Se crearon y desarrollaron los distintos elementos que constituyen el ambiente del juego. Es decir, el diseño del nivel, los elementos visuales y audiovisuales, y los elementos ambientales, tales como los colores, la iluminación y efectos gráficos.
9. Se realizó la validación del juego a nivel de testeo del software, lo cual implica la corrección de *bugs*, la realización de pruebas de funcionalidad verificando que cada elemento del juego se comporte de la manera esperada, el cumplimiento de los tiempos de las tareas específicas del programa y la ejecución de pruebas de rendimiento.
10. Finalmente, se concretó la etapa de validación del juego a nivel de usuario. Midiendo esto, a modo general, se verificó de manera preliminar la usabilidad y jugabilidad del software a través de una prueba de concepto utilizando un cuestionario estándar de jugabilidad.

1.5. Estructura del documento

El resto de este documento se estructura de la forma descrita a continuación. En el Capítulo 2 se presenta y discute la literatura relacionada que sustenta el trabajo ejecutado en el presente trabajo de título. El Capítulo 3 presenta las decisiones de diseño e ingeniería tomadas respecto a: (1) Los requerimientos que debe cumplir el juego; (2) la selección de dispositivo de VR y software de desarrollo de videojuegos; (3) el diseño de las interacciones, la inteligencia artificial (AI) de la mascota y del flujo del juego; y (4) los elementos visuales y estéticos. Además, expone el diseño de los elementos del juego desarrollado. A continuación, el Capítulo 4 presenta la implementación de los distintos elementos que componen el juego. 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 sobre el trabajo futuro.

Capítulo 2

Trabajo Relacionado

En la literatura académica se reporta un gran interés en ocupar la computación en pos de mejorar la salud mental de las personas. Por ejemplo, Torous et al. [42] discuten acerca de la intersección de HCI, la innovación tecnológica y la salud mental, abarcando opiniones de comunidades diversas, tales como provenientes del área de psicología, científicos de datos y computación. En dicho simposio se presentó una serie de posters y papers que abordan el tema desde distintos enfoques tecnológicos.

Asimismo, en la literatura existen varios intentos de abordar el apoyo a las interacciones de salud mental a través de los videojuegos. Al respecto, Denisevicz et al. [17] desarrollaron *Mono no Aware*, un juego para un solo jugador que fue desarrollado para generar empatía hacia quienes sufren de efectos mentales y físicos provocados por la ansiedad y la depresión. El núcleo del juego describe la experiencia de tener ansiedad y depresión, al tiempo que introduce técnicas de terapia de procesos comunes como estrategias de afrontamiento. Sin embargo, la mayoría de estos juegos son *juegos serios* [1] que se centran únicamente en estas enfermedades psicológicas en particular. En Dekker et al. [15] realizaron un mapeo sistemático sobre la literatura existente en este dominio de aplicación, donde identificaron 20 juegos serios desarrollados para prevenir, tratar o complementar las terapias existentes para la ansiedad y/o la depresión. La mitad de estos juegos se desarrollaron con el aporte de los usuarios finales previstos; es decir, participaron en el diseño del juego, ya sea como informantes o en funciones de diseño completamente participativas, y menos de la mitad de los juegos (45 %) incluyeron a los usuarios sólo en la fase de prueba.

Por otro lado, se han explorado alternativas tecnológicas para tratar la soledad. Por ejemplo, Baecker et al. [6] abordaron el problema de aislamiento social. En dicha investigación, como en la mayoría de las que abordan el tema de la soledad, se presta especial atención a los adultos mayores en comunidades de jubilados y en los entornos de cuidados a largo plazo o asilos. Los autores presentan las implicaciones de diseño de la tecnología para permitir y favorecer conexiones sociales. Otros acercamientos tecnológicos en cuanto a soluciones para la soledad son los llamados robots de compañía. Al respecto, Baecker et al. [5] ejemplifican este concepto y se centran en los robots que ayudan a los adultos mayores sanos que viven en aislamiento social y soledad, aprovechando las técnicas de la terapia para proporcionar apoyo emocional a través de la capacidad básica de conversación.

En la literatura también se encuentran publicaciones relativas a VR como herramienta tecnológica para abordar temas como la salud mental y la soledad. Por ejemplo, Antunes et al. [3] sostienen que los juegos en un entorno de VR estimulan la práctica de las habilidades comunicativas y cognitivas y también pueden aportar beneficios a los adultos mayores, por lo que se propusieron evaluar los efectos de la práctica de juegos de VR en la percepción de soledad entre los estudiantes de un centro de referencia para ancianos. Con respecto a los juegos en realidad virtual, Liszio et al. [33] sostienen que debido a la necesidad de aislar los sentidos humanos de las señales del mundo real, la VR ha sido criticada repetidamente como una experiencia solitaria y antisocial y llegan a la conclusión de que añadir un componente social a un juego de VR puede mejorar la experiencia del jugador al reducir la soledad. Sin embargo, este último requiere un examen minucioso del diseño y la aplicación, tanto de la mecánica de la interacción social como de la propia entidad social.

La simulación de mascotas virtuales tienen una larga trayectoria en el ámbito del entretenimiento. Por ejemplo, el *Tamagotchi* que se hizo popular a mediados de los años noventa (Bandai, 1996). Existen juegos de cuidado de mascotas virtuales populares tanto en consolas como por ejemplo *Nintendogs* (Nintendo, 2011), como en dispositivos Android o IOS como *Pou* (Paul Salameh, 2012) o en PC como *Pet Society* (Electronic Arts, 2018) y más recientemente *The Sims 4: Cats & Dogs* (Maxis Redwood Shores, Electronic Arts, 2017). Estos tipos de software comparten cierta similitud con el proyecto dado que el jugador es capaz de interactuar con modelos digitales de animales u otras criaturas y verlas reaccionar en consecuencia. La razón por la que es necesario construir un nuevo software para proporcionar una simulación de mascota en VR es dado que los programas existentes son videojuegos creados en primer lugar con fines de entretenimiento y no cumplen el rol de presencia, acompañamiento y entorno relajante básicos necesarios para ayudar en el tratamiento de problemas de salud mental.

En resumen, la comunidad de investigadores y profesionales en HCI ha estado planteando por años el uso de la tecnología y, en particular, los videojuegos para abordar problemas de salud mental. Sin embargo, la mayoría de los trabajos relacionados son sólo estudios de los efectos e influencias de dichas tecnologías en los aspectos de salud mental. A excepción de los trabajos discutidos, se encuentran algunos juegos como los mencionados anteriormente, diseñados específicamente para ayudar al tratamiento de la ansiedad y depresión. No obstante, la literatura es extensa sólo en lo que respecta a juegos serios. Además, el tratamiento de la soledad a través de videojuegos es un área poco explorada, existiendo únicamente algunos ejemplos de aplicaciones o juegos casuales de VR para adultos mayores.

Capítulo 3

Diseño del Juego

Esta sección contiene los detalles del proceso de diseño. Comienza exponiendo los requisitos derivados de los objetivos mencionados en la introducción, analiza las herramientas y el proceso de diseño, y finaliza con una descripción de la aplicación.

3.1. Requerimientos

Como se indica en el objetivo, el software tiene dos requisitos fundamentales. El primero es proporcionar un entorno y mascota virtual realista compatible con *hadware* VR, y el segundo una forma de interactuar con él. Se creó una lista completa de requisitos a partir de los dos más básicos ampliándolos con más detalle, lo que dio como resultado lo siguiente:

3.1.1. Requerimientos funcionales

1. El juego debe ser implementado con C# y desarrollado en el motor de Unity.
2. El juego debe verse a través del visor de VR Oculus Quest y debe controlarse con el mando *Touch controller* de Oculus Quest.
3. El juego debe jugarse en el sistema operativo Android con Android 4.4 'KitKat' (nivel de API 19) o superior compatible con VR.
4. El juego debe tener un nivel donde se desarrolla el contexto de juego. El juego debe transmitir sensación de presencia física y acompañamiento.
5. El juego debe contar con música y efectos de sonido. Sin embargo, no son cruciales para el juego.
6. El juego debe ser de realidad virtual, de simulación de vida y cría de mascota virtual en primera persona.
7. El software debe proporcionar información que refleje los cambios en el estado de la simulación de la mascota virtual.

8. El software debe facilitar la interacción con el usuario, recibiendo sus acciones y actualizando el estado de la simulación.

3.1.2. Requerimientos no funcionales

1. La interfaz de usuario debe ser clara y compatible con el entorno en VR. La interfaz de usuario debe controlarse a través de *Touch controller* de Oculus Quest.
2. El juego debe utilizar el *plug-in* de Unity llamado *XR SDK* que permite a los proveedores de *XR* integrarse con el motor de Unity y ofrecer soporte para las interacciones en VR.
3. El software no debe mostrar ningún problema de rendimiento evidente, como una baja velocidad de fotogramas o falta de respuesta de la entrada.
4. El juego debe tener gráficos 3D de alta calidad para otorgar una experiencia realista y agradable al usuario.
5. El software debe ser legal, y todos los activos que se utilicen en él deben tener los permisos adecuados.

3.2. Herramientas

Se utilizaron algunas herramientas externas para ayudar al desarrollo. En primer lugar, para crear una mascota visualmente atractiva y amigable, se decidió que crear un modelo tridimensional propio sería lo más apropiado. Por esta razón, se requirió un software específico que ayude a crear y animar dicho modelo para permitir que se mueva de la manera deseada. Se eligió Blender (Blender 2.92, 2021) como solución de modelado y animación 3D, ya que cuenta con varios recursos de aprendizaje de fácil acceso, además de estar disponible de forma gratuita debido a su naturaleza de código abierto. Por otro lado, para preparar las texturas de los *assets* utilizados en el juego se escogió Substance Painter (Substance Painter 2021.1.1 de Allegorithmic, 2003) que al ser un software de texturizado dispone de una alta variedad de texturas, para su uso se obtuvo una licencia de estudiante.

Para permitir que la mascota se mueva correctamente y gestionar el entorno 3D en que esta existirá, así como otros aspectos de la simulación que no están directamente relacionados con el comportamiento de la mascota, como la física, la cámara y la iluminación, se escogió el motor de videojuegos en 3D Unity (Unity 2020.2.5f1, 2021). Además, se proporcionaron algunas utilidades básicas como bibliotecas para ayudar a la interacción con el usuario y proporcionar archivos 3D de libre uso. El motor Unity se utiliza habitualmente en conjunto con Blender por lo que los problemas de compatibilidad se encuentran en su mayoría resueltos, proporcionando así una experiencia de trabajo más fluida.

En cuanto al motor, como alternativa se consideró Unreal Engine 4 (EpicGames, 2015), la cual es una opción profesional y que también está disponible de forma gratuita. La principal razón para elegir Unity sobre Unreal fue debido a la comparación de las capacidades y la dificultad de uso, y se consideró que Unity era adecuado y suficiente para la realización del

proyecto y que además presentaba una curva de aprendizaje menor. Esto último es importante debido a la limitada escala de tiempo del proyecto. Otro aspecto importante que se tuvo en cuenta es la poca optimización en los proyectos orientados a dispositivos móviles en Unreal Engine la cual es fundamental para un videojuego en VR.

Posteriormente, fue necesario adquirir un visor de VR para poder desarrollar el juego y para poder realizar las pruebas. Por tanto, se realizó una comparación entre los HMD que han tenido mayor éxito en el mercado. *Oculus* (Palmer Luckey, 2012) ofrece tres tipos de HMD: *Rift*, *Go* y *Quest*, con frecuencias de actualización de 80Hz, 60Hz y 72Hz respectivamente. El *Oculus Rift* dispone de *6DoF* (6 grados de libertad) y requiere de un PC especial para videojuegos y sistema operativo Windows 10 para funcionar, al cual debe conectarse mediante un cable, posee dos controles táctiles que representan las manos del usuario en la VR y 5 cámaras incorporadas. Dado que utiliza la potencia de las tarjetas gráficas de PC, este permite generar experiencias de realismo gráfico muy superiores a las de *Go* o *Quest*. *Oculus Go* por su lado es un HMD básico con *3DoF* (3 grados de libertad) autónomo, lo cual significa que sólo tiene seguimiento de la rotación, no de la posición. Posee un único controlador que es esencialmente un puntero láser rotativo. Debido a esto, el visor sólo puede ser utilizado correctamente cuando se está sentado en posición estacionaria, y además requiere conexión al PC. *Oculus Quest* es también un HMD autónomo con *6DoF*, que al igual que *Oculus Go*, no se conecta a un PC. No obstante, se diferencia en que tiene un seguimiento posicional a gran escala y utiliza controles táctiles similares a los de *Oculus Rift*, además de tener 4 cámaras incorporadas.

Por otro lado, *HTC* (Cher Wang, Peter Chou, 1997) ofrece dos tipos de HMD: *HTC Vive Cosmos* y *HTC Vive Pro*. Todos los visores de *Vive* requieren de un PC potente al igual que *Oculus Rift* con conexión mediante un cable, y además poseen frecuencia de actualización de 90Hz. El HMD *Vive* original contiene 32 sensores de seguimiento de movimiento en toda su superficie y una única cámara óptica ubicada en el centro inferior del panel. Tiene dos controladores táctiles con sensores de movimiento y posicionamiento. Además, depende de dos estaciones base para determinar las posiciones de los visores y los controles lo que permite experimentar la VR en toda la habitación. *Vive Pro* en cambio incorpora dos cámaras para una visión estereoscópica de seguimiento en 3D y utiliza las mismas dos estaciones base externas y los mismos dos controladores de movimiento que el *Vive* original. *Vive Cosmos* por su lado cuenta con 6 cámaras que rastrean la posición de los HMD monitoreando sus alrededores, eliminando completamente la necesidad de estaciones base o sensores externos.

Por último, *Valve* (Gabe Newell, Mike Harrington, 1996) ofrece el HMD *Valve Index*. El visor se puede configurar con una frecuencia de actualización de hasta 144 Hz, el sistema de audio del casco viene dado por dos auriculares *supra-aurals* abiertos que emulan la sensación de sonido procedente del exterior para proporcionar una inmersión más natural. Además, permite detectar movimiento en *6DoF*, y contienen un gran número de sensores que hacen un seguimiento del movimiento de cada uno de los dedos del usuario. Sin embargo, requiere de dos estaciones base que emiten láser infrarrojos para rastrear el visor y los controladores, además debe ser conectado a un PC con sistema operativo Windows o Linux que tengan instalada la aplicación Steam.

En la Figura 3.1 se muestra una comparación entre los distintos aspectos y precios de

los HMD antes mencionados. Para el contexto del presente trabajo de título se quiere lograr la mayor inmersión y realismo posible, por tanto el HMD seleccionado debía contar con seguimiento 6DoF para poder utilizar tanto la posición del usuario como su orientación. La 6DoF, a diferencia de la 3DoF, permite que el usuario se mueva en todas las direcciones, otorgando más libertad al usuario y haciendo que la experiencia sea lo más real posible. Por otro lado, se consideró que debía ser autónomo para que los jugadores no se vieran limitados por la capacidad de sus PC. Finalmente, teniendo en cuenta estas dos características fundamentales y además considerando un rango de precio más asequible, se optó por utilizar el HMD de *Oculus Quest*.

CIRCUIT STREAM — Decision Guide: HTC vs. Oculus

	OCULUS GO	OCULUS RIFT S	OCULUS QUEST	HTC VIVE PRO	HTC VIVE COSMOS	VALVE INDEX
TYPE	Self-contained	PCVR	Self-contained	PCVR	Wireless PCVR	PCVR
DISPLAY TYPE	LCD 1280 x 1440 2x	LCD 1280 x 1440	OCLED 1600 x 1400 2x	OCLED 1600 x 1400 2x	LCD 1440 x 1600 2x	LCD 1440 x 1600 2x
REFRESH RATE	72Hz	80Hz	72Hz	90Hz	90Hz	80 - 144Hz
FIELD OF VIEW	101°	110°	110°	110°	110°	130°
IPD	61.5 - 65.5 fixed at 63.5mm	0.5 - 65.5 fixed at 63.5mm	56 - 74 adjustable	60 - 73 adjustable	61 - 72 adjustable	58 - 70 adjustable
WEIGHT	468g	500g	571g	470g	645g	500 - 600g
BATTERY LIFE	1.5 - 2hrs	PC-powered	2 - 3 hrs	PC-powered	PC-powered	PC-powered
DEGREE OF FREEDOM	3DOF	6DOF	6DOF	6DOF	6DOF	6DOF
PRICE	$149 (32GB) $199 (64GB)	$399	$399 (64GB) $499 (128GB)	$599 (headset only)	$699	$999 (full VR kit)
RELEASE DATE	May 2018	May 2019	May 2019	January 2018	October 2018	June 2018

Figura 3.1: Tabla de comparación de aspectos de HMD [14]

3.3. Diseño de las interacciones

En esta sección se describen las principales decisiones de diseño tomadas con respecto al manejo de interacciones usuario-sistema, así como componentes que inciden en una mejor experiencia interactiva.

3.3.1. Inmersión

Cuando un sistema de VR intenta imitar el mundo real o persuadir a los usuarios de que lo crean, la inmersión juega un papel fundamental en este aspecto. En cuanto a los factores que influyen en la inmersión, se debieron tener en cuenta las siguientes variables: la percepción, la estereoscopía, el campo de visión, la resolución y la retroalimentación. El factor de percepción fue fundamental para lograr la inmersión, lo cual quiere decir que el entorno virtual se debía sentir lo suficientemente realista y coherente desde el punto de vista del usuario. Por ejemplo, un mundo virtual no coincide con uno realista cuando el juego tiene "lag" o *jitter*, es decir,

un estado en el que las imágenes generadas por ordenador no coinciden con la mirada del usuario real. Para evitar este problema de retardo, se debió otorgar principal importancia a optimizar el tiempo de renderizado, sobre todo haciendo énfasis en el *FrameTime* y los fotogramas por segundo (FPS).

Otros problemas de percepción se refieren a una complejidad de cálculo en la detección de colisiones y la simulación física, lo cual también son necesarias en la construcción de mundos virtuales en 3D. Para esto, también se puso énfasis en que la detección de colisiones entre objetos virtuales fuera coherente entre sí, sin traspasarse unos a otros y que sus movimientos físicos coincidan o al menos sean similares a lo esperable en el mundo real.

3.3.2. Navegación en el ambiente virtual

En cuanto a la navegación, se debió elegir la mecánica de locomoción, es decir, cómo se va a mover el jugador por el mundo. Existen muchos tipos de sistemas de locomoción para implementar en VR, pero de estos los más populares son la locomoción suave y el *teleport*. El primero, permite al jugador mover a su personaje utilizando un *stick* analógico de la misma manera que lo haría en un juego de pantalla plana. Asimismo, suele ir acompañada de una rotación suave o de giros bruscos con el segundo *stick*, de modo que se puede girar todo el cuerpo del personaje para que mire en otra dirección. Sin embargo, para algunas personas la locomoción suave puede provocar mareo, ya que sus ojos ven un movimiento que su cuerpo no está siguiendo físicamente. El segundo, consiste en apuntar con el *stick* la zona a la que el jugador se quiere mover, pulsando un botón para aparecer en la zona señalada. Sin embargo, esta forma de movimiento se interpone en la inmersión del jugador en el mundo.

Teniendo en cuenta estos dos sistemas, se implementaron ambos métodos de locomoción para dejar que el usuario escoja el que más le acomode, dado que es fundamental que el usuario se sienta a gusto dentro del entorno virtual. En la Figura 3.2 y Figura 3.3 se muestran ejemplos de cómo se ve un sistema de *teleport* usando trayectorias curvas.

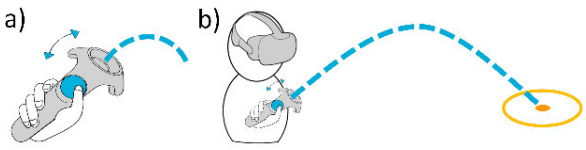


Figura 3.2: El *teleport* de parábola utiliza una visualización en forma de parábola para indicar la posición del objetivo del teletransporte [19]

3.3.3. Interacción con objetos

Desde el punto de vista de la interacción con los objetos virtuales, existen dos métodos principales para agarrar, manipular o accionar objetos: la interacción directa a través de tocar los objetos utilizando las manos virtuales, y la interacción a través de interacción de rayo (conocido como *ray interactor*). Con el fin de proporcionar una experiencia de juego más inmersiva, se eligió implementar el primer sistema de interacción directa para la manipulación

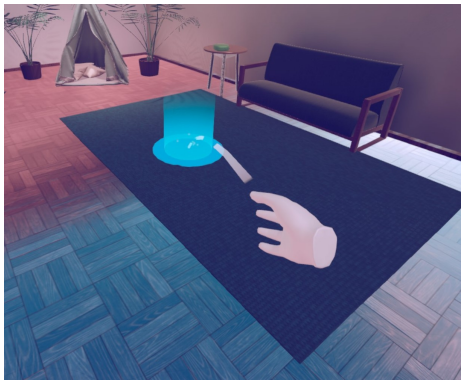


Figura 3.3: Visual del *teleport* en el juego

de objetos en la escena y la interacción con la mascota. Por otro lado, se escogió el segundo sistema de *ray interactor* para seleccionar entre los distintos elementos de la UI.

En conclusión, desde el punto de vista del desarrollo del entorno virtual inmersivo se tuvo como principal objetivo permitir que el usuario se involucre en un mundo sintético como si fuera uno real. Por lo tanto, teniendo en cuenta lo antes mencionado, las mecánicas principales del juego desarrollado consisten en el entorno virtual, el control de la navegación, la interacción y manipulación con los objetos virtuales y la UI. Como se muestra en la Figura 3.4 y Figura 3.5.

Figura 3.4: Interacción directa utilizando la función de agarrar objetos en el juego

3.4. Diseño de la mascota

La mascota virtual fue diseñada para mostrar un comportamiento realista e interactivo. Es decir, la mascota debía emocionarse, comportarse y reaccionar de forma coherente ante las interacciones con el usuario. Para otorgar más realismo al sistema de locomoción de la mascota se utilizaron animaciones del personaje. Además, la interacción del usuario con la mascota se dividió en dos enfoques: el impacto en el estado de la mascota y en su animación reflejada.

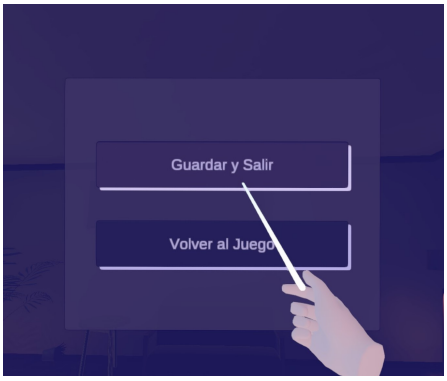


Figura 3.5: Interacción de *Ray interactor* para manipular los elementos de la UI del juego

3.4.1. Aspecto de la mascota

Como mascota del juego, se optó por aprovechar las libertades creativas de un videojuego y se utilizó una criatura de fantasía cuyo aspecto es el de un pequeño dragón. El dragón mascota no cuenta con un género específico para que este quede a la interpretación del usuario. Su diseño es simplificado y sus ojos utilizan un estilo en 2D, tal como se muestra en la Figura 3.6.

Figura 3.6: Aspecto del modelo final del dragón mascota visualizado en Blender

3.4.2. Estados de la mascota

El modelo de la mascota se diseñó para que actúe de forma independiente dentro del ambiente virtual y reaccione a los estímulos externos, lo cual es inspirado en el modelo basado en agentes descrito en Jonker et al. [29]. Este método se eligió porque es adecuado para simulaciones en las que varios agentes tienen que interactuar entre sí, este siendo el caso de la mascota y el usuario. Un agente tiene que mantener información sobre sí mismo y sobre el mundo que le rodea, y gestionar las interacciones entre él mismo y otros agentes o el entorno. Teniendo en cuenta esto, el modelo de la mascota contiene variables que son representaciones de cómo se siente en ese momento, es decir, en qué estado se encuentran sus

niveles de energía, de felicidad y de hambre. Además de esto, la mascota refleja dónde está su atención en ese momento y qué puede percibir del mundo que la rodea.

Estos parámetros internos de la mascota tienen que ser expuestos al usuario para que reaccione adecuadamente ante ella. En este caso, mediante la alteración de su comportamiento a través de animaciones, cambios en sus expresiones y mensajes en forma de iconos emergentes. Para conseguirlo, se incluyó una máquina de estados finitos en el modelo de la mascota para representar los estados de comportamiento en los que se encuentra. Estos estados incluyen acciones básicas cómo navegar y sentarse, y comportamientos más emotivos como animaciones de comer, dormir o de recibir cariño.

Las transiciones entre estados están mediadas por los parámetros internos de la mascota y por las interacciones del usuario. Por ejemplo, si la mascota se encuentra sin recibir input de interacción de parte del usuario se moverá libremente por el entorno, y si alguno de los parámetros antes mencionados llega a cero se dormirá. Si el usuario interactúa directamente con la mascota, esta cambiará al estado correspondiente.

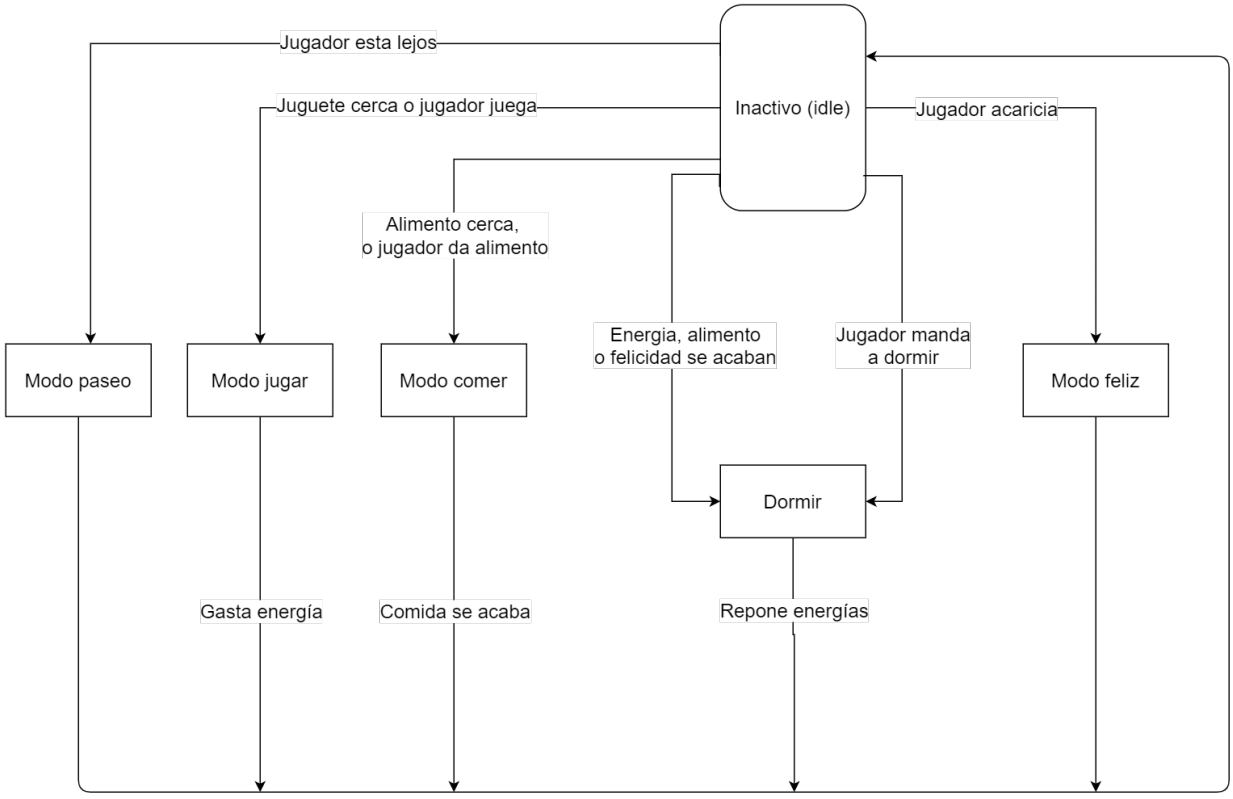


Figura 3.7: Diagrama de estados finitos del comportamiento de la mascota

Idle o de reposo

La mascota se encuentra en este estado cuando no está recibiendo ningún estímulo externo ni se encuentra en movimiento. La animación de reposo muestra a la mascota sentada con un leve movimiento en su cola, cabeza y alas, lo cual tiene como objetivo simular que tiene vida.

Movimiento

Si el jugador no se encuentra cerca de la mascota, esta se mueve cada cierto tiempo a un punto distinto dentro del entorno virtual. Esto tiene el objetivo de mostrar realismo.

Durmiendo

Para que la mascota se quede dormida, el usuario debe señalar su cama y esta debe tener la energía bajo el valor máximo. Asimismo, si la mascota se queda sin energía, o si sus valores de felicidad o alimento se encuentran en el mínimo valor, esta se queda dormida en la posición en la que se encuentra. Para ser despertada, si está en su cama su energía debe subir hasta ser cercana al máximo valor, y en el segundo caso, esta se despierta si el usuario satisface la o las necesidades requeridas. Al dormir, se reproduce la animación correspondiente a este estado y se cambia la expresión a una con los ojos cerrados. En la Figura 3.8 se puede ver a la mascota durmiendo en su cama.

Figura 3.8: La mascota en estado de dormir en su cama en al *Escena*

Comiendo

Para satisfacer el estado de hambre de la mascota, el jugador tiene que alimentar a su mascota cogiendo comida de alguno de los recipientes disponibles en la cocina y acercando la comida directamente a la boca de la mascota. En la Figura 3.9 y Figura 3.10 se muestra una escena de alimentación.

Feliz

La felicidad de la mascota viene dada por la cantidad de cariño que recibe del jugador. Para acariciar a su mascota el jugador debe acercar su mano a alguna parte del cuerpo de ella. En respuesta a esto, la mascota muestra una animación correspondiente y sus ojos muestran un corazón. Esto se puede apreciar en la Figura 3.11 y Figura 3.12.

Figura 3.9: Acción de agarrar un objeto alimento en el juego

Figura 3.10: Escena de alimentar a la mascota

3.4.3. Comunicación

La mascota cuenta con un sistema de comunicación mediante burbujas de diálogo emergentes, las cuales transmiten las necesidades de la mascota a través de imágenes. De este modo, el jugador puede saber qué es lo que la mascota quiere o necesita hacer. Por ejemplo comer, dormir, jugar o ser acariciada. La Figura 3.13 muestra una escena de las cuatro necesidades mostradas a la vez. Por otro lado, la mascota mantiene un icono en forma de corazón flotante sobre su cabeza el cual representa el estado de satisfacción de ella a través de un sistema de llenado el cual actúa como una barra de progreso.

3.5. Flujo del juego

El videojuego consiste en una simulación de la vida con factores de cuidado y nutrición de una mascota virtual. El entorno se presenta al usuario a través de una perspectiva en primera persona simulando un mundo virtual. El jugador vive en una casa modesta, interactúa con los elementos de su entorno y acaricia, alimenta y cuida a su mascota. La mascota emite mensajes en forma de iconos que proporcionan información sobre su estado actual y sus necesidades para facilitar las interacciones y el aprendizaje del juego. El objetivo principal del jugador es el de mantener a su mascota en el mejor estado y cubrir sus necesidades, y a su vez la mascota le proporciona al usuario compañía y le brinda feedback positivo a modo de animaciones y expresiones.

Al iniciar el juego por primera vez el usuario aparece dentro de la casa virtual y ve por

Figura 3.11: Escena de acariciar a la mascota en la cabeza

Figura 3.12: Escena de acariciar a la mascota en estómago

primera vez a su mascota en un estado con todas sus necesidades cubiertas, las cuales se verán en aumento a medida que pase el tiempo. Si no es la primera vez que el usuario entra al juego, al volver a encontrarse con la mascota verá que sus niveles de estado serán equivalentes a como se encontraban en la última sesión, sumándole un factor proporcional al tiempo en que el usuario se encontró desconectado del juego. Dichos valores se describirán con mayor detalle en el capítulo de implementación.

Si el usuario quiere salir del juego, puede abrir un menú del juego y elegir la opción de guardar y salir. Esto cierra el juego y guarda el estado y hora en el cual se cerró. Además, al momento de entrar al juego, el usuario se encuentra con un menú de opciones de estados de ánimo representados por iconos de expresiones: el usuario debe elegir una de las opciones que más lo represente en ese momento para así poder continuar con el juego. En la Figura 3.14 se muestra el diagrama del flujo del juego.

3.5.1. Exploración en el entorno

Para generar la sensación de presencia dentro del entorno virtual, el juego permite al usuario moverse libremente por su entorno doméstico, incluyendo el interior de la casa y la terraza frontal. Para acceder a la parte exterior, el usuario debe interactuar con la puerta principal abriéndola como si fuera una puerta real. El interior de la casa corresponde a un solo ambiente que se divide en una sección para la cocina que está conectada a la sala de estar en la cual se encuentra la casa de la mascota. En cuanto a los objetos del entorno, se dejan a la disposición del usuario para que pueda interactuar con ellos, cambiarlos de lugar,

Figura 3.13: Mensajes de las necesidades de la mascota

permitiéndole sentirse en control de su hogar artificial.

3.5.2. Estado del usuario

Desde la perspectiva de la salud mental, se cree que la atmósfera y el ambiente juegan un papel fundamental para proporcionar un lugar agradable y acogedor, en base a Joosten et al. [30] se tiene que colores más amarillos y anaranjados se asocian a la felicidad. Es por esto que se decidió utilizar técnicas de *post-processing* para ajustar los colores de ambiente de acuerdo al estado de ánimo en el que se encuentre el jugador. Para esto, se le pide a este contestar una pregunta al iniciar el juego, la cual se muestra en la Figura 3.15. Para la elección de los colores se utilizaron de referencia los resultados de Geslin et al. [20].

Figura 3.15: Encuesta inicial de estados de ánimo

3.5.3. Ciclo temporal

El juego cuenta con un medidor interno de las horas del juego y un ciclo de 24 horas para mostrar el paso del tiempo y proporcionar sensación de realismo. De esta forma, el cielo, el color y luz ambiental pasan por diferentes estados dependiendo de la hora del día. Dicho ciclo controla la iluminación, el color del cielo, la rotación de este, el color de ambiente y el color de la niebla.

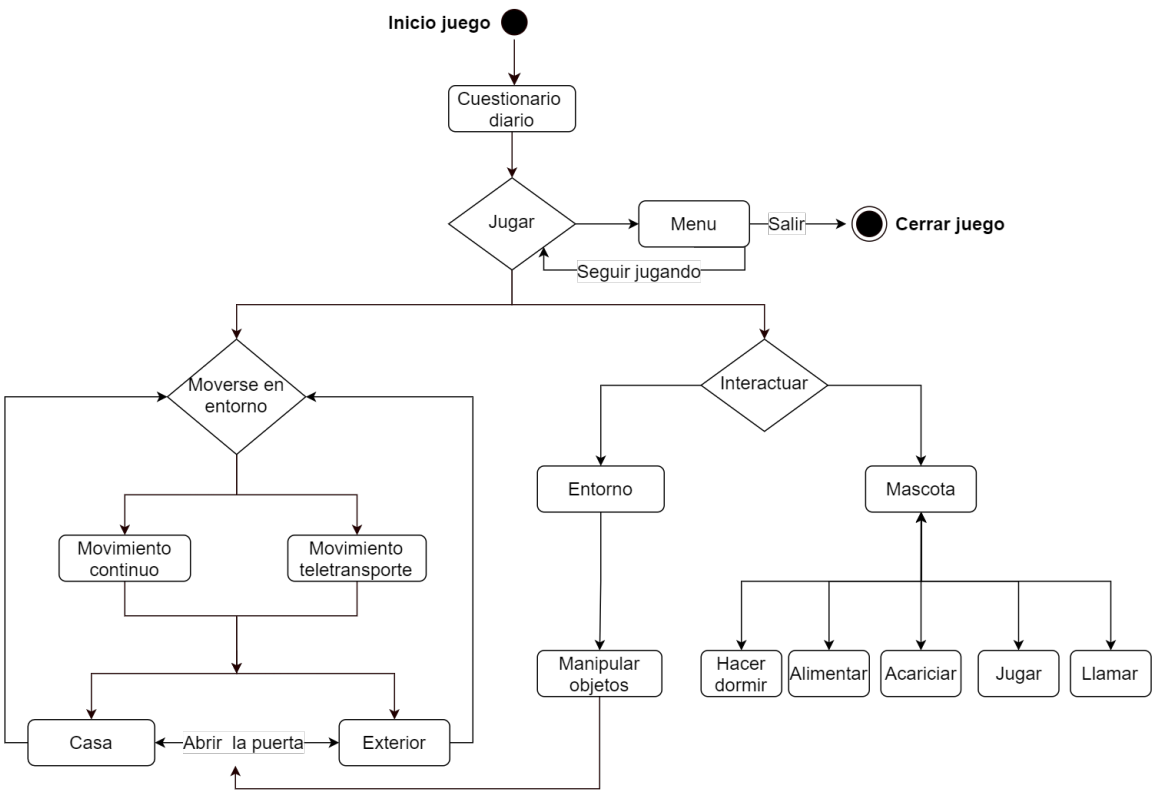


Figura 3.14: Diagrama de flujo del juego

3.6. Controles del juego

El jugador debe utilizar los controladores *Touch* de *Oculus Quest* para realizar las actividades dentro del juego. Además de los botones tradicionales del *gamepad*, los mandos *Oculus Touch* cuentan con superficies de control sensibles a la capacidad que detectan el contacto físico de los dedos o pulgares del usuario (Touch), así como su proximidad (NearTouch). Esto permite detectar varios estados distintos de la interacción de un usuario con una superficie de control específica. En la Figura 3.16 se puede ver la relación entre los distintos *inputs* de los mandos y su función dentro del juego.

3.7. Estética del juego

El diseño de los gráficos y el contenido visual de un juego es una parte importante del proceso de desarrollo de este. El estilo de arte realista emula a los personajes, objetos y entornos del juego lo más parecido posible a la realidad. Cuando los juegos se asemejan a la vida real, los jugadores se sienten inmersos en su mundo. Hay cuatro factores principales que hacen que el juego parezca realista: el número de polígonos, la textura, la iluminación y la animación. De este modo tanto los muebles como los objetos utilizan resolución realista y se utilizaron texturas PBR.

Figura 3.16: Sistema de *inputs* del juego

Por otro lado, el realismo estilizado o semi-realismo es un estilo híbrido de gráficos estilizados y fotorrealistas. El estilo se define por tener algunos rasgos exagerados que se encuentran en los dibujos animados y en los estilos artísticos, pero al mismo tiempo posee elementos fotorrealistas. El juego desarrollado al tener personajes, espacios y objetos estilizados y además al utilizar materiales realistas, texturas e iluminación avanzada en el entorno 3D se clasifica como semi-realismo. Se escogió este estilo al ser una mezcla entre otorgar experiencia realista y a su vez utilizar elementos fantasiosos como son la mascota dragón y colores atmosféricos exagerados.

Capítulo 4

Implementación del Juego

En este capítulo se detallan las principales decisiones técnicas tomadas con respecto al desarrollo del videojuego. En particular, se aborda la definición de elementos gráficos (*assets*), motor de implementación del software, e integración con el casco de realidad virtual.

4.1. Assets de juego

El primer paso de la implementación fue la adquisición de *assets* para el juego. Se adquirieron tanto los modelos 3D para el juego, las texturas *PBR* y *Skybox* de sitios externos de recursos, como por ejemplo *TurboSquid* [43]; *UnityAssetStore* [51] y *3dsky* [34] para modelos 3D, *cc0textures* [16] para texturas, *freepik* [41] para íconos 2D y *FesliyanStudios* [27] para efectos de sonido. En el Sección 6 se hará referencia a los activos individuales que se utilizan en el juego.

En el desarrollo del juego, el modelo y las animaciones de la mascota se crearon a través del software de creación 3D Blender (https://www.blender.org/). Se decidió modelar la mascota con el fin de controlar el impacto en el rendimiento de los polígonos y las texturas para utilizarlos de la forma más eficiente posible.

4.1.1. Modelo de mascota

El modelado 3D es una técnica de computación gráfica para producir una representación digital en 3D de cualquier objeto o superficie. Se utiliza un software especial, en este caso Blender, para manipular vértices formando una malla de polígonos. Una vez creado el modelo de la mascota se fue triangulando a través de una de las herramientas automáticas de Blender para agilizar el proceso dado que Unity maneja los objetos 3D como mallas de triángulos. El modelo de la mascota cuenta con aproximadamente 240 triángulos. Los modelos se hacen lo más simple posible dado que se utilizan principalmente los *Shaders* y las texturas para añadir los detalles.

Texturizado

La aplicación de texturas al modelo 3D determinan su estilo y realismo. Una textura es un mapa de bits llamado *UV map*, el cual se aplica a la superficie de un modelo para darle color, volumen o ilusión de relieve. El mapeo en relieve permite crear la superficie de un objeto con un diseño realista, y para los modelos del juego se utilizó el *Normal mapping*. En Unity las texturas usualmente utilizadas son: *Albedo*, *Metalic/Smoothness*, *Normal*, *Height* y *Ambient Occlusion* para un flujo de trabajo PBR típico. En la Figura 4.1 se muestran las texturas utilizadas para la mascota. Usualmente se utiliza un material por objeto, sin embargo algunos objetos requieren usar un material especial aparte, como por ejemplo para las expresiones de los ojos de la mascota, la cual solo utiliza una textura base y se configura a través del *Offset*. Se creó una malla aparte la cual cubre la zona de los ojos de la mascota para obtener un mejor control sobre las expresiones.

Para texturizar tanto la mascota como los demás elementos 3D del juego se utilizó Substance Painter, el cual permite generar texturas de manera procedural, es decir, utiliza un algoritmo para crear superficies realistas o representación volumétrica de elementos naturales. Sin embargo, solo se creo una textura para la *skin* de la mascota utilizando esta técnica, y para el resto se utilizaron texturas proporcionadas tanto por el software como por fuentes externas.

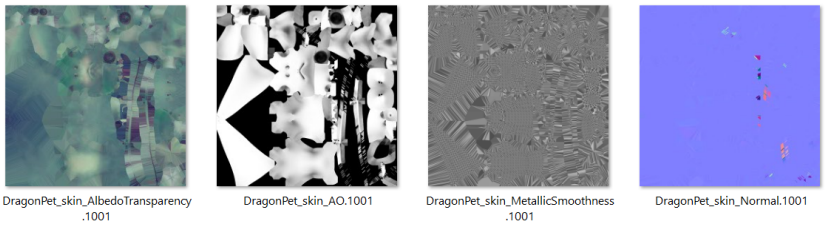


Figura 4.1: Mapas de texturas para el modelo de la mascota

Finalmente, para visualizar las texturas dentro de los modelos del juego, en Unity se crea un *material* el cual debe ser una instancia de un *shader*. En computación gráfica, un *shader* es un tipo de programa que corre en la GPU y le indica a cada píxel como debería verse en el juego, en particular le indica el color que debe tomar para visualizar la textura deseada dentro del juego. En el juego implementado se utilizaron en su mayoría los *shaders* de Unity por defecto.

Esqueleto

En la fase de *Rigging* se creó un esqueleto especial para la mascota, el cual permite posteriormente deformar la malla de forma coherente para la creación de las distintas animaciones. Un *Rig*, es decir, un esqueleto virtual, define los puntos principales que integran todo el cuerpo del personaje y permite comprender la interacción de sus diferentes partes. Blender permite escalar, mover y rotar un grupo específico de huesos para simplificar el proceso de animación y hacerlo más suave. Finalmente, se realizó el proceso de *Skinning* o *Weight Paiting*, es decir, la unión de la malla geométrica de la mascota con el *Rig* creado.

Animación

En la fase de animación, para cada una de ellas se indicó la posición de la mascota 3D en los primeros y últimos fotogramas del movimiento, y la herramienta de Blender realiza el cálculo de los fotogramas restantes generando una transición entre las poses. Se crearon distintas animaciones para los estados de quietud o *idle*, comer, recibir cariño, movimiento y dormir. Para los estados de movimiento y dormir se crearon dos animaciones adicionales para generar transiciones más fluidas entre los distintos estados. En la Figura 4.2 se muestra el proceso de animación en Blender.

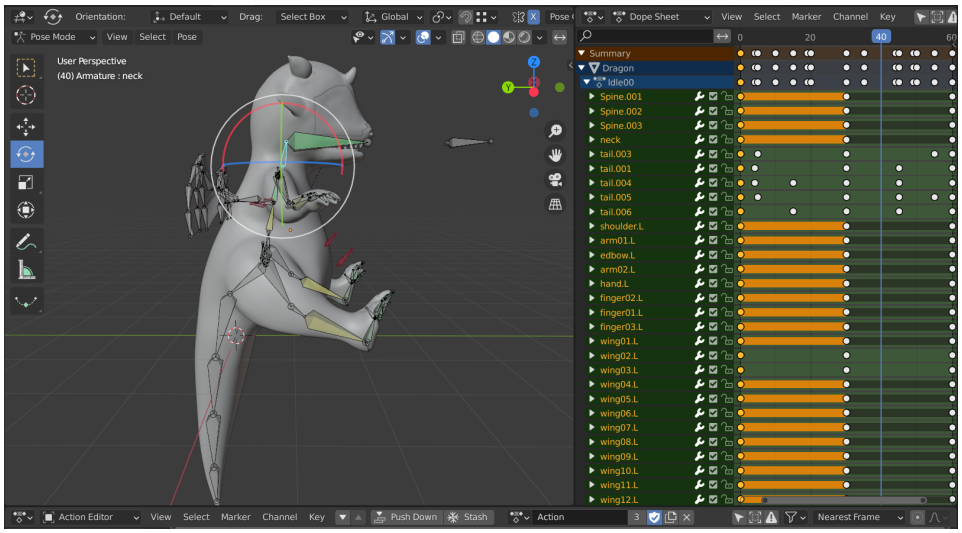


Figura 4.2: Proceso de *rigging* y animación de la mascota en Blender

Una vez realizados los pasos preparatorios, el modelo de agente de la mascota se implementa dentro del motor de Unity.

4.1.2. Modelo de la casa

En cuanto al principal escenario del juego, la casa, se creó fundamentalmente utilizando modelos 3D externos. Sin embargo, con fines de mejor manejo del rendimiento se modeló la estructura principal en Blender, es decir, el piso, las paredes, el techo, la puerta y las ventanas, creando separaciones convenientes para su posterior uso dentro del juego; por ejemplo, la puerta de la casa se separó de la pared y se configuró su origen en uno de los extremos a modo de pivote. En el Sección 6 se muestran el interior y el exterior de la casa en conjunto con el ambiente del juego en distintos momentos del día.

4.2. Estructura del motor de desarrollo

Unity es un motor basado en componentes, y es con una combinación de componentes con la que se construye el juego. Posee una jerarquía de alto nivel en la que las entidades

contienen a otras entidades. Dichos componentes se muestran en la Figura 4.3.

En términos generales, su arquitectura consiste en considerar que una *Escena* es una colección de `GameObjects`, y los `GameObjects` son una colección de Componentes que pueden implementar e incluir: Sistemas (cámaras y física), Datos (configuraciones, animaciones y texturas) y Comportamientos (mecánicas de juego y eventos a través de *script*). El motor está construido con C/C++ nativo internamente; sin embargo tiene una envoltura de C#, y dicho lenguaje se utiliza en los *scripts* de Unity. Además, cada *script* que manipule componentes de Unity debe heredar la clase `MonoBehaviour` que contiene todas las propiedades y métodos propios del motor.

Por otro lado, las clases de Unity también pueden heredar de la clase `ScriptableObject`, lo cual crea un contenedor de datos que puede guardar grandes cantidades de información, independientemente de las instancias de la clase. Uno de los principales casos de uso de los `ScriptableObjects` es reducir el uso de la memoria del Proyecto evitando copias de valores. Los datos que se guardan desde las herramientas del *Editor* a `ScriptableObjects` quedan guardados en la carpeta de *asset* del juego y por lo tanto son persistentes entre distintas sesiones.

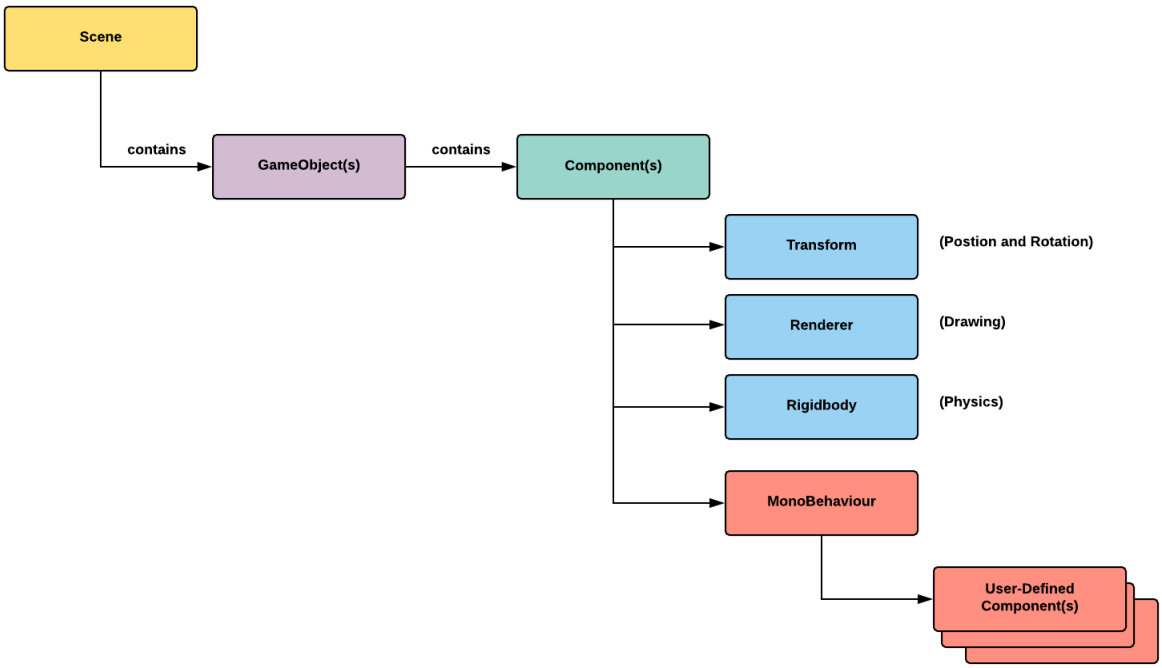


Figura 4.3: Diagrama de componentes en Unity [7]

4.3. Integración con Unity

Unity posee un *framework* de *plug-in*s llamado *XR SDK* que permite a los proveedores de *XR*, como Oculus, integrarse con el motor de Unity y hacer un uso completo de sus características. Consiste en una API que expone la funcionalidad común en todas las plataformas que soporta Unity.

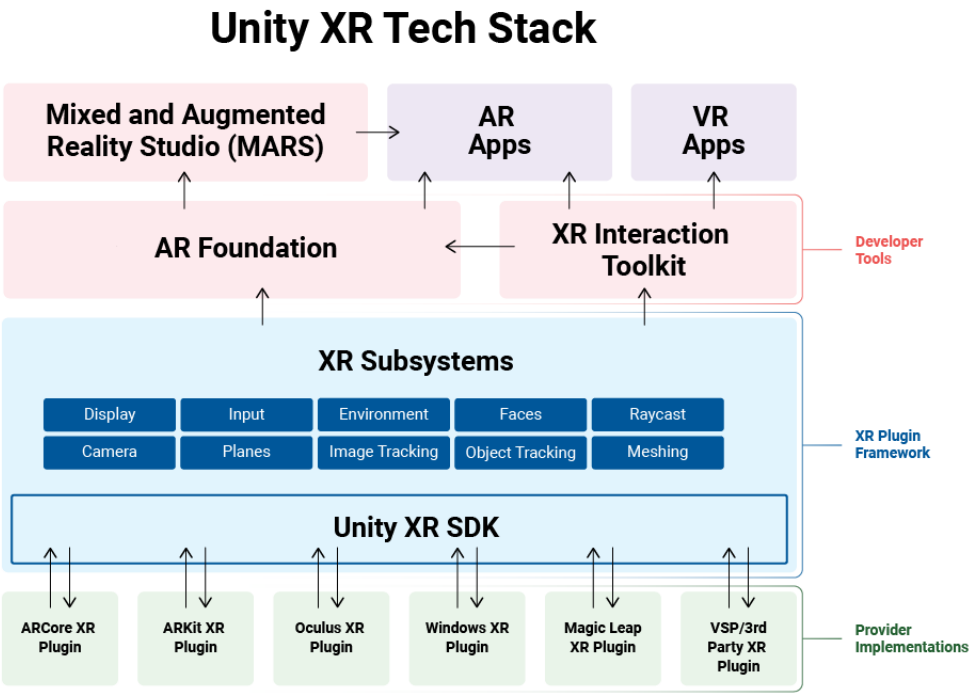


Figura 4.4: Plugins de Unity XR y funcionamiento con las implementaciones de proveedores de plataformas [50]

Para hacer uso de dichas funciones se instalaron los paquetes *XR Management* y *XR Interaction Toolkit* desde la ventana de *Package Manager* de Unity. Luego, se seleccionó un *Plug-in Provider*. En este caso, para Oculus, se habilitó Android desde la ventana de configuraciones del proyecto. Para que la conexión del juego a través del Oculus funcione correctamente, debe haber un `XRInteractionManager` y un `XRRig` en la *Escena*, el cual se encarga de establecer el seguimiento del jugador dentro de ella. Para configurar el `XRRig` se agregó una referencia a la posición de la cámara principal y un parámetro llamado *Y offset* para la altura en la que se posicionará la cámara simulando la altura del jugador.

En cuanto a la configuración del jugador, se agregó un componente llamado `TrackedPoseDriver` a la cámara principal y se seleccionó la pose como referencia al visor VR, lo cual permite realizar el seguimiento de la cámara del Oculus.

4.3.1. Sistema de locomoción

Para definir la zona de navegación válida se creó un plano que abarca dicha área y se agregó el *script* `TeleportationArea` proporcionado por el *plugin* de *XR*. Además, el `XRRig` se configuró con el *script* `LocomotionSystem`. Para el movimiento con teleoperación se agregó un `TeleportationProvider` y `SnapTurnProvider`.

Para visualizar la zona a la que el usuario se va a teletransportar se agregó un `XRRayin-`

`teractor` que se activa con el *thumbstick* del control derecho el cual permite interactuar con las áreas de *teleport* previamente definidas. El tipo de línea del `XRRayinteractor` se definió como *Projectile Curve* para que siga una trayectoria curva desde la mano del jugador hasta la zona señalada.

Además, se agregaron efectos visuales para resaltar la zona de *teleport*, se agregó una textura de línea punteada para mejorar la visualización en el `XRInteractorLineVisual` y se creó un objeto llamado *Teleport Reticle* el cual se utiliza para señalar la zona al final de la trayectoria, a la cual se le aplicó un *shader* customizado para simular efecto cilíndrico de transparencia, como se visualizó en la Figura 3.3, dicho *shader* fue creado utilizando la herramienta de Shader Graph [44] de Unity y se muestra en la Figura 4.5. Finalmente, se agregó un efecto de luces con un sistema de partículas emergentes de la *Reticle*.

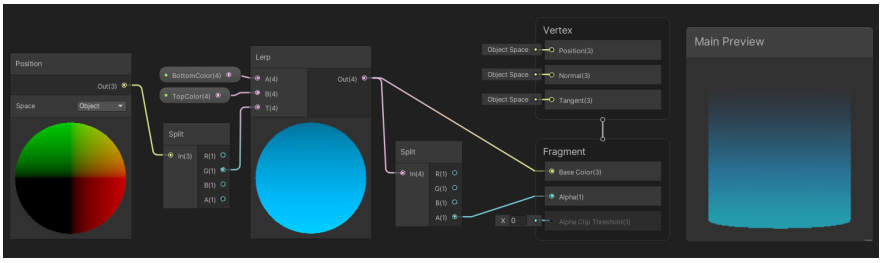


Figura 4.5: Visualización del *Shader Graph* creado para el *Teleport Reticle*

Por otro lado, con el fin de que se active el *Ray render* solo cuando se quiera utilizar, se creó el *script* `LocomotionController`, el cual hereda de `MonoBehaviour` y contiene el método `CheckIfActivated` que verifica que el *thumbstick* este presionado por cierta cantidad de tiempo, definida por el parámetro *Activation Threshold*.

Para el movimiento continuo se implementó el *script* `ContinuousMovement` que también hereda de `MonoBehaviour`, el cual permite al usuario moverse continuamente por el área válida de *teleport*. Dicho movimiento se controla con *thumbstick* del control izquierdo. El *script* obtiene la dirección del input del *thumbstick* y la dirección a la cual esta orientada la visual del usuario y mueve el *Character controller* del jugador en dicha dirección. Además, utiliza las componentes de gravedad para simular la caída en caso de que el jugador se mueva de un nivel en altura a uno más bajo.

4.3.2. Presencia de manos

Para configurar los controles y simular las manos del jugador se crearon dos componentes hijos en el XR Rig: `LeftHand Controller` y `RightHand Controller`. Ambos utilizan los *scripts* `XR Controller` y `XR Direct Interactor`. Para el modelo 3D de las manos y sus animaciones correspondientes se utilizaron los `CustomHandPrefab` del paquete de *Oculus Integration* proveniente de *Unity Asset Store*[51].

El `XRDirectInteractor` permite que las manos virtuales interactúen con los objetos del ambiente que posean algún *script* del tipo de *XR Interactor*, es decir, permite que se comuniquen las acciones realizadas a través de eventos de interacción.

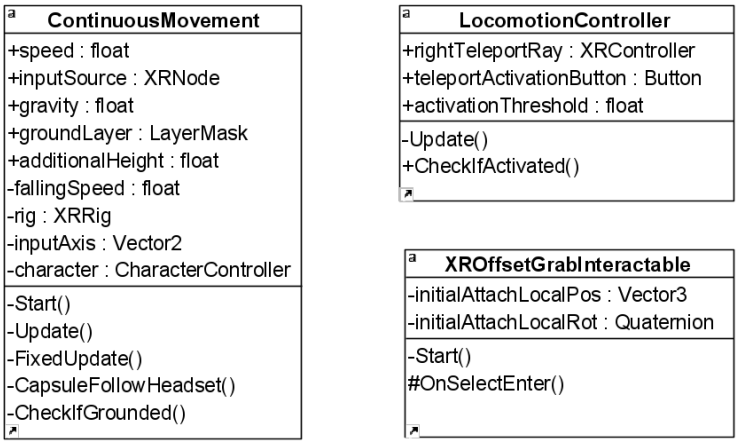


Figura 4.6: Diagrama de las clases creadas para los sistemas de locomoción y *offset grab*

Para la mecánica de agarrar los objetos con las manos se agregó el *script* `XRGrabInteractable` en los objetivos, el cual ofrece tres tipos de movimiento: *Velocity Tracking*, *Kinematic* e *Instantaneous*. En el entorno virtual, para simular el movimiento de abrir la puerta se utilizó *Velocity Tracking* dado que este movimiento sigue el movimiento de la mano. Para el resto de los objetos interactuables se utilizó el movimiento *Kinematic*, el cual se asimila más al comportamiento de los objetos en la vida real dado que detecta las colisiones con los demás elementos y tiene un movimiento basado en la física más preciso. Además, para conseguir un comportamiento de agarre más preciso de creó un *script* `XROffsetGrabInteractable` que hereda de `XRGrabInteractable`, el cual permite que el objeto se adhiera a las manos desde la zona en la que se toma en vez del centro de origen.

4.4. Configuración de la escena en Unity

El juego se desenvuelve en una única *Escena*, la cual está compuesta por los elementos del entorno, la mascota y el *XRrig* del jugador.

4.4.1. Mascota

Para permitir que la mascota navegue por el entorno de manera realista se utilizaron los componentes *NavMesh Agents* [47] y *NavMesh Baking* [45]. La *NavMesh* representa el área donde el centro del agente puede moverse. En el `NavMeshAgent` de la mascota se configuraron los parámetros físicos de movimiento como la velocidad y la aceleración. Por otra parte, para manejar las distintas animaciones de los estados por los cuales pasa la mascota se agregó un componente *Animator* de Unity el cual referencia al controlador de estados de animación.

Con el fin de lograr un mayor control sobre el movimiento de la mascota se utilizó un *Rig builder*, el cual es un componente proporcionado por el paquete *Animation Rigging* que permite manipular el *Rig* en tiempo de ejecución. Esto se utilizó para controlar la dirección

a la que la mascota mueve la cabeza, moviendo su atención a los distintos puntos por los que puede navegar.

Además, para tratar las colisiones con la mascota se le añadió un *collider* en forma de cápsula que cubría el cuerpo en su totalidad junto a un *Rigidbody*, y sub componentes con *colliders* en forma de cápsula para áreas especificas como la boca, la cabeza y su torso, a los cuales se les activó la característica de *trigger*, lo cual hace que el *collider* se pueda traspasar por otros objetos pero aún así se detecte dicha colisión.

Finalmente, se creó un componente que almacena puntos de ruta a los cuales la mascota puede acceder a través del `NavMesh`. Este consiste en 15 componentes que guardan las coordenadas de posición de distintos puntos distribuidos en el interior de la casa.

4.4.2. Entorno

A continuación se explican los diversos elementos que componen la escena del juego.

Casa: Es el conjunto de modelos 3D previamente configurado en Blender. Se encuentra amoblada principalmente con una cocina, una cama para la mascota y algunos objetos decorativos. Los objetos dinámicos como la comida o los juguetes se asociaron a componentes *Collider* y *Rigidbody*, además para ser accesibles por las manos del jugador se les asoció el *script* `XROffsetGrabInteractable`.

Terreno: Con el fin de proveer un ambiente exterior natural se utilizó el componente *Terrain* de Unity al cual se le agregaron relieve, *assets* y texturas de naturaleza para simular un paisaje diverso y orgánico para los jugadores.

Iluminación: Para la iluminación exterior de la escena se creó una luz direccional la cual representa la fuente de sol. Por otra parte, para la iluminación interior de la casa se utilizaron una mezcla entre *Spot light* y *Area light*. Además, se agregó un grupo de *Light Probes* y *Reflexion Probes* para la luz dinámica y las reflexiones del ambiente. Para el *Skybox* se utilizaron dos *assets*: uno para el día y otro para la noche, y tanto los colores ambientales como el *fog* cambian dinámicamente en tiempo real a través de *script*.

Volumen de Post Process: El volumen se agrega como componente dentro de la *Escena* y otorga efectos visuales como ajustes avanzados de color, como por ejemplo cambiar el color de las sombras o el tono y exposición del ambiente, y efecto de *bloom*, el cual potencia el brillo e intensidad de la iluminación de la escena.

4.5. Estructura del juego

La estructura utilizada para la implementación del juego posee diversos elementos que cumplen distintos propósitos, los cuales se extienden de `MonoBehaviour` o `ScriptableObject` dependiendo del caso. Para la mayoría de las clases que extienden de `MonoBehaviour` se utilizó

el *Singleton Pattern*, el cual es una forma de asegurar que la clase tenga una única instancia accesible globalmente disponible en todo momento. Además, esto se usó para dar acceso a variables y funciones globales a las que otras clases necesitan acceder. Esto se realizó haciendo referencia a una instancia estática de la misma clase.

4.5.1. Controlador del jugador

En cuanto al funcionamiento del jugador se creó un administrador `PlayerController`, el cual hereda de `MonoBehaviour`. Para hacer funcionar los inputs de los controles del jugador se crearon dos variables de tipo *InputDevice* que representan el control derecho y el izquierdo. Para obtener el input correcto del usuario y mapearlo con interacciones dentro del juego se utilizó como referencia la guía *XR input mappings* [49] de Unity, que muestra los nombres de cada botón.

El jugador cuenta con dos acciones a las que puede acceder a través de los botones *primaryButton* y *secondaryButton* del control *touch*. Dichas acciones se definen a través de los métodos `CallPet` y `SetMenuActive`, los cuales son para llamar a la mascota a que se acerque al jugador y desplegar el menú del juego respectivamente.

Además, se cuenta con el método *setState* el cual recibe información de la encuesta inicial y guarda el estado del jugador.

4.5.2. Inteligencia artificial (IA) de la Mascota

Controladores

Para el agente de la mascota se crean dos administradores o controladores `DragonController` y `NeedsController`. Estos mantienen conocimiento de los parámetros internos de la mascota y gestionan las interacciones y actualizaciones de dichos parámetros. Además, se crea una clase abstracta `PetState`, la cual es utilizada para abstraer la respuesta y manejo de los distintos estados de la mascota. Se crearon 6 clases que sobrescriben los métodos de `PetState` y describen cada estado, estos son: `IdleState`, `MoveState`, `HappyState`, `EatState`, `SleepState` y `FaintState`. El `DragonController` se encarga de realizar los cambios a los parámetros externos de la mascota, como lo son los cambios en los estados de animación, en las expresiones, en el movimiento hacia un nuevo objetivo y en el cambio de la orientación de la cabeza. Por otro lado, la clase `NeedsController` es la encargada de almacenar y modificar los parámetros internos de la mascota. Las variables principales son las que mantienen registro de su estado interno: `food`, `happiness` y `energy`, y los parámetros relacionados al máximo valor que pueden tomar, la tasa a la cual disminuyen en el tiempo y además se agregó un valor de retardo para cada variable el cual se utiliza al momento de mostrar los mensajes de la mascota.

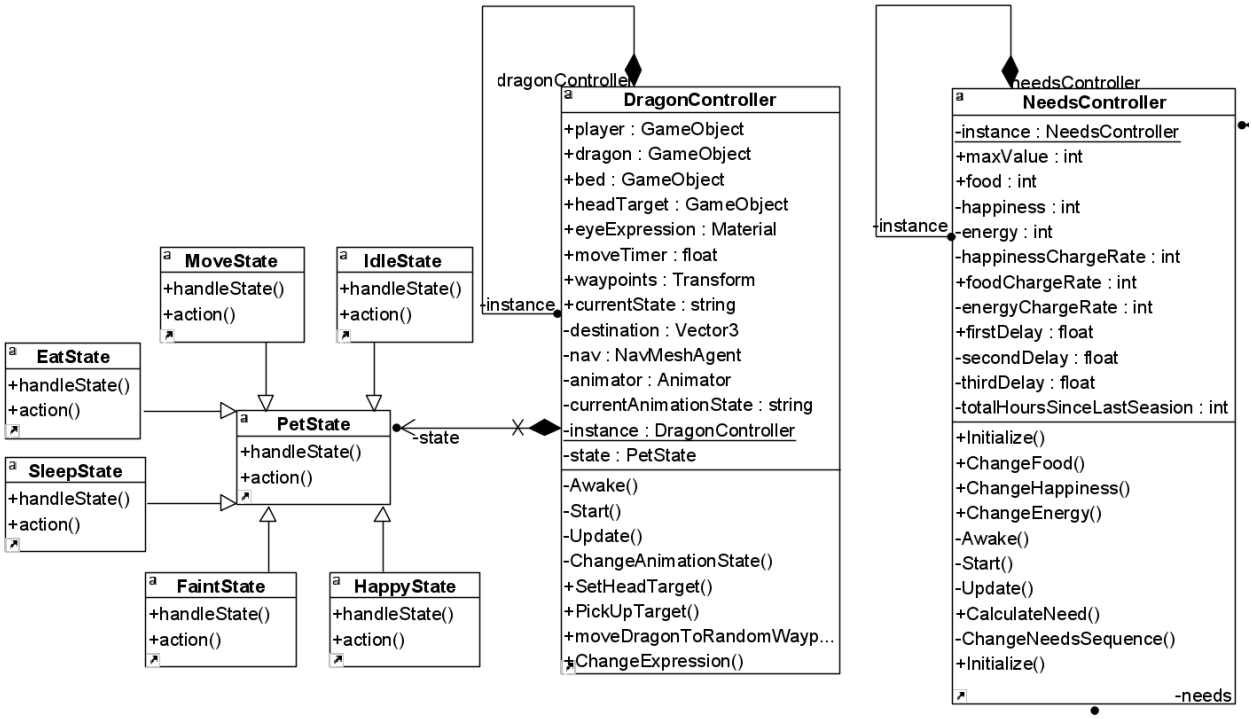


Figura 4.7: Diagrama de clases de `DragonController` y `NeedsController`

Animaciones

Para manejar de manera correcta las distintas transiciones entre las animaciones del personaje, se modeló la máquina de estados utilizando la herramienta de edición `Animator` de Unity, el cual se muestra en la Figura 4.9, donde cada nodo está ligado a una animación independiente. Los estados son los siguientes:

- Idle: La mascota se encuentra quieta en un punto de la *Escena* sin recibir acciones del usuario.
- ToMove: Cuando el `MoveTimer` llega a 0 o cuando el usuario llama a la mascota, se mueve a un punto distinto dentro de la Escena. Al terminar la animación de este estado se pasa inmediatamente a *Move*.
- Move: La mascota se encuentra en el trayecto desde un punto a otro en la Escena.
- MoveToIdle: Esta animación se llama cuando la mascota se encuentra en el punto de destino. Al terminar esta animación se pasa automáticamente al estado *Idle*.
- ToSleep: Cuando el usuario manda a dormir a la mascota o cuando alguno de los niveles internos de esta llegan a 0 y entra a estado *Faint*. Después de esta animación se pasa a *Sleep*.
- Sleep: Mientras la mascota se encuentre en el estado de estar durmiendo dentro de su cama.
- SleepToIdle: Cuando la mascota repone a casi el máximo su nivel de energía. Al terminar esta animación se pasa de inmediato a Idle.
- Eat: Esta animación se reproduce cuando el usuario le acerca a su boca un objeto con

el *tag* de *food*.

- Happy: Se reproduce cuando el usuario sobrepone sus manos virtuales en alguna zona del cuerpo de la mascota, como acariciándola.
- Play: Se llama cuando el usuario intenta jugar con la mascota, agarrando alguno de los objetos con el *tag* de `playtarget`.

Para realizar los cambios entre las distintas animaciones se implementó el método `ChangeAnimationState`, el cual recibe un estado y si es distinto al actual entonces da la orden al `animator` para que reproduzca dicho estado. Cada una de las clases que heredan de `PetState` se encargan de las transiciones de animación en el método `action`. Por otro lado, las animaciones son importadas desde Blender y desde el inspector de Unity se indica si son animaciones en *loop*, el *frame* inicial y final de la animación y la velocidad a la que se reproduce. Las transiciones indicadas en el `Animator` son inmediatas por lo que, por ejemplo, para la animación de mover basta con llamar al estado `ToMove`.

Expresiones

Por otro lado, aparte de las animaciones se crearon 4 distintas expresiones faciales para los ojos de la mascota, con la finalidad de mostrar un mejor *feedback* al usuario sobre la diferencia de los distintos estados de la mascota. Para implementar este efecto, se utilizó una *Sprite Sheet* con las distintas expresiones, y se asoció a una textura la cual es asignada a través del *UVmap* a la zona de los ojos, se utilizó como base el método explicado en el articulo Ben et al. [9]. Para optimizar la textura, solo fue necesario dibujar el ojo derecho para cada estado, ya que el modelo cuenta con el efecto de espejo el cual invierte y posiciona el ojo izquierdo en el *UVmap*, como muestra la Figura 4.8. Los ojos corresponden a los estados de *Idle* o en movimiento, feliz o jugando, durmiendo y por último recibiendo cariño.

Figura 4.8: Sprite Sheet para la textura de los ojos de la mascota en distintos estados

Para cambiar las expresiones de la mascota se creó un único material para los ojos, utilizando el *shader* de Unity *Unlit/Tansparent*, el cual sólo guarda información de la textura, su *tiling* y *offset*. Luego, se implementó el método `ChangeExpression` el cual utiliza el método `SetTextureOffset` proporcionada por el motor. A través de esto se definen 4 *Xoffset* para indicar el estado de los ojos. Las transiciones entre los distintos *Xoffset* se manejan del mismo modo que las animaciones.

Interacciones con el usuario

Las interacciones en VR del usuario con la mascota se manejaron a través de un sistema de eventos basado en el *Observer Pattern*, en donde el *Observable* es la mascota y los *Ob-*

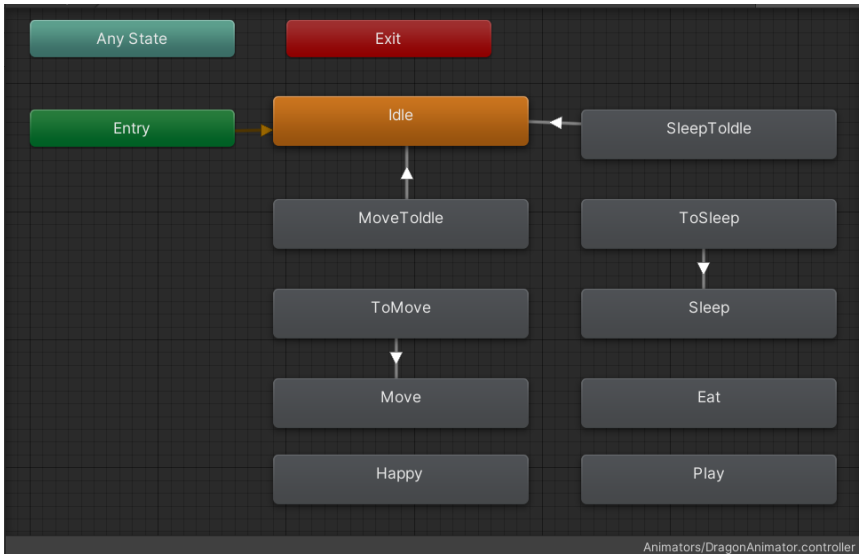


Figura 4.9: `Animator Controller` de los estados de animación de las mascota

server las clases que manejan los distintos estados, animaciones y expresiones de esta. Para esto, a cada subcomponente *collider* de la mascota que tiene un *trigger* se le asoció también un componente llamado `XRSimpleInteractable`, el cual hace que el componente pueda recibir interacciones con las manos VR del usuario. Este contiene una lista de eventos de interacción, de los cuales se utilizan `OnHoverEnter` y `OnHoverExit` para tratar las distintas acciones del usuario. Dependiendo del tipo de evento se utilizan métodos de `DragonController` y `NeedsController`. Los *colliders* de la cabeza y el torso son para detectar si el usuario acaricia a la mascota, y el de la boca es para detectar si se le proporciona alimento. Los objetos de alimento en la Escena tienen un *collider* en forma esférica y el componente `XROffsetGrabInteractable`, junto con el *tag Food* asociado.

Por otro lado, la cama de la mascota tiene un *collider* en forma rectangular en la base y dos componentes para hacer que este sea manejado mediante VR: `XRSimpleInteractable` y `XRTintInteractableVisual`. Esto hace que al señalarla con el control cambie de color y se pueda activar, lo cual da la orden a la mascota de ir a dormir. Esta interacción se maneja a través del evento `OnSelectEnter`.

Finalmente, los objetos con forma de pelota los cuales permiten al usuario jugar con la mascota tienen un *collider* en forma de esfera y se manejan los cambios en el estado de la mascota con los eventos de `OnSelectEnter` y `OnSelectExit` del componente `XROffsetGrabInteractable` asignado.

De este modo, los niveles de felicidad, energía y comida de la mascota suben a medida que el usuario va supliendo dichas necesidades alimentándola, acariciándola y haciéndola dormir, aunque por otro lado jugar hace que pierda energía. Cada una de estas acciones tienen un valor constante asociado el cual se va sumando en el `NeedsController` a través del sistema de eventos con los métodos `ChangeFood`, `ChangeHappiness` y `ChangeEnergy`.

Asimismo, para lograr que la mascota oriente su cabeza y torso hacia el objetivo al que se debe mover se utiliza el *Rig builder* mencionado anteriormente. Para esto, dentro del modelo

de la mascota se crea un componente llamado *Rig* el cual tiene dos componentes hijos: *HeadAim* y *ChestAim*. Estos determinan qué parte del *Rig* de la mascota será controlada a través del objetivo asignado, el cual es un punto con posición definida dinámicamente a través de código con el método `SetHeadTarget` del `DragonController`. Para asignar la parte del *Rig* y cuánto control se tendrá sobre el movimiento de esta se utiliza el componente *Multi-Aim Constraint*, con el cual además se asigna un límite de rotación para evitar movimientos poco realistas o exagerados.

Para implementar el movimiento de la mascota se le asignó el componente *NavMeshAgent* mencionado anteriormente, en el cual se definen la altura, el radio, la velocidad, la aceleración y la distancia a la cual se detiene. Luego, dentro de la implementación del método `Move` en el controlador de la mascota basta con utilizar `SetDestination` en la *NavMesh* para que la mascota se mueva a dicho objetivo.

4.5.3. Interfaces de usuario

A continuación se describen las componentes principales que permiten la interacción del usuario con la aplicación desarrollada.

Comunicación con el usuario

Los parámetros internos de la mascota deben ser expuestos al usuario para que reaccione adecuadamente a las necesidades de la mascota. Para ello, se crea un *Canvas* sobre el modelo de la mascota en el cual se visualizan las necesidades de ella a través de iconos. De este modo el *Canvas* se mueve en conjunto con la mascota, pero al ser en 2D en un espacio 3D se debió implementar un efecto *Billboard*, el cual hace que el *Canvas* se oriente para estar siempre frente a la vista del usuario. Se creó un *script* llamado `PetUIController` que hereda de `MonoBehaviour` y se encarga de manejar los distintos elementos del *Canvas*. Dichos elementos son: *heartSlider*, el cual es un icono de corazón en donde su color de relleno representa el porcentaje de satisfacción de las necesidades de la mascota; *foodBubble*; *loveBubble*; *playBubble* y *sleepBubble*, los cuales son *iconos* en forma de burbuja emergente que tienen en su interior una imagen que representa cada una de las necesidades de la mascota. Cada elemento está conformado por un *Panel* el cual contiene su respectiva imagen, las cuales corresponden a texturas 2D de tipo *Sprite*.

Para administrar los parámetros de dichos elementos se crearon los métodos `SetHeartSlider` y `ShowPetNeeds`. Para no confundir al usuario, los iconos de las necesidades se muestran cada cierta cantidad de tiempo, por lo que para hacerlas desaparecer se utiliza un parámetro *timer* y *Coroutines* implementando el método `SetVisibleAfterTimer`, el cual es un `IEnumerator` y retorna una vez que pasa el tiempo del *timer*. Además, en el `NeedsController` se utilizan también *Coroutines* para que a medida que los parámetros de las necesidades de la mascota sean más bajos, el mensaje de dicho estado se muestre más seguido. Esto se hace a través del método `CalculateNeed`, el cual asigna el valor de *delay* correspondiente.

Menú del juego

Para mostrar un menú interactivo en VR se creó un componente que contiene un `EventSystem`, el cual permite posicionar la UI en espacio de mundo en 3D y además incluye su propio sistema de *Raycaster* gráfico para 3D, lo cual permite controlar los botones y otros elementos de la UI a través del *Ray interactor* del control. Además, se agregó el *script* `XRUIInputModule` al sistema de eventos, con el cual se pueden definir la distancia a la que se encuentra el menú de la vista del usuario, su tamaño, la velocidad a la que se mueve y la velocidad de recepción del input. La lógica de la UI en un espacio 3D se solucionó haciendo que esta se encuentre siempre orientada frente a la vista del usuario, para que este no pierda de vista la interfaz. La velocidad a la que se mueve junto a la orientación de la cabeza del usuario se definen con el parámetro *Move Deadzone* del `XRUIInputModule`, el cual define el valor absoluto requerido por una acción de movimiento en cualquiera de los ejes para desencadenar un evento de movimiento.

Para crear el menú en sí, se creó un nuevo *Canvas* en la escena con un componente *Background* que es un *Panel* que oscurece el fondo. Se creó un componente *Panel* llamado *MenuUI* el cual es asociado al *script* `MenuUIController` cuyos principales métodos son `QuitGame` y `QuitMenu`, los cuales son asociados a los dos botones del menú a través del evento *OnClick* de la UI.

Encuesta de estados de animo

La encuesta que recibe el usuario al iniciar el juego consiste en un *Panel* con 5 elementos los cuales son botones en forma de iconos asociados a imágenes de distintos emoticones o expresiones en formato de *sprite*. Para manejar los distintos elementos se creó el *script* `SurveyUIController`, el cual básicamente asigna el estado seleccionado al jugador, esto se hace a través del evento *OnClick* de cada elemento.

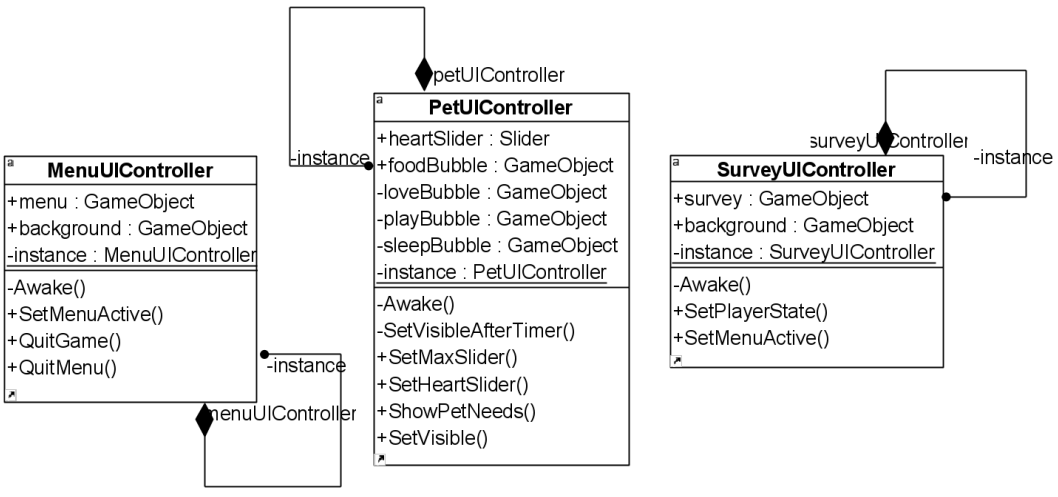


Figura 4.10: Diagrama de clases de los controladores de las UI implementadas

4.5.4. Manejo de Datos

Con el objetivo de proporcionar una línea temporal coherente entre cada sesión de juego se creó una clase `Database` a través de la cual se permite guardar datos de cualquier tipo en formato *Json* en la carpeta de *Saves* dentro del proyecto. Para la implementación del sistema de guardado de la sesión del juego se implementa una clase `DatabaseManager` que administra el guardado y la carga de los parámetros internos de la mascota y por otro lado de la fecha y hora en la que se cerró el juego. Luego, estos datos son utilizados para calcular el estado en que se encontrará la mascota en la próxima sesión basado en el estado en que esta se encontraba la última vez que fue visitada y en el tiempo en el que usuario se mantuvo fuera del juego.

Para guardar los datos tanto de la mascota como de la fecha de sesión se generalizó el método `SaveData` en `Database` para que permita guardar en *Json* cualquier tipo de objeto. Dado esto, se crearon las clases `DragonPet` y `DateTimeObject` que tienen solo un constructor, los cuales se utiliza para guardar y cargar los datos correspondientes.

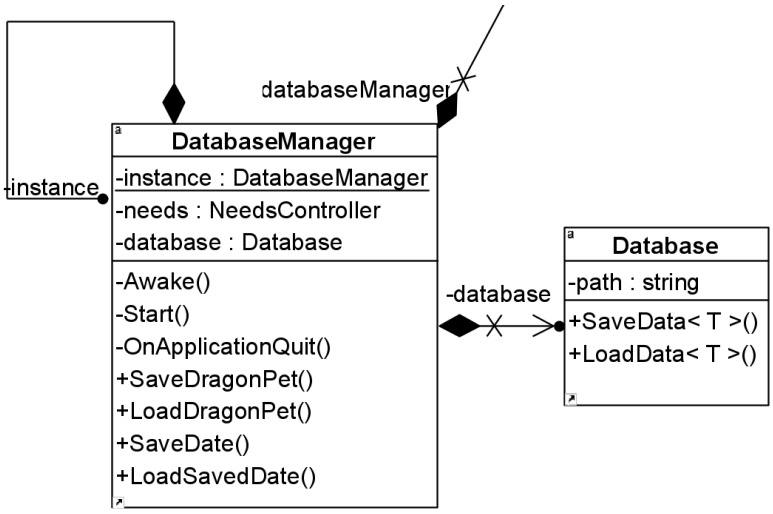


Figura 4.11: Diagrama de clases de `Database` y `DatabaseManager`

4.5.5. Manejo de Ambiente

En el juego el tiempo pasa por un ciclo de día y noche. Para simular este efecto se creó un `LightingPreset` el cual se hereda de la clase `ScriptableObject` de Unity. A partir del `LightingPreset` se instanció un *asset* en el proyecto en el cual se definen los campos de *Ambient Color*, *Directional Color*, *Fog Color* y *Sky Color* en escala de *Color Ramp*, el cual es un mapa de degrade de un rango de colores ordenados linealmente con la intención de dar visualmente una transición suave y progresiva entre dos o más colores. Estos se muestran en la Figura 4.12.

En cuanto al manejo de dichas variables visuales en la *Escena*, se creó el *script* `LightingManager`, el cual calcula en el *Update* el valor de *TimeOfDay* y además *renderiza* los

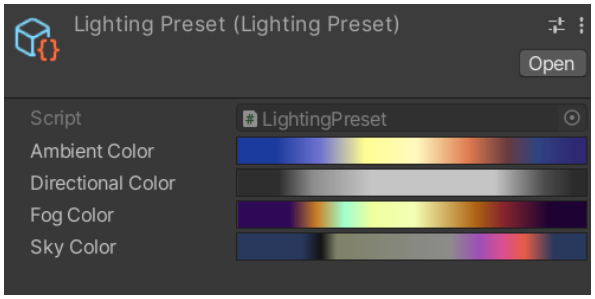


Figura 4.12: Los *Color Ramp* definidos en la instancia del `LightingPreset`

ReflectionProbe para que se realicen los cambios en las reflexiones de la *Escena* con el cambio de iluminación.

El *script* con 2 métodos fundamentales para generar la iluminación del juego:

- `UpdateLighting`: Este método se encarga principalmente de actualizar los *RenderSettings* en la sección de *Environment* en los *Lighting Settings* de la *Escena*, los cuales corresponden al *ambientLight*, *fogColor* y el tinte del *skybox*. Los valores de dichos colores se obtienen evaluando el valor del *TimeOfDay* y sus correspondientes *Color Ramp* en el `LightingPreset`. Además, se define la rotación en el eje Y del *skybox* con el valor de *TimeOfDay*. El valor *Directional Color* se utiliza para asignar tanto el color como la posición de la *DirectionalLight* de la *Escena*. Finalmente, se utiliza el valor de *TimeOfDay* para asignar la textura del *skybox* de cielo nocturno durante la noche.
- `SetPlayerAmbientColor`: Este método se usa para cambiar el color de ambiente según el estado en el cual se encuentra el jugador. Para lograr esto, se accede al volumen de *Post-process*, específicamente a la opción de *colorAdjustments* y se asigna un color para el *colorFilter*, el cual es un Vector4 que está previamente definido para cada estado.

Finalmente, con lo que respecta al sonido en la Escena, se optó por utilizar un único audio en *loop* a modo de música de fondo. Para esto, se le agrego el componente *Audio Source* al *XRRig* del jugador, se inicia el sonido al empezar el juego y este se mantiene a un volumen bajo.

4.6. Rendimiento

El rendimiento es una característica clave de los juegos de realidad virtual en este momento, debido a su limitación de potencia de hardware. El motor de juego utilizado admite una serie de técnicas para optimizar el juego 3D. En esta sección se repasarán las técnicas utilizadas para la optimización del juego. Como objetivo de rendimiento se utilizaron los parámetros definidos en el artículo de Oculus sobre *Testing and Performance* [39], donde se especifican 72 FPS para Oculus Quest 1.

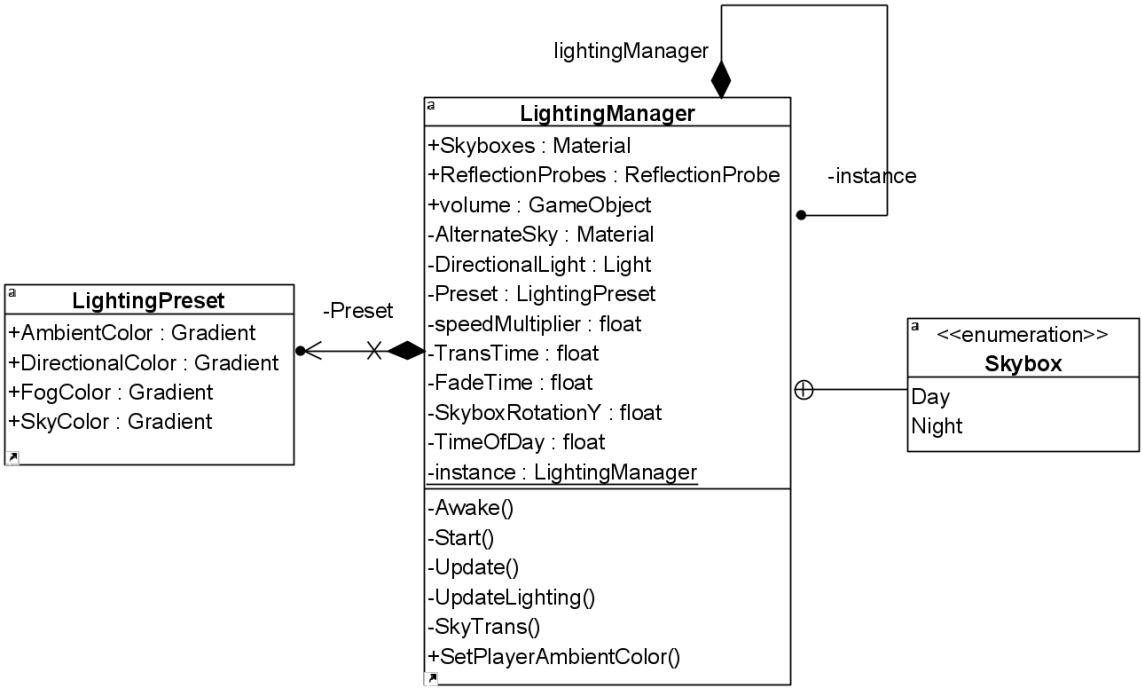


Figura 4.13: Diagrama de clases de `LightingPreset` y `LightingManager`

4.6.1. Físicas

En Unity para que un objeto simule un comportamiento físico realista se utiliza un componente llamado *Rigidbody* con gravedad activada mientras que para la detección de colisiones sin simulación de la física se usa un *Rigidbody* con el campo de cinemática activo. Además, Unity utiliza componentes llamados *colliders* [46] para comprobar la colisión entre los objetos del juego. Estos se clasifican en dos tipos: estático y dinámico. El *collider* estático es un *collider* sin cuerpo rígido, se utiliza para objetos estáticos en la escena, por ejemplo los componentes de la casa, muebles y objetos no manipulables por el jugador. El *collider* dinámico se utiliza para las manos del jugador, la mascota y los objetos que pueden ser manipulados por el usuario. Para reducir el coste de rendimiento se utilizó el método de *colliders* compuestos, los cuales se aproximan a la forma de un *GameObject* manteniendo una baja sobrecarga del procesador a través de la combinación de varios *colliders* primitivos, en lugar de *colliders* de malla.

4.6.2. Vértices y polígonos

El número de vértices de los modelos 3D afecta tanto a la CPU como a la GPU para su posterior procesamiento. Basado en los parámetros definidos por Oculus se estiman un máximo de 300.000 a 500.000 triángulos por *frame*.

Otra de las técnicas utilizadas para reducir la cantidad de vértices renderizados es la de *Occlusion culling* [48], el cual es un proceso que evita que Unity realice cálculos de renderizado para *GameObjects* que están completamente ocultos de la vista de la cámara por

otros *GameObjects*. En cada *frame*, la cámara realiza operaciones de culling que examina los Renderers en la Escena y excluye (cull) aquellos que no necesitan ser dibujados. *Occlusion culling* genera datos acerca de la Escena en el Editor de Unity, y luego utiliza esos datos en tiempo de ejecución para determinar lo que la Cámara puede ver. El proceso de generar datos es conocido como *baking*.

4.6.3. Texturizado

En el proceso de texturizado se maximizó el espacio en el UV *map*, dado que el espacio de textura que no se utiliza ocupa parte de la memoria en el juego. Por lo general, un solo objeto tiene un solo material y un solo conjunto de texturas. Esto asegura que el objeto puede ser visualizado con una sola llamada de dibujo para un rendimiento óptimo. La principal ventaja del uso de Substance Painter fue proporcionar una única textura para diversos materiales, y dicho software facilitó crear en la mayoría de los casos un solo material por objeto.

4.6.4. Iluminación

En los videojuegos la iluminación es uno de los aspectos más importantes a considerar cuando se trata de rendimiento, sobre todo en cuanto a dispositivos móviles, dado que el coste de proporcionar iluminación en tiempo real es muy alto. Sin embargo, para lograr el efecto realista e inmersivo deseado, el juego debe contar con efectos realistas de iluminación y sombras. Para lograr esto, se utiliza el método de *Lightmapping*, lo cual guarda los datos de iluminación estáticos en un proceso llamado *baking* para ser utilizados en tiempo real en vez de computarlos por cada *frame*. Sin embargo, este método no percibe cambios de luz ni de sombra en los objetos dinámicos. Por tanto, la escena cuenta con *Light probes* los cuales guardan información de iluminación en distintos puntos de la Escena.

Capítulo 5

Prueba de Concepto

Con el objetivo de comprobar tanto la jugabilidad como la inmersión en VR, se reunió a un grupo de usuarios a los que se les pidió interactuar con la aplicación para luego aplicarles un cuestionario.

5.1. Metodología

Para la realización de las pruebas se utilizó el protocolo Thinking Aloud, el cual consiste en que el usuario realizando la prueba piense continuamente en voz alta mientras utiliza el sistema. Al verbalizar sus pensamientos, los usuarios de prueba permiten entender cómo ven el sistema, lo que facilita la identificación de los principales conceptos erróneos de los usuarios finales. El tamaño de la muestra está en correspondencia con lo requerido para aplicar dicha técnica, de acuerdo a lo que plantea Holzinger et al. [23]. Se pide a los participantes que digan todo lo que se les ocurra mientras realizan la tarea.

5.1.1. Participantes

Se seleccionó un grupo de 5 personas, compuesto por estudiantes de edades entre 20 y 26 años, dado que se encuentran dentro del rango del público objetivo del juego desarrollado.

5.1.2. Equipo

Las pruebas fueron realizadas con un único visor de realidad virtual Oculus Quest conectado a un notebook con sistema operativo Windows 10 y la aplicación de Oculus Link, que permite correr las aplicaciones en el visor. La aplicación se inició desde el modo de juego de Unity para poder ver en la pantalla del computador el campo de visión del usuario dentro del juego.

5.1.3. Procedimiento

Los pasos seguidos para la realización de la prueba consistieron en primero explicar a cada participante el contexto de la creación del juego, es decir, que el juego había sido desarrollado para una memoria y que consistía en un juego VR de simulador de mascota el cual se juega a través del HMD de Oculus Quest. En primer lugar se realizó la configuración del Oculus siguiendo los pasos que se detallan en la guía de *Primeros pasos con Quest* [38]. Estos consisten en ajustar el visor y la vista. La conexión con los controles se realiza solo una vez y luego estos se conectan automáticamente.

Posteriormente se les pide a los usuarios realizar el tutorial *First steps* [37] proporcionado por Oculus. Dicho tutorial presenta una breve experiencia que muestra cómo utilizar los controles básicos para manipular objetos. Luego, se procedió a explicar los mandos y sus funciones dentro del juego.

Una vez finalizado el proceso de familiarización con el HMD y sus controles se procedió a iniciar el juego. Se observó la realización de la prueba a través de la pantalla del computador con el fin de monitorear el cumplimiento de las tareas solicitadas. En simultáneo con la ejecución de las tareas, se registró la cantidad y tipo de errores, qué tan críticos son y cuánto tiempo se demoró en cumplir cada tarea. Además, se fue registrando lo que el usuario decía acerca de cómo realizaba las tareas en voz alta. Estas tareas corresponden a las acciones e interacciones básicas que cubren la totalidad de la funcionalidad desarrollada en la aplicación, las cuales se describen a continuación:

1. Contestar cuestionario de estados de ánimo
2. Teletransportarse a una zona válida
3. Usar movimiento continuo para navegar el entorno
4. Agarrar y soltar algún objeto dentro del entorno
5. Llamar a la mascota
6. Abrir la puerta y salir al exterior de la casa
7. Cumplir las necesidades de la mascota a medida que las vaya requiriendo: acariciarla, alimentarla, jugar y hacerla dormir

Al final de la prueba se proporcionaron 5 minutos de exploración libre con el propósito de registrar comportamientos e interacciones interesantes para considerar en los futuros cambios o complementos en cuanto a funcionalidad y jugabilidad al desarrollo del juego. Finalmente se le agradeció su participación a los participantes y se les pidió contestar una encuesta implementada en Google Forms.

La encuesta solicitada a los usuarios consistió de cuatro secciones. El primer módulo se conformó de la información personal del usuario como el nombre, género, edad, la cantidad de horas que juega videojuegos a la semana y cuanta experiencia posee con juegos en VR. Las siguientes dos secciones correspondieron a una traducción del módulo base y post juego de *The Game Experience Questionnaire* [26], un cuestionario ampliamente utilizado para medir experiencias de juego en investigación académica e industria de Computación Centrada en las Personas. Los módulos del cuestionario deben administrarse inmediatamente después de

terminar la sesión de juego.

Finalmente, la última sección consistió de 3 módulos. El primero mide la satisfacción con el juego, la cual consiste en preguntar acerca de la dificultad percibida en completar cada tarea. El segundo módulo, que evalúa de una escala del 1 al 5 qué tan motivante, entretenido e inmersivo resultó el juego y si lo volvería a jugar, en conjunto con una sección de justificación a las respuestas proporcionadas. Por último, el tercer módulo consistió en preguntas de respuesta libre, las que permiten saber qué aspectos mejoraría, que nuevas mecánicas o características agregaría o si hubo mecánicas que resultaron confusas y una sección de comentarios y sugerencias adicionales. El cuestionario realizado a los usuarios de prueba se encuentra detallada en su totalidad en el Sección 6. En el cual el módulo base y el módulo post juego son una versión traducida de un extracto del *The Game Experience Questionnaire*, y el módulo de satisfacción y de comentarios son de autoría propia.

5.1.4. Recolección y análisis de datos

De la prueba de concepto se recolectó datos de dos fuentes. La primera corresponde a apuntes tomados de la retroalimentación y verbalización de ideas de los participantes a medida que interactuaban con la aplicación. Y la segunda, el cuestionario de usabilidad realizado al finalizar la sesión.

Dado los datos recolectados, se analizó la usabilidad y utilidad percibida del juego. Se analizaron cualitativamente los comentarios proporcionados por los participantes durante la prueba y las respuestas proporcionadas por ellos en la encuesta de salida. Asimismo, se buscó contrastar las opiniones entre los participantes asignados en base a sus experiencias anteriores con VR y con videojuegos en general.

5.2. Resultados

Para la prueba de concepto se reclutaron N = 5 participantes. Dado el bajo número de usuarios de prueba los resultados no son generalizables y no es posible realizar un estudio significativo de los valores obtenidos en promedio para el grupo de prueba. Dado esto, se realizó un análisis cualitativos de los datos recolectados y se hizo énfasis en los comentarios sobre la experiencia de juego y sugerencias en las respuestas de los participantes, tanto en el momento de la prueba como en la encuesta realizada.

A continuación se presentan los resultados principales obtenidos durante la prueba de concepto.

5.2.1. Usabilidad y utilidad percibida

Dentro de los comentarios recibidos por los usuarios durante la prueba se destaca que la mayoría encontró más cómodo usar el modo de teletransportarse en vez del movimiento continuo, ya que este generaba un poco de desconcierto o malestar, esto se dio particularmente en los usuarios sin previa experiencia en VR. De esto se puede rescatar lo importante que fue proporcionar dos opciones de movimiento dentro del entorno VR para que el jugador pueda elegir el que más le acomode.

Además, se pudo notar que pese a estar haciendo otras interacciones la mayoría de los participantes mantenían activado el *teleport* a cada momento dado que al ser *touch* no se acostumbraban a soltar el *thumbstick* del control. Dado esto, se concederá asignar un mayor valor de *Activation threshold* al control de la locomoción o cambiar el mapeo de la función del *teleport* a un input *clickeable* en vez del *touch* que se utiliza actualmente.

Por otro lado, de los datos recolectados en la etapa de exploración libre del juego se tiene que lo más destacable fue el hecho de que cada participante intentó realizar acciones que se podrían realizar en la vida real, como por ejemplo abrir la llave de agua de la cocina, abrir el resto de puertas de los gabinetes de la cocina o tomar las sillas. Esto, por un lado, demuestra lo inmersivo que resultó ser el ambiente ya que se esperaban este tipo de funcionamientos realistas. Por otro lado, dichas interacciones se pueden considerar para agregar a futuras versiones del juego. Dada la base de la aplicación implementada, dicha tarea es bastante directa, se puede añadir la misma función de movimiento que utiliza la puerta principal al resto de las puertas y cajones de la *Escena*, y para la mecánica de abrir la llave de agua esto se puede lograr creando un efecto con *shaders*, con las cuales se está familiarizado ya que se utilizan en el efecto de la zona del teleport.

En cuanto a las interacciones con la mascota, estas resultaron de la manera esperada en todos los casos de prueba. Algunos usuarios tardaron un poco más en darse cuenta cómo hacer dormir a la mascota o cómo alimentarla, pero todos lograron cumplir con las tareas de manera satisfactoria. Los mensajes que la mascota proporcionaba acerca de sus necesidades también fueron comprendidos por los usuarios. En la sección de juego libre, se puede destacar que la mayoría de los usuarios decían en voz alta lo que la mascota necesitaba e inmediatamente buscaban cómo complacerla. Sin embargo, hubo un menor numero de usuarios que se vieron absortos por el ambiente, el paisaje exterior y los objetos de la *Escena* de tal forma que olvidaron por un momento la presencia de la mascota. Para poder evitar esta situación en el futuro se plantea agregar sonidos provenientes de la mascota y además hacer que esta pueda volar hasta la altura del jugador, aprovechando sus cualidades de criatura fantástica, ya que al estar a nivel del suelo es mas propensa a pasar desapercibida.

Finalmente, se tuvo que la recepción general de los usuarios fue bastante favorable. Cabe destacar que se observó que los usuarios se veían motivados y entretenidos tanto por explorar la casa, el nivel principal del juego, y por las reacciones que producían en el dragón mascota, lo cual los motivaba a seguir interactuando con él. Desde el punto de vista de las características ambientales del juego, los usuarios expresaron que la música ambiental reproducida les resultaba relajante junto a los colores cálidos del ambiente y el paisaje natural observable desde el exterior de la casa. Asimismo, les resultó fascinante el cambio de iluminación con el

pasar del tiempo el cual muestra diversos ambientes dependiendo de si es en la mañana, el atardecer o la noche.

Asimismo, de los comentarios obtenidos en los resultados de la encuesta aplicada se tiene que la mayoría encontró motivante, entretenido e inmersivo el juego, y la totalidad de los participantes lo volverían a jugar. A modo de justificación, la mayoría se vio motivado por la retroalimentación positiva que daba la mascota al cuidarla, esto es, las necesidades expresadas por la mascota acompañante impulsan a mantener una constante atención a su estado (representado por el corazón) y a atender sus constantes necesidades, dando así un objetivo y una meta al usuario, que seria el mantener el bienestar del acompañante que posee dentro del juego.

Por otro lado, con respecto a la inmersión, comentaron que la realidad virtual permite estar inmerso dentro del mundo del juego, junto al interés que se genera por querer cuidar al dragón mascota. Además, sentían estar en un espacio del tamaño de la habitación simulada. Cabe destacar que el ambiente también fue un punto señalado por los usuarios, estéticamente el escenario de un hogar aislado en la naturaleza sin vista de otras distracciones permite al jugador enfocarse plenamente en las interacciones de su propio hogar y de su mascota.

En cuanto a los aspectos a mejorar, la mayoría de los participantes mencionó que la mascota podría ser más notoria. En particular, se comentó que sería de utilidad que la mascota emitiera sonidos para comunicarse con el usuario, a modo de analogía de una mascota real, de tal modo que los sonidos ayuden a mantener la atención en sus necesidades. Además, se podría hacer que la mascota volara a una mayor altura, para así facilitar la interacción con él, ya que actualmente hay que bajar al nivel cercano al suelo para interactuar con él.

Con respecto a los comentarios de sugerencias para agregar al juego en las siguientes versiones, se destacan los siguientes:

- La adición de opciones de customización de la mascota, ya sea color o distintos accesorios. La personalización permite que los usuarios se comprometan e involucren más con las mecánicas del juego, y en el caso del juego, en el cuidado de la mascota.
- La adición de moneda virtual y tiendas, ya sea que vendan alimentos u objetos para el hogar del jugador. Esto aumentaría en gran medida la jugabilidad y la satisfacción dado la sensación de recompensa.
- Se podría implementar un reloj, ya sea en la interfaz o uno físico dentro de la casa, de tal manera de tener algo de noción del pasar del tiempo, o como alternativa, disponer de un cronómetro para así asignar previamente el tiempo de juego. Esto es, dado que en VR el jugador se desconecta con la realidad esto puede ocasionar que al jugarlo por muchas horas se generen problemas para algunos usuarios.
- Se podría indicar con algún brillo, o borde de acuerdo a la cercanía, a los elementos con los cuales se puede interactuar. Dado que el no poder interactuar con algún objeto en el entorno parece producir cierto nivel de frustración en el usuario.

5.3. Discusión

A partir de los resultados, fue posible validar el diseño del videojuego en realidad virtual de simulación de cuidado de mascota virtual. A pesar de que los resultados no son estadísticamente significativos, y que el tamaño de la muestra fue de sólo 5 participantes, esto no impidió que se lograra observar la usabilidad y jugabilidad percibida.

Si bien no se logró llevar a cabo un estudio a mayor escala el cual permitiera realizar un seguimiento y validar si efectivamente el juego en realidad virtual implementado logra regular los niveles de soledad en los estudiantes universitarios, sí se observó una clara tendencia a la sensación de bienestar entre los participantes al jugar el juego, ya sea por lo estéticamente agradable de su ambiente como por las mecánicas ofrecidas para interactuar con la mascota. En los resultados del cuestionario hubo una clara tendencia a indicar el juego como una experiencia enriquecedora y que hizo que perdieran la noción del mundo exterior, esto apoya la posibilidad de que el juego pueda servir de ayuda terapéutica en futuras versiones.

En relación a los objetivos específicos propuestos inicialmente, es posible analizar su cumplimiento dados los resultados obtenidos. Respecto al primer objetivo, se logró escoger y conseguir un visor VR que cumplía con las características propuestas, esto es con el Oculus Quest 1. Con dicho visor fue posible implementar el juego en VR en Unity y realizar las pruebas sin problemas. Con respecto al segundo objetivo, se logró validar la experiencia de juego inmersiva en VR a través de los comentarios obtenidos de la prueba de concepto. El tercer objetivo se logró validar también con los comentarios de los participantes de las pruebas que señalaron lo motivante que resultó la retroalimentación proporcionada por los iconos de la mascota. Por otro lado, el cuarto objetivo específico se validó dado que los usuarios consideraron realista y agradable visualmente el entorno proporcionado por el juego. Finalmente, el quinto objetivo se validó demostrando durante la prueba de concepto que tanto los controles como las mecánicas de jugabilidad fueron comprendidas y logradas de manera satisfactoria por los usuarios.

5.4. Limitaciones de la evaluación

Si bien los resultados obtenidos en esta memoria son valiosos, cabe destacar que no por ello están exentos de poner en duda su validez interna (es decir, si son confiables) y externa (es decir, si son generalizables). Debido a que el experimento fue realizado con un número pequeño de usuarios (N = 5), los resultados obtenidos no pueden ser generalizables a la población de estudio. No obstante lo anterior, dado el éxito relativo de los resultados obtenidos, se propone reclutar a una muestra más grande y representativa de participantes, para estudiar la validez de las hipótesis (i.e., regular el estado de soledad en estudiantes universitarios) en etapas posteriores de este trabajo.

Capítulo 6

Conclusión y Trabajo Futuro

A continuación se concluye sobre el trabajo realizado, además haciendo una retrospectiva sobre qué podría haberse hecho mejor, y qué otras facultades podrían agregarse al videojuego a futuro.

Durante el desarrollo del presente trabajo de título, se evidenció que implementar un videojuego en VR que cumpla con los estándares básicos tanto de usabilidad y jugabilidad como de rendimiento, no corresponde a un proceso que pueda ser desarrollado de manera inmediata puesto que requiere de una gran cantidad de fases. Estas no solo incluyen el proceso de diseño e implementación del juego, sino que también se deben preparar los *assets* de manera previa para que sean compatibles con las características esperadas de un juego en VR en cuanto al aspecto visual y sobre todo rendimiento. Esto último es de especial importancia pues se requiere también configurar y probar las conexiones del dispositivo en VR con el motor del juego y se deben planificar cuidadosamente las mecánicas e interacciones para que estas resulten intuitivas y familiares dentro de un entorno 3D en donde los límites entre lo que el jugador puede o no hacer se encuentran ligados al comportamiento del usuario en sí mismo y no a un modelo totalmente controlado por el programa.

De esta forma, trabajar de manera unísona dichos aspectos permitió obtener resultados favorables dentro de los parámetros establecidos. Para ello, en primer lugar fue necesario establecer las herramientas a utilizar, en específico el visor en VR en torno el cual se llevaría a cabo la implementación y las pruebas y los *software* de desarrollo para la implementación del videojuego y los *assets*. Luego, se realizó el proceso previo a la implementación, el cual consistió en el diseño del flujo del juego y de las mecánicas de interacción, además se prepararon los modelos 3D y las texturas. Asimismo, se desarrolló un sistema de estados para manejar las animaciones y los comportamientos de la mascota y se implementó el sistema de eventos utilizando el método de *Observer pattern* en base al cual funcionaría todo el juego. Finalmente, se aplicaron diversas técnicas de optimización entre las cuales se destacan el *baking* de los mapas de iluminación de la *Escena* y el*Occlusion culling*, con el fin de obtener un rendimiento de juego equivalente al requerido para el dispositivo de VR utilizado.

Con el objetivo de evaluar el trabajo realizado, se realizó una prueba de concepto con una muestra reducida de usuarios reales, los cuales cumplían con las características de usuarios

finales a los cuales estaba dirigido el videojuego desarrollado. Los resultados obtenidos, si bien no son generalizables dado el escaso tamaño de la muestra, fueron cualitativamente favorables lo cual se considera valioso en términos de una prueba de concepto. Se logró evidenciar durante las pruebas que los usuarios se mostraban motivados e inmersos en el ambiente y propósito del juego, y evidenciaron que les resultó una experiencia enriquecedora. Asimismo, los participantes demostraron su agrado general hacia el juego, tanto en el diseño del entorno VR y los elementos gráficos utilizados como en la jugabilidad ofrecida. En particular, destacaron su agrado con respecto a las expresiones y reacciones de la mascota y a la sensación de paso del tiempo pudiendo admirar un paisaje realista y apacible a la vista.

Finalmente, tanto el proceso de desarrollo como la evaluación permite concluir con respecto al trabajo presentado que se pudo cumplir con todos los objetivos propuestos. En particular, se programó un videojuego para un dispositivo VR cuya funcionalidad es cuidar y proteger a una mascota virtual. En base a las pruebas que se realizaron con usuarios pertenecientes al público objetivo se propuso en la discusión que los objetivos particulares presentados en la sección de objetivos fueron logrados con éxito, al observar que efectivamente los usuarios de prueba percibieron motivación e inmersión utilizando la aplicación presentada.

Debido a que el juego fue implementado en base a la plataforma Android y optimizado para funcionar en ella utilizando el *Universal render pipeline* de Unity, se puede tanto construir dentro del mismo Oculus Quest para funcionar de manera independiente como también puede jugarse conectándose al computador. Lo anterior permite otorgar la flexibilidad necesaria para adaptarse a los requerimientos del usuario objetivo.

Si bien se concluye que se lograron los objetivos con éxito, el juego implementado tiene mucho potencial en cuanto a la posible regulación en estado de soledad y ansiedad. Para ello, se desprenden distintos elementos que pueden ser utilizados para la realización de trabajos futuros.

En primer lugar, si bien se obtuvo una respuesta favorable por parte de la muestra escogida, para poder generalizar los resultados hacia la población de estudio es necesario replicar el experimento pero un mayor número de participantes y con un seguimiento en el tiempo, deseablemente diario, recolectando datos de sus estados de ánimo antes, durante y después de experimentar las experiencias de sesiones de juego, lo cual permita estudiar el efecto real conseguido por el juego.

En particular, como se puede observar en los comentarios dejados tras la prueba con usuarios, existen varios detalles que se pueden mejorar para entregar una mejor experiencia final. Entre estas se encuentran las mejoras en las mecánicas de movimiento y reducir confusión en el mapeo de los controles, agregar distintos elementos de customización tanto en la mascota como en la casa, proporcionar más autonomía y comodidad al momento de interactuar con el usuario en comportamiento de la mascota, ya sea agregando más animaciones y funciones a la máquina de estados de la mascota, como también haciendo que esta se acerque con más precisión al usuario elevándose hasta la altura de su visión.

También, se propone crear interacciones manipulables en VR para más objetos dentro de la *Escena*. Por ejemplo, permitir abrir las puertas y cajones visibles, abrir y cerrar cortinas, crear efectos como prender y apagar luces, abrir la llave del agua o encender fuego en la

cocina o elementos como una chimenea. Sin embargo, se debe evaluar cuidadosamente el sistema de orden de dichos objetos dado que al introducir tal nivel de libertad puede resultar contraproducente si se desordena la casa o los objetos se caen o se desaparecen de manera indeseada. Para ello, se propone agregar una funcionalidad de generar orden en la mascota, como por ejemplo mientras tenga un buen nivel de energía o felicidad se dedique a ordenar o devolver los objetos a su estado original. Esta funcionalidad, por otro lado, desprende una gran cantidad de posibilidades futuras como por ejemplo hacer un sistema que acumule polvo en los días de ausencia del usuario, lo cual genere un incentivo para que este visite diariamente su casa virtual.

En términos de efectos visuales y de sonido, si bien no afectan de manera directa a la jugabilidad se pueden agregar varias opciones que pueden enriquecer la experiencia de juego. Entre los efectos de audio, se encuentra agregar sonidos provenientes de la mascota para dar mayor sensación de presencia y compañía, y por otro lado agregar efectos de sonido a los elementos del entorno como las puertas o las colisiones entre los objetos. También se puede asignar un catálogo de música que suene dependiendo del estado de ánimo escogido por el usuario o del momento del día dentro del juego por ejemplo. Por otro lado, se pueden utilizar efectos visuales como partículas para añadir más fluidez a las reacciones de la mascota, como por ejemplo utilizar partículas de corazones emergentes al hacerle cariño o añadir un efecto de polvo que indique que esta requiere una limpieza.

De igual manera, se podrían introducir nuevos elementos al juego. Entre estos se encontraría generar una cinemática de introducción al inicio del juego en el cual se muestre el nacimiento de la mascota y se pueda elegir un nombre y quizás también su color. Introducir un tutorial al inicio del juego, el cual muestre los controles en el mundo virtual y señale las funciones de los distintos botones y cómo realizar las distintas interacciones con el entorno, así como también agregar una opción en el menú del juego que muestre el mapeo de los controles a modo de recordatorio. Generar un sistema de simulación de clima que se integre con el sistema de ciclo de día y noche, el cual permita simular climas fríos y cálidos en conjunto de efectos como lluvia, nieve, tormentas o arco iris. Dichos efectos de clima pueden programarse para estar ligados al estado de ánimo del usuario, o bien funcionar de manera independiente.

Otras ideas de trabajo futuro más ambiciosas consisten en agregar un sistema de recompensas y moneda virtual para luego poder utilizarlas en otros escenarios como distintas tiendas, e implementar un sistema específicamente dedicado a la edición del aspecto de la casa, permitiendo cambiar el diseño del piso y las paredes, y posicionar objetos desde lejos a través de *Ray interactor*. Por otro lado, implementar un sistema de ajustes dinámicos de las emociones del jugador durante la experiencia de juego para así realizar respuestas correspondientes a dicho estado tanto en el ambiente como en la mascota. Agregar una sala de juegos especial en donde se pueden implementar minijuegos más elaborados, como simular la mecánica de volar montando al dragón mascota, hacer que la mascota acierte con proyectiles de fuego en aros, entre otras muchas otras experiencias de juego que se pueden agregar.

Por otro lado, pese a que el objetivo del juego desarrollado está enfocado en la salud mental en estudiantes universitarios principalmente, es factible extender el rango de edades en el público objetivo para poder utilizarse por ejemplo en el caso de adultos mayores, los cuales suelen encontrarse en estados de soledad como se mencionó al inició del documento.

Finalmente, si bien este trabajo de título consistió en implementar un juego en VR de cuidado de una mascota virtual, la lógica y fundamentos expuestos en esta memoria pueden ser utilizados para construir un juego con el potencial de regular la percepción de soledad en estudiantes universitarios, como también reducir su estrés y ansiedad. Para esto, el software quedará a disposición del profesor guía del presente trabajo para que otras personas lo puedan utilizar y continuar expandiéndolo.

Bibliografía

[1] Abt, C. C. Serious games. *University press of America.* (1987).

[2] Alton, L. Cheaper vr is coming: will it be enough to kick-start consumer interest? *thenextweb [Web Blog]* (2018).

[3] Antunes, T. P. C., De Oliveira, A. S. B., Crocetta, T. B., de Lima Antão, J. Y. F., de Almeida Barbosa, R. T., Guarnieri, R., and De Abreu, L. C. Computer classes and games in virtual reality environment to reduce loneliness among students of an elderly reference center: Study protocol for a randomised cross-over design. *Medicine 96(10)* (2017).

[4] Baader, T., Rojas, C., Molina, J. L., Gotelli, M., Alamo, C., Fierro, C., and Dittus, P. Diagnóstico de la prevalencia de trastornos de la salud mental en estudiantes universitarios y los factores de riesgo emocionales asociados. *Revista chilena de neuro-psiquiatría 52(3)* (2014), 167–176.

[5] Baecker, A. N., Geiskkovitch, D. Y., González, A. L., and Young, J. E. Emotional support domestic robots for healthy older adults: Conversational prototypes to help with loneliness. *In Companion of the 2020 ACM/IEEE International Conference on Human-Robot Interaction* (2020), 122–124.

[6] Baecker, R., Sellen, K., Crosskey, S., Boscart, V., and Barbosa Neves, B. Technology to reduce social isolation and loneliness. *In Proceedings of the 16th international ACM SIGACCESS conference on Computers & accessibility* (2014), 27–34.

[7] Baron, D. *Hands-On Game Development Patterns with Unity 2019: Create engaging games by using industry-standard design patterns with C.* Packt Publishing Ltd, 2019.

[8] Batada, A., and Leon Solano, R. Harnessing technology to address the global mental health crisis. *World Bank* (2019).

[9] BenJones. Twilight princess eyes breakdowne. `https://www.benjones.us/twilight-princess-eyes-breakdown/` (Último acceso 05 Abril 2021).

[10] Birk, M. V., Wadley, G., Abeele, V. V., Mandryk, R., and Torous, J. Video games for mental health. *interactions 26(4)* (2019), 32–36.

[11] Brown, B. C. Why virtual reality is more accessible now than ever before. *highrockstudios [Web Blog]* (2018).

[12] Calleja, G. In-game: From immersion to incorporation. *mit Press* (2011).

[13] Cigna. Survey of 20.000 americans examining behaviors driving loneliness in the united states, 2018.

[14] Circuitstream. Htc vive vs oculus: An in-depth guide on which headset is better for business and personal use. `https://circuitstream.com/blog/htc-vs-oculus/#:~:text=Both%20HTC%20headsets%20are%20slightly,visual%20quality%20for%20the%20price.` (Último acceso 05 Abril 2021).

[15] Dekker, M. R., and Williams, A. D. The use of user-centered participatory design in serious games for anxiety and depression. *Games for health journal 6(6)* (2017), 327–333.

[16] Demes, L. *Assets from CC0Textures.com, licensed under CC0 1.0 Universal.*, 2021. `https://cc0textures.com/` (Último acceso 31 Marzo 2021).

[17] Denisevicz, S., and Zhu, J. Interweaving narrative and gameplay to cultivate empathy for anxiety and depression. *In Extended Abstracts of the Annual Symposium on Computer-Human Interaction in Play Companion Extended Abstracts* (2019), 247–252.

[18] Diehl, K., Jansen, C., Ishchanova, K., and Hilger-Kolb, J. Loneliness at universities: Determinants of emotional and social loneliness among students. *International journal of environmental research and public health 15(9)* (2018), 1865.

[19] Funk, M., Müller, F., Fendrich, M., Shene, M., Kolvenbach, M., Dobbertin, N., Günther, S., and Mühlhäuser, M. Assessing the accuracy of point & teleport locomotion with orientation indication for virtual reality using curved trajectories. 1–12.

[20] Geslin, E., Jégou, L., and Beaudoin, D. How color properties can be used to elicit emotions in video games. *International Journal of Computer Games Technology 2016* (2016).

[21] Granic, I., Lobel, A., and Engels, R. C. The benefits of playing video games. *American psychologist 69*, 1 (2014), 66.

[22] Holt-Lunstad, J. The potential public health relevance of social isolation and loneliness: Prevalence, epidemiology, and risk factors. *Public Policy & Aging Report 27(4)* (2017), 127–130.

[23] Holzinger, A. Usability engineering methods for software developers. *Communications of the ACM 48*, 1 (2005), 71–74.

[24] House, J. S. Isolation kills, but how and why? *Psychosomatic Medicine 63(2)* (2001), 273–274.

[25] Hämmig, O. Health risks associated with social isolation in general and in young, middle and old age. *PloS one 14(7)* (2019).

[26] IJsselsteijn, W. A., de Kort, Y. A., and Poels, K. The game experience questionnaire. *Eindhoven: Technische Universiteit Eindhoven 46*, 1 (2013).

[27] Inc, F. S. *Fesliyan Studio, Royalty Free Music And Sound Effect.* https://www.fesliyanstudios.com/royalty-free-music/download/serenity/897 (Último acceso 31 Marzo 2021).

[28] Injuv. 9°encuesta nacional de juventud, 2018.

[29] Jonker, C. M., and Treur, J. Agent-based simulation of animal behaviour. *Applied Intelligence 15*, 2 (2001), 83–115.

[30] Joosten, E., Van Lankveld, G., and Spronck, P. Colors and emotions in video games. In *11th International Conference on Intelligent Games and Simulation GAME-ON* (2010), pp. 61–65.

[31] Kruger, K. A., and Serpell, J. A. Animal-assisted interventions in mental health: Definitions and theoretical foundations. In *Handbook on animal-assisted therapy*. Elsevier, 2010, pp. 33–48.

[32] Lasgaard, M., Friis, K., and Shevlin, M. "where are all the lonely people?" a population-based study of high-risk groups across the life span. *Social psychiatry and psychiatric epidemiology 51*, 10 (2016), 1373–1384.

[33] Liszio, S., Emmerich, K., and Masuch, M. The influence of social entities in virtual reality games on player experience and immersion. *In Proceedings of the 12th International Conference on the Foundations of Digital Games* (2017), 1–10.

[34] LTD, D. *3dsky.org, service for sharing and selling 3D models.*, 2021. https://3dsky.org/ (Último acceso 31 Marzo 2021).

[35] McNicholas, J., and Collis, G. M. Animals as social supports: Insights for understanding animal-assisted therapy. *Handbook on animal-assisted therapy: Theoretical foundations and guidelines for practice 2* (2006), 49–72.

[36] Micin, S., and Bagladi, V. Salud mental en estudiantes universitarios: Incidencia de psicopatología y antecedentes de conducta suicida en población que acude a un servicio de salud estudiantil. *Terapia psicológica 29(1)* (2011), 53–64.

[37] Oculus. First steps. https://www.oculus.com/experiences/quest/1863547050392688/?locale=es_ES (Último acceso 06 Abril 2021).

[38] Oculus. Primeros pasos con quest. https://support.oculus.com/855551644803876/?locale=es%5C_LA (Último acceso 06 Abril 2021).

[39] Oculus VR, Facebook Technologies, L. *Oculus Testing and Performance Analy-*

sis, 2021. https://developer.oculus.com/documentation/unity/unity-perf/ (Último acceso 31 Marzo 2021).

[40] Robertson, A. "it's 2019 - which vr headsets can you actually buy?". *The Verge [Web Blog]* (2019).

[41] S.L, F. C. *Freepik, plataforma en la que los diseñadores pudieran encontrar recursos gráficos gratuitos.* https://www.freepik.es/ (Último acceso 31 Marzo 2021).

[42] Torous, J., Wolters, M. K., Wadley, G., and Calvo, R. A. 4th symposium on computing and mental health: Designing ethical emental health services. *In Extended Abstracts of the 2019 CHI Conference on Human Factors in Computing Systems* (2019), 1–9.

[43] TurboSquid. *TurboSquid, The World's Source for Professional 3D Models.* https://www.turbosquid.com/ (Último acceso 31 Marzo 2021).

[44] Unity. *About Shader Graph*, 2021. https://docs.unity3d.com/Packages/com.unity.shadergraph@6.9/manual/index.html (Último acceso 8 Marzo 2021).

[45] Unity. *Building a NavMesh*, 2021. https://docs.unity3d.com/Manual/nav-BuildingNavMesh.html (Último acceso 31 Marzo 2021).

[46] Unity. *Colliders*, 2021. https://docs.unity3d.com/Manual/CollidersOverview.html (Último acceso 31 Marzo 2021).

[47] Unity. *Nav Mesh Agent*, 2021. https://docs.unity3d.com/Manual/class-NavMeshAgent.html (Último acceso 31 Marzo 2021).

[48] Unity. *Occlusion culling*, 2021. https://docs.unity3d.com/Manual/OcclusionCulling.html (Último acceso 31 Marzo 2021).

[49] Unity. *Unity XR Input*, 2021. https://docs.unity3d.com/Manual/xr_input.html (Último acceso 31 Marzo 2021).

[50] Unity. *XR Plug-in Framework*, 2021. https://docs.unity3d.com/Manual/XRPluginArchitecture.html (Último acceso 31 Marzo 2021).

[51] Unity. Unity asset store. https://assetstore.unity.com/ (Último acceso 31 Marzo 2021).

[52] Young, J. S. Pet therapy: Dogs de-stress students. *Journal of Christian Nursing 29(4)* (2012), 217–221.

Apéndices

Apéndice A: Atribución de los *Assets* del Juego

1. "Texturas PBR" por *CC0Textures.com*, con licencia CC0 1.0 Universal
2. "Audio: Serenity" por David Renda, con licencia Royalty Free
3. "Imagen: Emoticonos vector" por *Freepik*, con licencia de *Freepik*
4. "Grass And Flowers Pack 1" por *@Vladislav Pochezhertsev* en *assetstore.unity.com*, con licencia de *Standard Unity Asset Store EULA*
5. "Fantasy Skybox FREE" por *@Render Knight* en *assetstore.unity.com*, con licencia de *Standard Unity Asset Store EULA*
6. "Modelo 3D: Dining Chair by IKEA" por *@a313* en *3dsky.org*, con licencia Royalty Free
7. "Modelo 3D: feyka" por *@Vikinger* en *3dsky.org*, con licencia Royalty Free
8. "Modelo 3D: Tent" por *@Valufka* en *3dsky.org*, con licencia Royalty Free
9. "Modelo 3D: Riva side Table by Jasper Morrison", por *@Arminradan* en *turbosquid.com*, con licencia *Standard (Editorial Uses Only)* de *turbosquid*
10. "Modelo 3D: RG Model Pack 01 model, floor lamp", por *@ryangregory* en *turbosquid.com*, con licencia *Standard (Editorial Uses Only)* de *turbosquid*
11. "Modelo 3D: Saucepan", por *@3d_conjurer* en *turbosquid.com*, con licencia *Standard* de *turbosquid*
12. "Modelo 3D: Forged table", por *@Elena_Shvets* en *turbosquid.com*, con licencia *Standard* de *turbosquid*
13. "Modelo 3D: Houseplant", por *@Ishtvan* en *turbosquid.com*, con licencia *Standard* de *turbosquid*
14. "Modelo 3D: Collar Brewing Collection", por *@dariopinat* en *turbosquid.com*, con licencia *Standard* de *turbosquid*
15. "Modelo 3D: Couch 3D", por *@donnichols* en *turbosquid.com*, con licencia *Standard* de *turbosquid*
16. "Modelo 3D: Modern kitchen", por *@os_car* en *turbosquid.com*, con licencia *Standard* de *turbosquid*

Apéndice B: Cuestionario de Experiencia de Juego

Sólo se presentan los módulos del cuestionario [26] 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	Mas o menos	Bastante	Totalmente
0	1	2	3	4

1. Me sentí contento
2. Me sentí hábil
3. Estuve interesado en su propósito
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	Mas 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

Módulo de satisfacción

Para cada uno de los siguiente puntos indique su percepción de cómo encontró el juego, usando la siguiente escala::

Para nada	Un poco	Mas o menos	Bastante	Totalmente
0	1	2	3	4

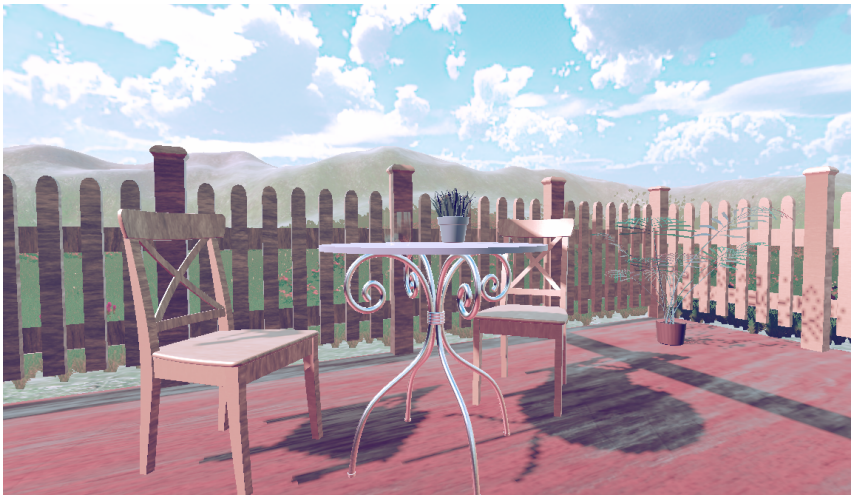
1. Motivante (Justifique)
2. Entretenido (Justifique)
3. Inmersivo (Justifique)
4. Lo volvería a jugar (Justifique)

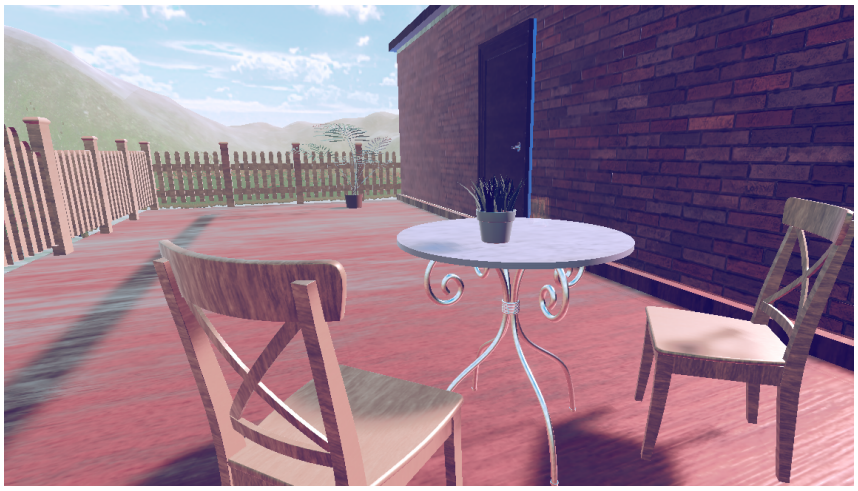
Módulo de comentarios y sugerencias

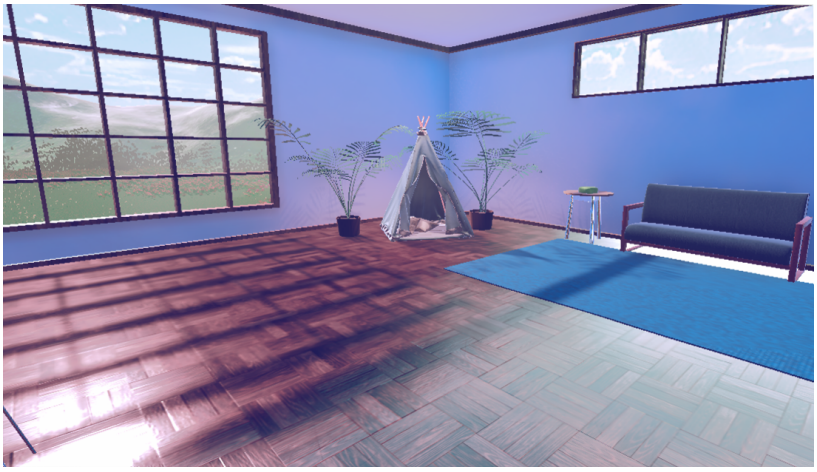
1. ¿Qué nuevos elementos o características le gustaría ver en el juego?
2. ¿Qué cambios le realizaría a las características actuales del juego?
3. ¿Hubo mecánicas confusas durante el juego? ¿Cuáles?
4. Comentarios adicionales

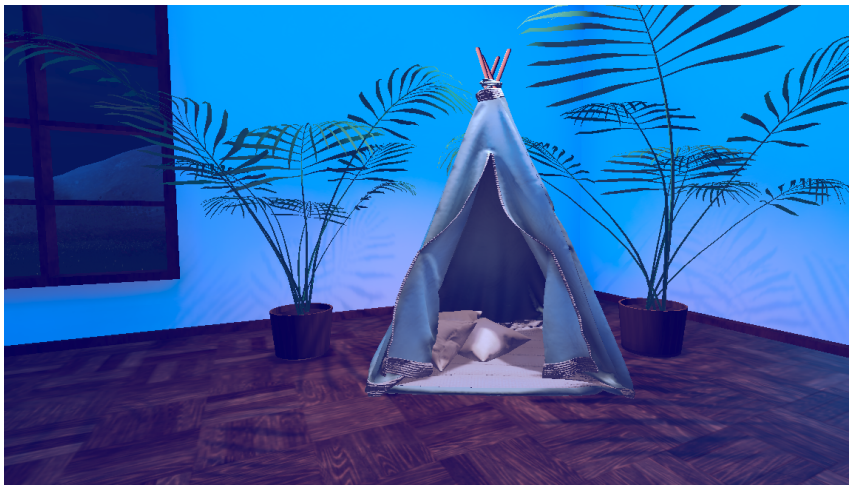
Apéndice C: Ambiente del Juego

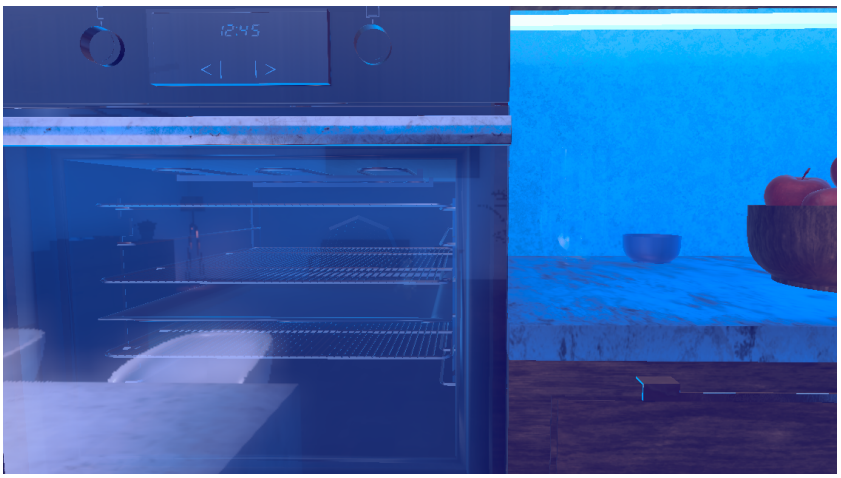
Se presentan imágenes tomadas dentro del juego del ambiente interior y exterior en distintos momentos del día.










12:45