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

IMPLEMENTANDO INTERACCIÓN GESTUAL EN MUNDOS DE REALIDAD VIRTUAL APLICADO A UN MUSEO

MEMORIA PARA OPTAR AL TÍTULO DE
INGENIERO CIVIL EN COMPUTACIÓN

RODRIGO ELÍAS VERA ALVARADO

PROFESOR GUÍA:
NELSON BALOIAN TATARYAN

MIEMBROS DE LA COMISIÓN:
JOSÉ PINO URTUBIA
PATRICIO INOSTROZA FAJARDIN

SANTIAGO DE CHILE
2024

Resumen

Los rápidos avances tecnológicos del último tiempo han permitido a instituciones como museos la búsqueda de alternativas y mejoras a las exhibiciones físicas. En particular, el uso de distintas tecnologías ha permitido la implementación de museos virtuales, que cuentan con algunas diferencias con respecto a las exhibiciones físicas tradicionales. Con estos museos virtuales se busca mejorar la experiencia de los usuarios, sobre todo en lo que respecta a la accesibilidad e inmersión. Por otro lado, en el ámbito de las interfaces de usuario también han habido avances, con diversas tecnologías que buscan lograr que las interacciones sean más intuitivas, accesibles, etc. Una de estas tecnologías involucra el uso de interfaces *touchless*, es decir, interfaces que no requieren la interacción física del usuario. En el presente trabajo de título se buscó incorporar tecnología *touchless* a un museo virtual existente.

La primera parte de este trabajo consistió en estudiar cuáles dispositivos *touchless* existen en el mercado, de forma de poder elegir el más apropiado para el proyecto.

Una vez elegido el dispositivo, se procedió a diseñar los gestos e interacciones a utilizar en la nueva versión del museo. Esta fase involucró la identificación y clasificación de las interacciones presentes en la versión original del museo, así como un estudio más detallado de las capacidades del dispositivo para tener mayor claridad sobre las posibilidades ofrecidas por él.

La solución consistió en la implementación de los gestos e interacciones diseñados, reemplazando las interacciones presentes en la versión original. El resultado obtenido es que el 100 por ciento de las interacciones identificadas en la versión original del museo son posibles de realizar mediante gestos de las manos o movimientos de la cabeza por parte del usuario. Sin embargo, en la fase de evaluación del proyecto se detectaron algunos problemas que se espera poder corregir en el futuro.

Tabla de Contenido

Índice de Tablas

Índice de Ilustraciones

Capítulo 1

Introducción

Los museos tienen un rol importante en la educación: permiten transmitir conocimiento mediante exposiciones de objetos. Esta forma de transmitir conocimiento tiene sus limitaciones: usualmente, solo es posible interactuar con los objetos "a distancia", es decir, observarlos de la forma que el museo los dispone. Otra limitación es la ubicación física del museo. Una persona interesada en ver los objetos de un museo no necesariamente puede asistir a cualquier museo que le interese.

Dadas estas limitaciones, los Museos Virtuales ofrecen valor adicional. Por un lado, la ubicación física del museo deja de ser un factor. Además, la naturaleza virtual del Museo Virtual permite una interacción más directa con los objetos exhibidos en éste, e incluso algunos permiten el armado de exhibiciones personalizadas [1, 2].

Actualmente, existen muchos museos que han digitalizado parte de sus colecciones. Algunos presentan dichas colecciones mediante páginas web (por ejemplo, el Virtual Museum of Canada [3]), mientras que otros utilizan modelos 3D (por ejemplo, la colección de modelos 3D del British Museum [4]). Independiente del método utilizado, lo importante es que la experiencia del usuario termine siendo interesante e inmersiva.

En un museo, las exhibiciones tienen como objetivo contar una historia: la selección y disposición específica de los objetos en la exhibición muestran aspectos particulares de la historia. Por ejemplo, una exhibición que busca mostrar la evolución histórica de los moais en Isla de Pascua dispondrá los moais lado a lado, en orden cronológico de construcción. A esta característica de las exhibiciones se le llama *storytelling* [5].

El permitirle a un usuario que construya sus propias exhibiciones en un Museo de Realidad Virtual que contenga objetos reales digitalizados en 3D, logra que dicho usuario pase de un rol de espectador a un rol más activo, que permite crear artefactos y compartirlos con otros visitantes virtuales. Esto implementa el principio de aprendizaje basado en el constructivismo [6].

Dada la importancia de la interacción en un Museo Virtual, queremos explorar la posibilidad de mejorar dicha interacción, especialmente en aquellos Museos que exhiben objetos 3D. Específicamente, queremos explorar el caso en que el museo se presenta como ambiente

de Realidad Virtual, donde el usuario puede interactuar directamente con los objetos, no solo para explorarlos, sino que permitiendo el armado de colecciones personalizadas.

La motivación para este trabajo deriva de la existencia de una implementación parcial de un Museo Virtual [7] de Khatchkars, que son piedras talladas con una imagen de una cruz, populares en la cultura armenia (ver Figura 1.1). En su versión original, este Museo cuenta con cinco salas de exhibición, donde mediante el uso de mouse y teclado el usuario puede desplazarse dentro de la sala para examinar las piedras, que corresponden a modelos 3D de piedras reales. Además, el museo permite seleccionar piedras para obtener más información sobre ellas. Finalmente, permite instanciar otras piedras para armar una colección personalizada.

Sin embargo, pruebas informales que se han hecho sobre esta aplicación indican que la forma de interactuar implementada usando mouse y teclado no es del todo cómoda para los usuarios, lo que desalienta su uso. Particularmente, en el año 2020 se realizó una sesión de evaluación de la aplicación en la Universidad Americana de Yerevan, en Armenia. Dicha evaluación consistió en dos partes. En primer lugar, se realizó una presentación inicial de la aplicación. A continuación, se invitó a los asistentes a interactuar con el software. Finalmente, se le pidió a los asistentes llenar un cuestionario para evaluar su experiencia. Este cuestionario fue contestado por 17 personas. Los resultados muestran que, si bien los usuarios calificaron en forma positiva la experiencia completa, un número importante de usuarios no encontró particularmente fácil la interacción con el software. Los resultados completos pueden verse en el anexo A. Si bien no se especifican las dificultades concretas con las que se encontraron los usuarios, la existencia de dificultades en la interacción lleva a explorar distintas formas de corregir estos problemas y mejorar la experiencia del usuario en este museo.

El objetivo principal de este trabajo tiene dos componentes. El primero es reemplazar la interacción humano-computador basada en mouse y teclado por interacción basada en gestos con las manos. El segundo es permitir la visualización del museo utilizando tecnologías de Realidad Virtual. Esto permitirá que en un eventual trabajo futuro sea posible evaluar la experiencia del usuario y compararla con la versión original.

Si bien existen otras tecnologías que en teoría permitirían mejorar las experiencias e interacciones del usuario, se optó por utilizar tecnología de captura de gestos. Esto porque el proyecto de agregar otra forma de interacción es parte de una serie de trabajos que buscan investigar y evaluar la interacción mediante gestos.

El Museo Virtual

Actualmente existe una implementación parcial de un Museo Virtual de Khatchkars. La aplicación está implementada en la versión 2018.3 de Unity [8], que es un motor gráfico multiplataforma para el desarrollo de videojuegos. Unity permite desarrollar videojuegos para Microsoft Windows, Linux, Mac, Android, iOS, WebGL, entre otros. Además, permite desarrollar videojuegos en 2D, 3D, Realidad Virtual, Realidad Aumentada, etc. Unity también permite el uso de scripts en C#. La aplicación del museo está implementada en WebGL, lo que permite al usuario acceder a ella desde cualquier navegador compatible con esta tecnología.

Figura 1.1: Khatchkars en la realidad

En esta aplicación el usuario tiene la posibilidad de elegir entre 5 ambientes o "salas de exhibición" distintas, las que se muestran con un icono al ingresar a la aplicación (ver Figura 1.2). Una vez dentro de una sala, el usuario puede realizar las siguientes acciones:

- Moverse dentro de la sala utilizando las teclas "W" (para avanzar), "S" (para retroceder), "A" (para desplazarse a la izquierda) y "D" (para desplazarse a la derecha) (ver Figura 1.3). Esta perspectiva de la Sala también se llama Vista Principal.
- Cambiar hacia dónde se está mirando ("mover la cámara") mediante movimientos del mouse.
- Examinar y obtener mayor información de los Khatchkars específicos presentes en la sala (ver Figura 1.4). En particular, mediante gestos con el mouse el usuario puede examinar un Khatchkar desde distintos ángulos. Esta ventana es la Vista de Información Adicional.
- Modificar los Khatchkars presentes en la sala (ver Figura 1.5); en particular, el usuario puede hace los siguientes cambios:
  - Seleccionar un Khatchkar de un conjunto predefinido y agregarlo a la sala.
  - Eliminar Khatchkars presentes en la sala.
  - Mover Khatchkars a una ubicación distinta dentro de la sala.
  - Cambiar la orientación de los Khatckars en la sala.

  El usuario realiza estos cambios desde una perspectiva elevada, llamada Vista Superior.
- Guardar y cargar modificaciones hechas a la sala. El usuario puede guardar el estado actual de la sala, es decir, los Khatchkars presentes, sus ubicaciones y sus orientaciones. Para esto la aplicación permite al usuario asignar un nombre a la configuración actual. Hecho esto, el usuario puede cargar configuraciones guardadas previamente.
- Obtener ayuda (tecla "H").

Es entonces esta la aplicación que se quiere modificar para que el usuario pueda interactuar usando gestos con las manos sumido en un ambiente de realidad virtual. Para ello se utilizará

un dispositivo de captura de gestos de manos combinado con otro que permite la visualización inmersa en un ambiente de realidad virtual. Los detalles de estos dispositivos se encuentran en el siguiente capítulo. El alcance de este trabajo no considera determinar cuáles gestos son más intuitivos o permiten una mejor inmersión o interacción con el ambiente del Museo Virtual. Eso requeriría de un estudio mucho más acabado y gran cantidad de pruebas con usuarios, lo que sobrepasaría los límites de tiempo establecidos para un trabajo de título. El resultado de este proyecto es permitir interacciones mediante el sensor elegido, de forma que un eventual futuro estudio determine cuáles son los gestos más apropiados para este tipo de aplicación. Por lo mismo, no se realizarán pruebas de usabilidad, sólo de funcionalidad para evaluar el correcto funcionamiento de las nuevas interacciones implementadas. Es decir, se evaluará que las interacciones ejecuten la acción que corresponde, buscando reducir lo más posible el número de errores de software.

Figura 1.2: Menú de entrada a la aplicación

Figura 1.3: Navegación en una sala del museo (Vista Principal)

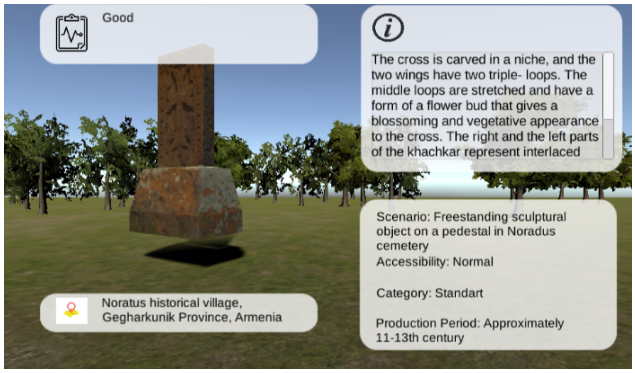

Figura 1.4: Información de piedra particular (Vista de Información Adicional)

Figura 1.5: Interfaz de modificación de una sala (Vista Superior)

Capítulo 2

Estado del Arte

2.1. Museos Virtuales

En la actualidad, muchos museos ofrecen algún tipo de exhibición mediante el uso de Realidad Aumentada y/o Realidad Virtual. Las tecnologías de Realidad Aumentada pueden usarse, por ejemplo, para entregar información adicional a los usuarios, como en el caso del Museo de la Mina de Estaño de Geevor, donde una aplicación móvil de Realidad Aumentada cuenta con un guía virtual y con texto, video y audio superpuestos para entregar más información [9, 10].

En el caso de las exhibiciones que utilizan Realidad Virtual, el tipo de exhibición, así como el tipo y nivel de interacción con dicha exhibición, dependen en gran parte del contenido de la exhibición. Podemos agrupar las exhibiciones de Realidad Virtual en las siguientes categorías:

1. Paseo virtual de trayectoria fija. En estas exhibiciones, el usuario es trasladado en forma automática a través de la exhibición, en un trayecto fijo y predefinido. El usuario sólo puede mirar a su alrededor y recibir información adicional, sin interactuar con el entorno. Un ejemplo de este tipo de exhibición es el paseo virtual del Asentamiento de La Draga, que consiste de un recorrido de cinco minutos del asentamiento y sus alrededores [11]. Los entornos presentados en la exhibición pueden incluso ser completamente imaginarios: el Palacio Ducal de Venecia cuenta con una exhibición basada en la obra Visiones del Más Allá del Bosco [12].
2. Paseo con libertad de movimiento. El usuario no está limitado por una trayectoria fija, sino que cuenta con libertad de movimiento. Al igual que en el paseo de trayectoria fija, el usuario no puede interactuar con su entorno, sólo observarlo y recibir información adicional. Un ejemplo de este tipo de exhibición se encuentra en el Museo de la Mina de Estaño de Geevor, donde el uso de Realidad Virtual permite visitar un túnel de la mina que actualmente es inaccesible [9, 10]. Una forma distinta de este tipo de exhibición se encuentra en la Anise Gallery de Londres, donde se presenta al usuario un conjunto de modelos a escala de edificios de un distrito de Londres. Esta es una exhibición multisensorial, ya que el usuario también puede percibir los olores asociados al entorno presentado [13].

Figura 2.1: Museo del Asentamiento de La Draga

Figura 2.2: Museo del Centro Comunitario de Old-Segeberg

3. Paseo interactivo. El usuario cuenta con libertad de movimiento y además puede interactuar con su entorno. En este contexto existen distintas formas de interacción. Algunas exhibiciones permiten tomar objetos y examinarlos con mayor detalle: por ejemplo, el British Museum cuenta con una exhibición que permite a los usuarios visitar asentamientos rurales de hace más de 5000 años, donde pueden explorar y examinar objetos de dicho período [14]. Otras exhibiciones incluso cuentan con juegos que permiten aprender sobre el contenido de la exhibición. Uno de estos juegos se encuentra, nuevamente, en el Asentamiento de La Draga, donde el juego consiste en eliminar los objetos que no pertenecen al periodo Neolítico [11]. En el caso del museo del Centro Comunitario de Old-Segeberg, el usuario puede interactuar con distintos objetos dentro del edificio e incluso visualizar distintas etapas de construcción del mismo [2].

2.2. Captura de Gestos

Los dispositivos que capturan gestos del usuario pueden clasificarse en activos o pasivos. Su clasificación depende de si requieren contacto físico con el dispositivo. Los dispositivos activos requieren ser tomados o usados como prendas de vestir para capturar gestos. Algunos ejemplos de dispositivos activos son los guantes de datos, el Nintendo Wiimote y el brazalete Myo.

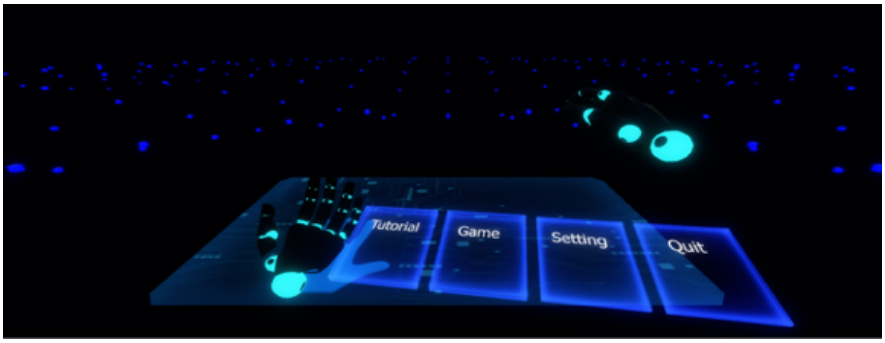

Figura 2.3: Interfaz de usuario de un juego en Realidad Virtual con interacción touchless

Los dispositivos pasivos utilizan visión por computador (y ocasionalmente otros medios, como sonido, luz o ruido electromagnético) para capturar los gestos, sin requerir contacto físico. Dos de estos dispositivos son el Kinect de Microsoft y el sensor Leap Motion de UltraLeap.

El Kinect es un dispositivo que incluye una cámara RGB, sensores de profundidad, micrófonos, e inclinación motorizada. Permite detectar y seguir puntos específicos del cuerpo obtenidos mediante mapas de profundidad y algoritmos de detección de postura, lo que a su vez permite su funcionamiento sin necesidad de calibración previa al uso. Se ha determinado que el Kinect es apropiado para capturar comandos gestuales gracias a que su alcance, precisión y errores son aceptables. De todas formas, el Kinect cuenta con limitaciones: por un lado, su rendimiento es limitado bajo ciertas condiciones de iluminación; por otro lado, tanto su campo de visión como el número de personas que puede reconocer son características que podrían mejorarse en versiones futuras.

El sensor Leap Motion solo permite el seguimiento de las manos, pero a su vez permite un mejor seguimiento de los dedos. Consiste de tres emisores LED infrarrojos y dos cámaras infrarrojas, cuenta con una precisión de menos de 1 milímetro y *frame rate* máximo de captura de alrededor de 100 frames por segundo. Tiene limitaciones similares al Kinect en cuanto a campo de visión.

Una parte importante de la interacción *touchless* involucra la detección de gestos realizados con las manos. Un uso de este tipo de interacción se encuentra, por ejemplo, en el desarrollo de juegos de Realidad Virtual [15, 16]. En este caso, las tecnologías utilizadas son el HTC Vive para el ambiente de Realidad Virtual, el sensor Leap Motion para capturar gestos de las manos y un Ultrahaptics TOUCH Development Kit ("UHDK5") para entregar retroalimentación háptica (*haptic feedback*) al usuario. Este sistema fue utilizado para interactuar con un juego de ritmo, en el que el usuario debe presionar y deslizar "notas" en el momento apropiado. Como suele ser el caso en juegos de ritmo, el usuario debe poder sentir cuándo el botón es presionado, lo que en este sistema es provisto por el UHDK5. Un sistema similar fue utilizado para un juego de Realidad Virtual en el que el usuario busca aprender distintos "hechizos", como lanzar rayos o fuego con sus manos. En este caso, la retroalimentación háptica se usa para generar las sensaciones de interacción con el ambiente y de los distintos hechizos.

La detección de gestos de las manos tiene otras aplicaciones. Por ejemplo, se ha utilizado

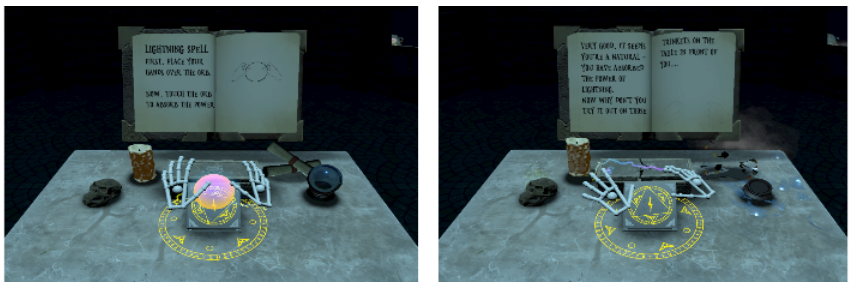

Figura 2.4: Hechizos en un juego con interacción touchless

Figura 2.5: Interacción touchless mediante avatar

un sensor Kinect en conjunto con un proyector, para estudiar posibles formas de facilitar la accesibilidad de la interacción a niños con autismo [17]. En este sistema, el usuario controla un avatar mediante sus movimientos. Este avatar se muestra en la pantalla e interactúa con los demás elementos presentes en ella.

Otra aplicación de la detección de gestos de las manos es la interacción con *public displays* [18]: displays presentes en espacios públicos, que permiten obtener información adicional sobre el lugar o información de interés general como noticias, el tiempo, etc. En un sistema como este, el principal desafío consiste en cómo lograr que el uso sea intuitivo para usuarios con distintos niveles de conocimiento tecnológico. Esta dificultad también se ha visto, por ejemplo, en estudios sobre interacción touchless en automóviles [19]: al estudiar posibles gestos para realizar distintas tareas, como cambiar la música o realizar un giro con el vehículo, los usuarios muestran una alta variedad de gestos que consideran apropiados e intuitivos.

Otros sistemas de detección de gestos requieren el uso de marcadores en las manos para lograr el seguimiento de distintos puntos en ellas. Estos sistemas proveen mayor precisión y confiabilidad que sensores como el Kinect o el Leap Motion. Esto es necesario, por ejemplo, al estudiar el rendimiento de la mano dominante en sistemas interactivos touchless comparado con la mano no dominante [20]. En un caso como este, dado que se esperan efectos pequeños a medianos, se prefiere un sistema de motion capture como el VICON, que utiliza marcadores.

A veces puede ser necesario detectar gestos realizados con otras partes del cuerpo. En el caso de procedimientos médicos, existen casos en que el médico debe interactuar con software de imágenes mientras opera sobre el paciente. Dificultades como la restricción de movimiento

Figura 2.6: Interacción touchless mediante VICON

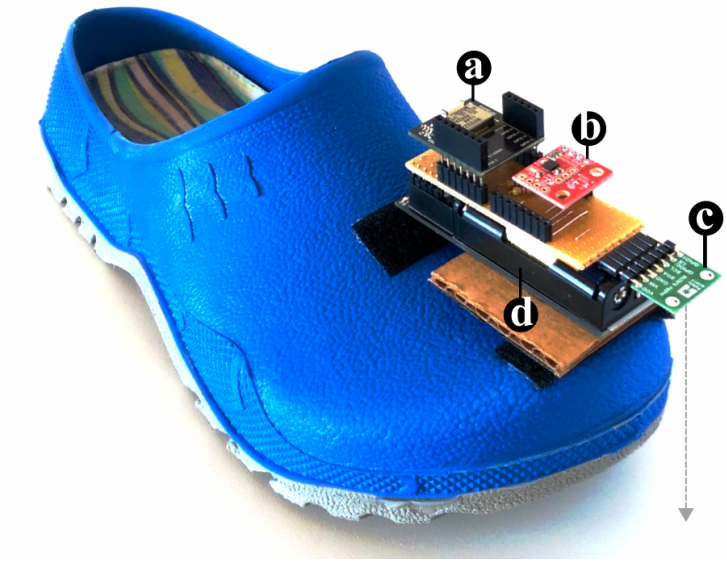

Figura 2.7: Sensor para pie

de los brazos y el problema de detectar gestos en manos que están sujetando instrumentos, hacen que el uso de tecnologías como las mencionadas anteriormente no sea el más apropiado. Para esto se ha estudiado la posibilidad de realizar gestos con los pies, mediante un sensor que se coloca en el zapato del médico [21].

2.3. Ambiente de Desarrollo 3D

El Museo Virtual está implementado en el ambiente de desarrollo 3D *Unity*. Unity es un motor gráfico que permite el desarrollo de aplicaciones en 2D, 3D y Realidad Virtual. Para esto cuenta con herramientas como simulación física, edición de animaciones, etc. En general, todo proyecto de Unity cuenta con al menos tres elementos:

1. Escenas. Una escena es un espacio que contiene elementos de la aplicación. En el caso del Museo Virtual, cada una de las salas, así como los distintos menús, corresponden internamente a una escena distinta.
2. GameObjects. Un GameObject es un elemento de una escena. Por sí mismos no hacen mucho, pero actúan como contenedores de *componentes*, que le otorgan las características y funcionalidades al GameObject. Por ejemplo, un GameObject que representa a un cubo tendrá entre sus componentes, una malla de polígonos que permitirá dibujarlo y un detector de colisiones que le permitirá interactuar físicamente con otros GameObjects. Además, Unity permite la programación de scripts en C# para modificar la funcionalidad de los GameObjects. Estos scripts también cuentan como componentes. Los *prefabs* son una forma específica de GameObject que consisten en conjuntos reutilizables de GameObjects que representan un objeto. Por ejemplo, si cubos y cilindros son GameObjects, es posible armar un *prefab* llamado Automóvil que consiste en un cubo (el cuerpo del automóvil) y cuatro cilindros en posiciones específicas con respecto al cubo (las ruedas del auto). Este *prefab* luego puede ser reutilizado sin tener que volver a armar el conjunto de cubos y cilindros.
3. Cámara. La cámara es un GameObject que permite visualizar una escena.

Cuando uno crea un nuevo proyecto en Unity, lo primero que debe elegir es el nombre del proyecto, su ubicación y el tipo de proyecto: si es 2D, 3D, etc., como se ve en la figura 2.8. Una vez definido esto, el desarrollador ingresa al Editor, que contiene todo lo necesario para empezar a desarrollar la aplicación. Algunas de las partes importantes del Editor son las siguientes:

1. El panel de jerarquía (ver figura 2.9). Este panel muestra la estructura completa del proyecto, con sus distintas escenas y GameObjects. Permite crear nuevas escenas y GameObjects y modificar la estructura jerárquica de éstos.
2. El explorador del proyecto (ver figura 2.10). Aquí se muestra el contenido del directorio del proyecto. Dicho directorio contiene todos los *Assets* del proyecto, tales como texturas, modelos 3D, scripts, etc.

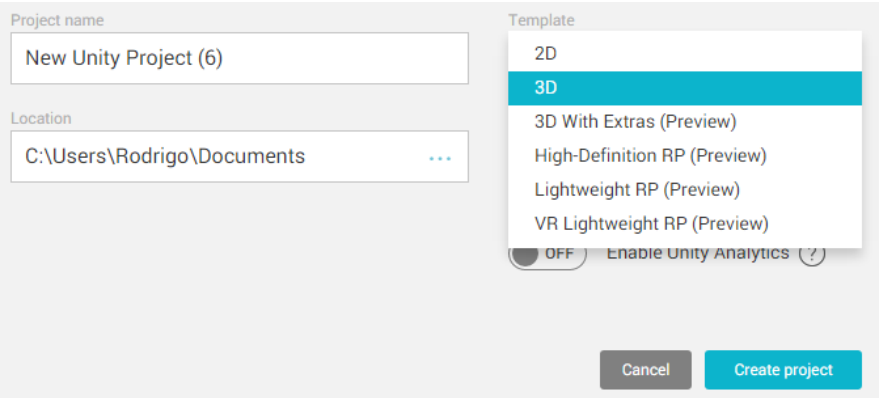

Figura 2.8: Diálogo de creación de un nuevo proyecto en Unity

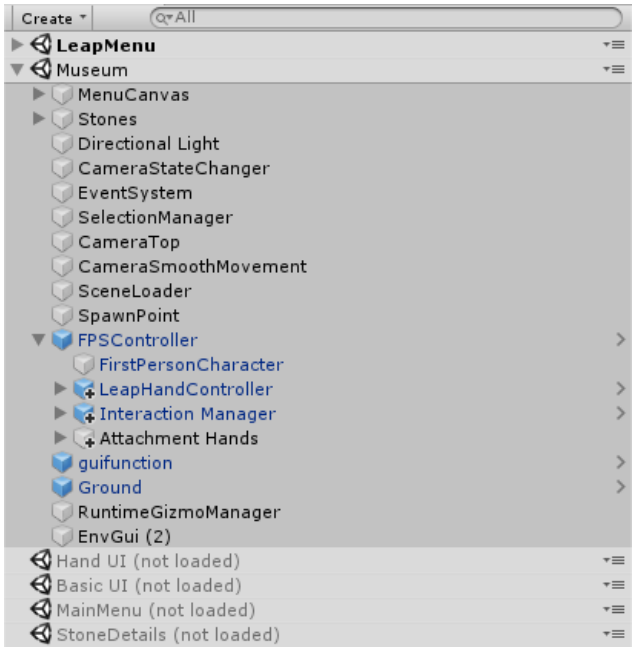

Figura 2.9: Panel de jerarquía de un proyecto

3. El Inspector (ver figura 2.11). Este panel permite modificar los componentes de cada GameObject y sus parámetros. El contenido del Inspector dependerá del GameObject seleccionado.

4. El visor de escena (ver figura 2.12). Este panel permite visualizar y modificar la ubicación física de los distintos elementos de la escena actual.

Además, Unity permite modificar opciones globales del proyecto, tales como parámetros de la física del proyecto (gravedad, colisiones, etc.), gráficas, inputs, entre otros. También permite elegir el tipo de plataforma a la que está destinado el proyecto: PC, iOS, Android, entre otros.

Existen otros ambientes de desarrollo en el mercado: Unreal Engine es un sistema muy usado en la industria de los videojuegos AAA, mientras que Godot es popular en el mundo de desarrolladores independientes. Unreal Engine es un ambiente de desarrollo de código abierto

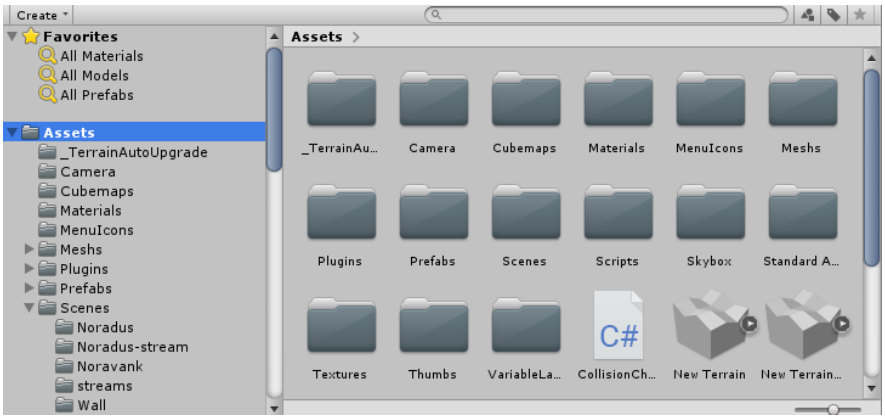

Figura 2.10: Explorador del directorio de un proyecto

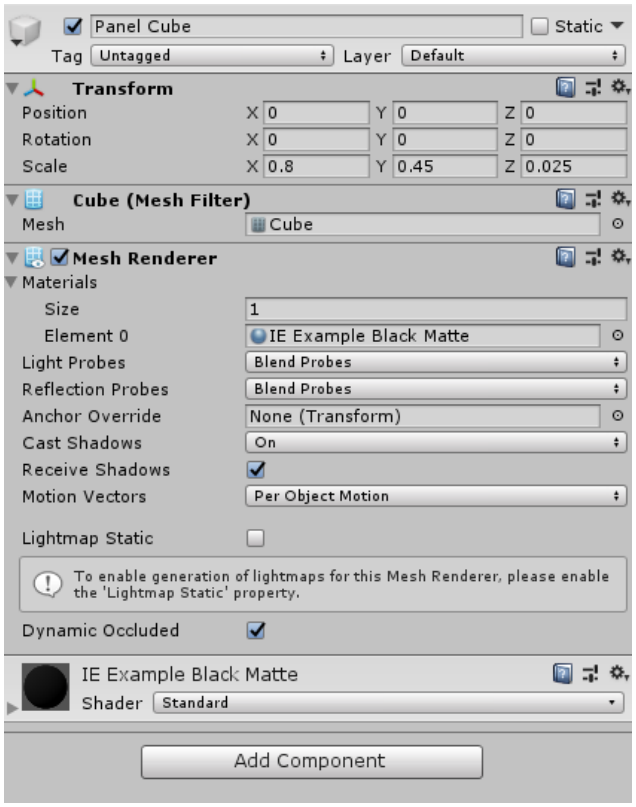

Figura 2.11: Inspector de un GameObject

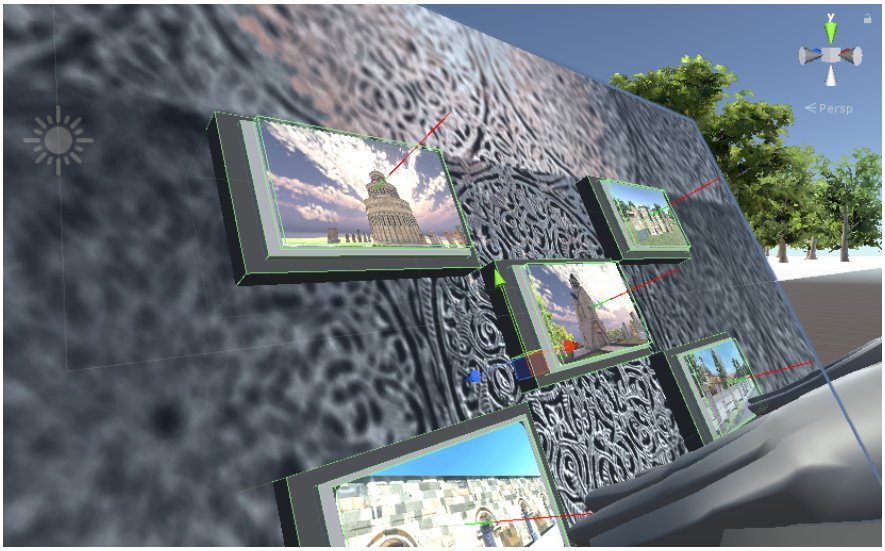

Figura 2.12: Visualización de una escena

creado por Epic Games y es similar a Unity en cuanto al uso de escenas, GameObjects, componentes, etc, para armar una aplicación. A diferencia de Unity, Unreal Engine utiliza C++ para sus scripts. Por otro lado, Godot utiliza un sistema más simple, en el que todo elemento es un Nodo y cada Nodo puede tener un script asociado. Cada Nodo, a su vez, puede estar conformado por combinaciones de otros Nodos. Godot utiliza un lenguaje propio, llamado GDScript, para la programación de scripts, aunque también permite el uso de scripts en C#. Dado que el sensor Leap Motion cuenta con soporte para Unity por parte del fabricante, se decidió continuar trabajando con el Museo Virtual en Unity en lugar de crear una versión para otro motor gráfico.

Capítulo 3

Diseño de la Solución

Para diseñar e implementar las nuevas interacciones en el Museo, se siguieron los siguientes pasos:

1. Incorporación del sensor Leap Motion al proyecto en Unity.
2. Identificación y clasificación de interacciones.
3. Diseño de adaptación para cada interacción.
4. Implementación de cada adaptación.

A continuación se explican los detalles de los pasos 1, 2 y 3. El paso 4 se explica en el capítulo 4.

3.1. Incorporación del sensor Leap Motion al proyecto en Unity

Antes de empezar a diseñar cualquier interacción, es fundamental preparar el proyecto de Unity para que permita utilizar el sensor Leap Motion.

El primer paso consistió en la instalación de los módulos *Core* e *Interaction Engine* provistos por UltraLeap. Core provee la funcionalidad básica del dispositivo: leer datos del sensor, dibujar manos en la pantalla a partir de los datos obtenidos, adjuntar objetos a puntos específicos de las manos, entre otros. Además, incluye la API necesaria para trabajar con el sensor en los scripts de Unity. Interaction Engine contiene los elementos necesarios para permitir que las manos generadas por el sensor puedan interactuar en forma física (es decir, tocar, agarrar, empujar, etc.) con otros objetos. Los otros módulos ofrecidos por UltraLeap no son relevantes para el proyecto: *Graphic Renderer* se encarga de optimizar el proceso de renderizado, mientras que *Hands* permite al desarrollador convertir un modelo 3D de manos en un modelo compatible con el sensor Leap Motion.

A continuación, se incorporaron los siguientes objetos (incluidos en los módulos recién instalados) en cada escena de Unity que requiere el uso del sensor:

- *LeapHandController* se encarga de dibujar las manos en la pantalla a partir de los datos que obtiene del sensor. Este objeto se ubicó en la jerarquía de objetos de la sala siguiendo la recomendación de UltraLeap: LeapHandController debe ser hijo de la cámara activa, de forma que si la ubicación de la cámara cambia, las manos mantienen su posición con respecto a ella.
- *InteractionManager* se encarga de administrar el estado de todos los objetos "interactuables" mediante el sensor. Todos los objetos que el usuario puede deslizar, agarrar, presionar, etc., estarán conectados al InteractionManager.
- *AttachmentHands* permite al desarrollador "adjuntar" objetos a puntos específicos de las manos, como las palmas, las puntas de los dedos, etc. AttachmentHands no es necesario para el funcionamiento del sensor en Unity, pero permite tener opciones adicionales para el diseño y la implementación de interacciones.

Una vez comprobado el funcionamiento del sensor, se procedió a identificar las interacciones presentes en el Museo.

3.2. Identificación y Clasificación de Interacciones

Para identificar todas las interacciones posibles en el Museo, se realizaron dos actividades:

- Utilización de la aplicación, buscando cada una de las interacciones posibles.
- Revisión de la documentación, mensajes de ayuda, etc.

Realizar ambas actividades hizo posible crear una lista con todas las interacciones que serían reemplazadas por el uso del sensor. Además, para facilitar el diseño de cada una de las adaptaciones, se buscó categorizar las interacciones según su tipo. Se encontraron tres categorías de interacción.

La primera categoría consiste en interacciones "acotadas" o "discretas": aquellas interacciones que comienzan y terminan al presionar una tecla, hacer click con el mouse, etc. Interacciones tales como:

- Presionar la tecla "M" para volver al Menú Principal.
- Hacer click en el botón "+" para agregar un Khatchkar.
- Presionar la tecla "E" mientras se mira hacia una Khatchkar para ver más información.

caen dentro de esta categoría.

En segundo lugar están las interacciones "continuas": aquellas interacciones que requieren una acción continuada por parte del usuario. En esta categoría se encuentran interacciones tales como:

- Mantener presionada una tecla para moverse dentro de la Sala.
- Hacer click y arrastrar para cambiar la ubicación de un Khatchkar.
- Hacer click y arrastrar para examinar un Khatchkar desde distintos ángulos.

Finalmente, se tienen interacciones que no necesariamente entran en alguna de las categorías anteriores. Por ejemplo, "ingresar el nombre para guardar una configuración" está en esta categoría.

Una vez identificadas las interacciones, se procedió a diseñar una posible adaptación para cada una de ellas, de forma que el usuario pueda obtener los mismos resultados utilizando el sensor. A continuación se describen las soluciones ideadas.

3.3. Diseño de adaptación para cada interacción

3.3.1. Interacciones discretas

Menú de Comandos

Muchas de las interacciones discretas identificadas consisten en comandos que el usuario puede invocar en cualquier momento, independiente de su ubicación, qué está haciendo, etc. Ejemplos de este tipo de interacciones incluyen:

- Presionar la tecla "M" para volver al menú principal.
- Presionar la tecla "T" para ir a la Vista Superior
- Hacer click sobre botón "Save" para abrir el diálogo para guardar configuración

Al leer la documentación del sensor [22], se observó que uno de los ejemplos provistos consiste en una interfaz de usuario portátil, a la que el usuario puede acceder en cualquier momento orientando la palma de su mano izquierda hacia su cara. Como es necesario que el usuario tenga acceso a las acciones globales recién descritas usando únicamente el sensor Leap Motion, se decidió implementar un "Menú de Comandos", que consiste en una versión modificada de la interfaz de usuario portátil. Este menú aparece cuando el usuario orienta la palma de su mano izquierda hacia la cámara (es decir, hacia el usuario) y contiene un botón para cada uno de los posibles comandos. Los botones disponibles en un momento dado dependerán del contexto: por ejemplo, el botón para ir a la Vista Superior no se encontrará disponible si el usuario ya está en la Vista Superior. La idea es que el usuario sólo tenga acceso a los comandos que "tengan sentido" según el contexto en que se encuentre. La lista de botones presentes en

el Menú de Comandos irá cambiando a medida que se vaya decidiendo cuáles interacciones corresponde incluir en él. Inicialmente, el Menú de Comandos tendrá dos botones: un botón "Menú" para volver al Menú Principal y un botón "Salir" para cerrar la aplicación.

Inicialmente se consideró la posibilidad de implementar gestos para realizar las acciones asociadas al Menú de Comandos. Sin embargo, dado que cada una de estas acciones requeriría diseñar un gesto, que posiblemente agregaría complejidad adicional para el usuario, se optó por utilizar el Menú de Comandos. Esto además permite tener las interacciones discretas en un solo lugar. Por estas razones, se decidió que varias de las interacciones descritas a continuación consistieran en presionar un botón del Menú de Comandos.

Ir a Vista Superior

En la versión original del Museo, cuando el usuario presiona la tecla "T" es enviado a la Vista Superior de la Sala: en lugar de ver la Sala como si estuviera dentro de ella, el usuario para a ver la Sala desde arriba, donde tiene la posibilidad de agregar, modificar y eliminar los Khatchkars presentes en la Sala.

Dado que esta acción se realiza mediante presionar una tecla, la solución propuesta consiste en asignar esta funcionalidad a un botón del Menú de Comandos en lugar de, por ejemplo, diseñar, implementar y detectar algún gesto apropiado.

Volver a la Sala

En la versión original del Museo, el usuario puede presionar la tecla "P" para volver a la Vista Principal, cuando se encuentra en la Vista Superior o en la Vista de Información Adicional.

La solución propuesta consiste en asignar esta funcionalidad a uno de los botones del Menú de Comandos. Al igual que en la interacción anterior, se decidió no implementar un gesto para volver a la Sala.

Activar Menú para Agregar Khatchkars

En la versión original del Museo, cuando el usuario se encuentra en la Vista Superior tiene la posibilidad de agregar Khatchkars a la Sala. Para hacer esto, el usuario puede hacer click sobre un botón etiquetado "+", lo que despliega un menú con los distintos Khatchkars que es posible agregar.

La adaptación de este menú se verá aparte; la solución propuesta para activarlo consiste en asignar esta funcionalidad a uno de los botones del Menú de Comandos.

Obtener Ayuda

En la versión original del Museo, cuando el usuario presiona la tecla "H" se despliega un mensaje de ayuda (ver figura 3.1). Este mensaje contiene instrucciones sobre cómo usar el Museo, incluyendo cuáles teclas presionar y cómo realizar las acciones que en principio son menos intuitivas (por ejemplo, cómo mover los Khatchkars en la Vista Superior). Si el usuario vuelve a presionar la tecla "H", el mensaje desaparece.

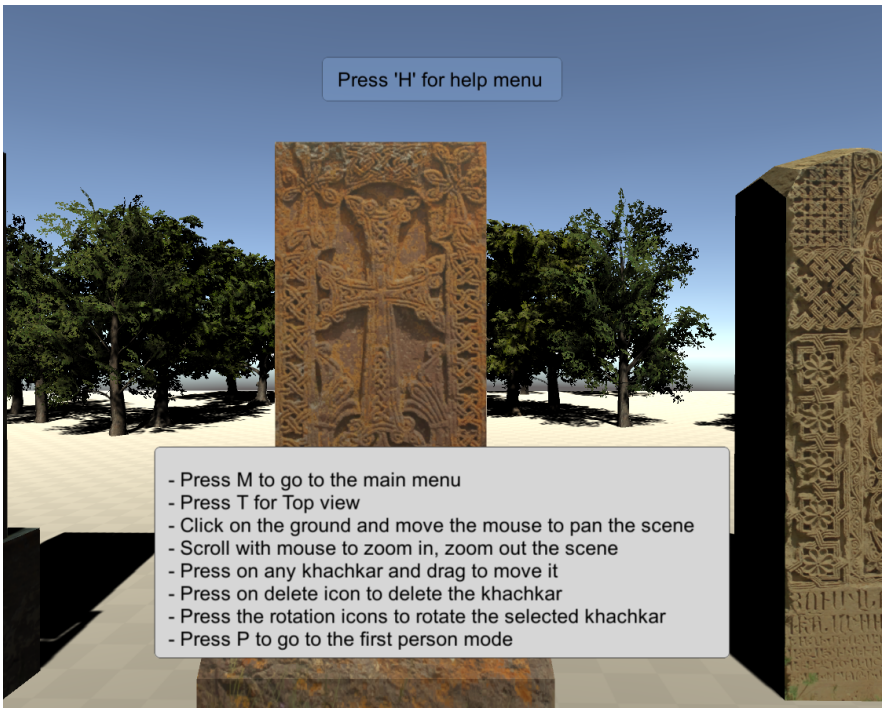

Figura 3.1: Mensaje de ayuda en la versión original del Museo

La solución propuesta en este caso consiste en asignar la funcionalidad descrita a un botón del Menú de Comandos. Implementar esta solución requerirá modificar el contenido del mensaje de ayuda, de forma que muestre las instrucciones para las nuevas interacciones. Esta solución contempla una mejora con respecto a la versión original del Museo: el mensaje original contiene las instrucciones para la Vista Principal y la Vista Superior. Por lo tanto, se propone implementar funcionalidad que permita que el mensaje de ayuda para ambas vistas sea distinto, conteniendo solamente las instrucciones apropiadas para la vista en la que se encuentre el usuario.

Activar diálogo para guardar configuración

En la versión original del Museo, cuando el usuario se encuentra en la Vista Superior tiene la posibilidad de guardar la configuración de la Sala. Al hacer click sobre un botón etiquetado "Save", se despliega un diálogo que permite al usuario asignarle un nombre a la configuración actual de la Sala (es decir, los Khatchkars presentes, sus ubicaciones y sus orientaciones) y guardarla (ver figura 3.2).

Figura 3.2: Diálogo para guardar configuraciones en la versión original del Museo

Al igual que en el caso del menú para agregar Khatchkars, la adaptación del diálogo se verá aparte. La solución propuesta para activarlo consiste en asignar esta funcionalidad a uno de los botones del Menú de Comandos.

Activar diálogo para cargar configuración

En la versión original del Museo, cuando el usuario se encuentra en la Vista Superior tiene la posibilidad de cargar una configuración guardada anteriormente. Al hacer click sobre un botón etiquetado “Load”, se despliega un diálogo que permite al usuario ingresar el nombre de una configuración de la Sala guardada con anterioridad y cargarla (ver figura 3.3).

Nuevamente, la adaptación de este diálogo se verá más adelante. La solución propuesta para activarlo consiste en asignar esta funcionalidad a uno de los botones del Menú de Comandos.

Menú Principal

En la versión original del Museo, al iniciar la aplicación el usuario puede interactuar con un menú como el que se ve en la figura 1.2. El menú consiste en cinco botones, donde cada botón representa una de las Salas del Museo. Al hacer click sobre uno de estos botones, se carga y se muestra la Sala correspondiente al botón.

Dado que esta interacción es equivalente a elegir un botón y presionarlo, la solución propuesta consiste en implementar una versión de este menú que sea compatible con el sensor

Figura 3.3: Diálogo para cargar configuraciones en la versión original del Museo

Leap Motion. Esto implica la construcción de un menú nuevo, hecho a partir de componentes sujetos a la simulación física de Unity, por el siguiente motivo: las manos generadas por el sensor Leap Motion están hechas para interactuar físicamente con objetos, mientras que el menú original está implementado mediante componentes de interfaz de usuario de Unity, no sujetos a la simulación física. Esto hace que el menú original sea incompatible con el sensor.

El menú nuevo consistirá de cinco botones. Al presionar uno de ellos con una de sus manos, el usuario podrá acceder a la Sala correspondiente. El menú será, a la vista, muy similar al menú original, utilizando las mismas imágenes y organizado de la misma forma. Esto permitirá que los usuarios que hayan utilizado la versión original del Museo puedan reconocer con mayor facilidad la función de los elementos presentes.

Agregar Khatchkar

Como se explicó anteriormente, al presionar el botón "+" aparece un menú con los Khatchkars disponibles para agregar a la Sala. Al hacer click sobre uno de ellos, el Khatchkar seleccionado aparece en la Sala, en el lugar que corresponde al centro de la pantalla.

Agregar un Khatchkar requiere la presencia de un menú, lo que tiene el mismo problema del Menú Principal. Además, para poder agregar un Khatchkar en forma satisfactoria, se necesita que la Sala siga siendo visible mientras se realiza esta acción. Por lo tanto, la solución propuesta es la siguiente: implementar un panel adicional para el Menú de Comandos, que contenga botones para agregar los distintos Khatchkars. Este panel sólo se despliega cuando el usuario presiona el botón del Menú de Comandos descrito anteriormente. Para implementar este panel se debe considerar la posibilidad de que el número de Khatchkars disponibles sea

Figura 3.4: Menú para agregar Khatchkars

muy grande para mostrarlos todos a la vez.

Eliminar Khatchkar

En la versión original del Museo, cuando el usuario se encuentra en la Vista Superior y selecciona un Khatchkar, tiene la opción de eliminarlo de la Sala haciendo click en el botón "X" que aparece en la pantalla.

Una posible solución consiste en utilizar algún gesto de la mano derecha (por ejemplo, "apretar" el Khatchkar con el pulgar y el índice) para eliminar el Khatchkar. Otra posible solución consiste en delimitar una zona de la pantalla, tal que si el usuario suelta un Khatchkar en esta zona, se elimina el Khatchkar. Se explorará la factibilidad de ambas soluciones y se elegirá una de ellas.

Orientar Khatchkars

En la versión original del Museo, el usuario puede modificar la orientación de los Khatchkars cuando se encuentra en la Vista Superior. Al hacer click sobre uno de los Khatchkars, aparecen botones que permiten rotar el Khatchkar con respecto a la vertical (ver figura 3.5).

Dado que la solución propuesta para modificar la posición de los Khatchkars involucra utilizar la posición de la mano del usuario en la pantalla (ver más adelante), la solución propuesta consiste en utilizar la rotación de la mano del usuario de forma similar para modificar la orientación del Khatchkar.

Figura 3.5: Botones para rotar y eliminar Khatchkar

3.3.2. Interacciones continuas

Mover Khatchkars

En la versión original del Museo, el usuario puede modificar la posición de los Khatchkars cuando se encuentra en la Vista Superior. El usuario puede lograr esto haciendo click sobre uno de los Khatchkars visibles y arrastrándolo hacia la nueva posición. Además, cuando el puntero del mouse se encuentra sobre un Khatchkar, éste último se ilumina (como se ve en la figura 3.6) para indicar que ese será el Khatchkar que se moverá.

En otras palabras, esta interacción puede desglosarse de la siguiente forma:

- Elegir: se coloca el puntero del mouse sobre el Khatchkar a mover.
- Desplazar: se hace click sobre el Khatchkar elegido y se mueve el mouse manteniendo el click.
- Soltar: se suelta el click cuando el Khatchkar está en la posición deseada.

Estos tres pasos pueden convertirse a gestos en forma directa, de la siguiente forma:

- Elegir: se coloca la mano sobre el Khatchkar a mover. El Khatchkar se ilumina para indicar que la mano está sobre él.
- Desplazar: se hace el gesto de agarrar con la mano sobre el Khatchkar elegido y se mueve la mano manteniendo el agarre.

Figura 3.6: Khatchkar iluminado

- Soltar: se suelta el agarre cuando el Khatchkar está en la posición deseada.

Debido a lo directo de esta conversión, se propone que esta sea la forma de mover Khatchkars en la Vista Superior. Además, como se menciona en la sección de rotación de Khatchkars, este formato de interacción permite incorporar la rotación de la mano a la fase de desplazamiento, permitiendo al usuario determinar la orientación del Khatchkar.

Movimiento del Usuario

En la versión original del museo, cuando el usuario se encuentra en la Vista Principal puede usar el teclado para desplazarse: la tecla “W” para avanzar, la tecla “S” para retroceder y las teclas “A” y “D” para desplazarse en forma lateral.

Una posible solución consiste en agregar botones al Menú de Comandos que cumplan el mismo rol que las teclas correspondientes. Esta posible solución significaría agrandar aún más el Menú de Comandos y podría incluso ser necesario agregar un tercer panel. Dadas estas complicaciones, se opta por otra solución: detectar si el usuario está apuntando con un dedo en una dirección y desplazarlo en esa misma dirección.

Desplazar Vista

En la versión original del Museo, cuando el usuario se encuentra en la Vista Superior puede desplazar la posición de la cámara, de forma que se vea otra parte de la sala en la pantalla. Para esto cuenta con dos opciones:

- Hacer click derecho y arrastrar para desplazarse en el plano horizontal.
- Scrolling con la rueda del mouse para cambiar la altura de la cámara.

El sensor Leap Motion permite determinar la posición 3D de las manos. Adicionalmente, la posición de la mano derecha se utilizará para el desplazamiento de Khatchkars. Por lo tanto, la solución propuesta consiste en convertir las opciones mencionadas en una sola interacción: el usuario podrá "agarrar y arrastar" con la mano izquierda para cambiar la posición de la cámara en cualquier dirección. Al igual que en el caso del movimiento del usuario, se optó por esta solución en vez de un panel que permitiera controlar la posición.

Scrolling de Texto

En la versión original del Museo, la Vista de Información Adicional incluye un campo de texto con información sobre el Khatchkar elegido. El campo de texto incluye un scrollbar vertical ya que a veces hay más texto del que puede visualizarse. Usando la rueda del mouse, el usuario puede avanzar y retroceder la posición en el campo de texto.

Para diseñar una posible solución, fue necesario examinar cómo funciona un scrollbar de Unity. También fue necesario estudiar la documentación del sensor para ver qué posibilidades ofrece. Se determinó que los scrollbars en Unity utilizan un valor entre 0 y 1 para representar su posición actual. Una primera propuesta consistió en controlar la posición del scrollbar mediante botones en el Menú de Comandos. Estos botones aparecerían solamente en la Vista de Información Adicional, y al presionarlos modificarían el valor del scrollbar. Sin embargo, al leer la documentación del sensor, se encontró un componente posiblemente más apropiado para esta tarea: el Interaction Slider. El Interaction Slider es un deslizador, como aquellos presentes en interfaces de usuario estándar, con el cual el usuario puede interactuar físicamente. Este Interaction Slider permite "mover" un valor entre dos límites arbitrarios, por lo que se decidió utilizar uno de estos componentes para controlar la posición del scrollbar. Al igual que en la propuesta inicial, este slider será parte del Menú de Comandos y sólo estará presente cuando el usuario se encuentre en la Vista de Información Adicional.

Examinar Khatchkars

En la versión original del Museo, cuando el usuario se encuentra en la Vista de Información Adicional puede hacer click sobre el Khatchkar presente y arrastrar hacia los lados, lo que permite examinar el Khatchkar desde distintos ángulos.

La solución propuesta, dado que una de las capacidades importantes del sensor es la posibilidad de manipular objetos directamente con las manos, consiste en permitir la manipulación directa del Khatchkar mediante las manos del usuario. El usuario podrá agarrar y girar el Khatchkar para examinarlo desde distintos ángulos.

Figura 3.7: Vista de Información Adicional

Orientación del usuario en la Sala

En la versión original del Museo, cuando el usuario se encuentra en la Vista Principal puede mover el mouse para cambiar hacia dónde está mirando: por ejemplo, al mover el mouse hacia la derecha, el usuario mirará a la derecha; al mover el mouse hacia adelante, mirará hacia arriba.

Uno de los objetivos de esta seria de trabajos es estudiar cómo mejorar la inmersión del usuario en el Museo. Una de las opciones consiste en utilizar tecnología de Realidad Virtual. Una de las capacidades importantes de tecnologías de este tipo consiste en detectar cambios en la orientación de la cabeza del usuario, lo que permite, por ejemplo, utilizar movimientos de la cabeza para controlar la orientación de la cámara en aplicaciones que utilizan perspectiva de primera persona. Por lo tanto, se propone utilizar tecnología de Realidad Virtual para controlar la orientación del usuario en la Sala. Al igual que los gestos, en este proyecto no se evaluarán los niveles de intuición o inmersión generados por el uso de esta tecnología, sino que quedarán disponibles para su uso en trabajos futuros.

3.3.3. Otras interacciones

Guardar configuración

Como se explicó anteriormente, el usuario puede activar un diálogo que permite guardar la configuración de la Sala. Este diálogo permite al usuario ingresar un nombre para la configuración, de forma de poder identificarla cuando quiera cargar la configuración.

La solución propuesta consiste en implementar un panel físico que permita realizar esta misma acción. La implementación de esta solución tiene dos partes:

- Como la idea de este proyecto es que el usuario pueda interactuar con el museo utilizando únicamente el sensor Leap Motion (y adicionalmente, un casco de Realidad Virtual), se hace necesario buscar una solución que permita al usuario ingresar el nombre de la configuración sin tener que usar el teclado. Además, el uso del casco de Realidad Virtual dificulta e incluso impide este tipo de interacción. Por lo tanto, se propone crear un panel físico con todos los elementos necesarios para la interacción: un teclado físico (con botones que representan las teclas) para ingresar un nombre, un campo de texto donde ingresar el nombre y botones para guardar la configuración y para cerrar el panel.
- Conectar el panel a la funcionalidad para guardar existente o implementar nueva funcionalidad para guardar configuraciones y conectar el panel a ésta. Esto último se debe a limitaciones presentes en la funcionalidad original: por ejemplo, no se almacena la Sala a la que corresponde la configuración.

Cargar configuración

Como se explicó anteriormente, el usuario puede activar un diálogo que permite cargar una configuración guardada previamente. En este diálogo, el usuario puede ingresar el nombre de la configuración que desea cargar.

La solución propuesta consiste en implementar un panel físico que permita realizar esta misma acción. La implementación de esta solución tiene dos partes:

- Al igual que en el caso anterior, la solución debe prescindir del uso del teclado. Sin embargo, se notó que incluso en su versión original el diálogo para cargar configuraciones no es el más fácil de usar, ya que requiere que el usuario sepa el nombre exacto de la configuración que desea cargar. Dadas estas consideraciones, se decidió utilizar un proceso de selección distinto al original: en lugar de ingresar el nombre de la configuración, el usuario verá una serie de botones con los nombres de las configuraciones disponibles. Al presionar uno de ellos, se cargará la configuración elegida. Al igual que en el caso del menú para agregar Khatchkars, se debe tener en cuenta el caso en que el número de configuraciones guardadas sea muy grande.
- Conectar el panel a la funcionalidad para cargar existente o implementar nueva funcionalidad para cargar configuraciones y conectar el panel a ésta, debido a limitaciones presentes en la funcionalidad original.

Capítulo 4

Solución

La versión original del Museo cuenta con una serie de GameObjects presentes en cada escena, cuyo propósito es permitir la funcionalidad implementada. La implementación de la solución requiere modificar algunos de estos objetos o incluso reemplazarlos por versiones completamente distintas. Los objetos relevantes presentes en cada Sala se describen a continuación.

MenuCanvas. Contiene elementos de interfaz de usuario, como botones, menús y mensajes. Estos objetos están en *screen space*: no existen en el espacio tridimensional de las escenas, sino en el espacio bidimensional de la pantalla.

CameraStateChanger. Contiene la funcionalidad para cambiar entre la Vista Principal y la Vista Superior.

SelectionManager. Contiene la funcionalidad para mover Khatchkars y mover la cámara en la Vista Superior.

CameraTop. La cámara asociada a la Vista superior.

CameraSmoothMovement. Contiene la funcionalidad para elegir un Khatchkar e ir a su Vista de Información Adicional.

FPSController. Contiene la funcionalidad que permite controlar la posición del usuario dentro de la Sala y la cámara asociada a la Vista Principal.

Guifunction. Contiene la funcionalidad que permite guardar y cargar configuraciones, incluyendo funcionalidad para instanciar Khatchkars.

Además, el Museo incluye el módulo *StaticValues*, cuyo propósito es almacenar información en forma independiente de la escena activa.

Para el funcionamiento del sensor Leap Motion en el Museo, se agregan los siguientes objetos a cada Sala:

InteractionManager. Se encarga de administrar el estado de todos los objetos "interactuables" mediante el sensor. Todos los objetos presionables, agarrables, deslizables, etc. estarán

conectados al InteractionManager de la Sala.

LeapHandController. Se encarga de (1) obtener la información de las manos desde el sensor y (2) usar esa información para dibujar las manos en la pantalla. El LeapHandController siempre será hijo de la cámara activa, de forma que si la cámara se mueve, las manos se dibujan en frente de la cámara.

Las interacciones implementadas se clasifican en tres categorías:

- Interacción mediante *botones*. Esta categoría incluye aquellas interacciones que requieren presionar uno o más botones. En este caso, "botón" es un objeto sujeto a la simulación física de Unity, que el usuario puede presionar con sus manos.
- Interacción mediante *manipulación de objetos*. Esta categoría incluye aquellas interacciones en que la posición, orientación, etc. de un objeto es manipulada directa o indirectamente mediante las manos del usuario.
- Otras interacciones. Esta categoría incluye interacciones que no calzan con las categorías anteriores.

Muchas de las interacciones descritas a continuación fueron diseñadas e implementadas considerando usuarios diestros. Es decir, dichas interacciones se basan principalmente en el uso de la mano derecha. Si bien no se consideró durante el diseño y la implementación la posibilidad de permitir al usuario configurar la mano predominante, esta opción sería fácil de añadir, ya que los métodos responsables de capturar los gestos realizados por la mano incluyen consultas sobre la lateralidad de la mano. La consulta es provista por la API de Leap Motion y permitiría agregar la lateralidad de la mano como parámetro para los métodos de captura de gestos.

4.1. Interacción mediante Botones

4.1.1. Menú Principal

El nuevo Menú Principal contiene los mismos elementos del Menú Principal original: cinco botones, organizados de la misma manera, con las mismas imágenes para cada botón. Al presionar uno de los botones, el usuario entra a una de las salas. La diferencia es que los botones nuevos son objetos físicos que utilizan el script *InteractionButton* de Leap Motion, que hace que actúen como botones que el usuario puede presionar con sus manos.

Además, mediante el uso de otro script de Leap Motion llamado *SimpleInteractionGlow*, los botones cambian de color según el estado de la interacción: el botón se ilumina cuando la mano está cerca y pasa a estar verde al ser presionado. Esto permite entregar retroalimentación al usuario sobre lo que está pasando al interactuar con el botón. Por lo mismo, se decidió utilizar este script cada vez que se implementara un botón.

Figura 4.1: Nuevo Menú Principal

Comando	Descripción	Principal?	Superior?	Información Adicional?
Menu	Ir al Menú Principal	✓	✓	✓
Salir	Salir de la aplicación	✓	✓	✓
Top	Ir a la Vista Superior	✓		
Main	Ir a la Vista Principal		✓	✓
Agregar	Desplegar panel para agregar piedras		✓	
Ayuda	Desplegar mensaje de ayuda	✓	✓	✓
Guardar	Guardar configuración		✓	
Cargar	Cargar configuración		✓	
Scroller*	Permite hacer scrolling de texto			✓

Tabla 4.1: Disponibilidad de comandos en las distintas Vistas

4.1.2. Menú de Comandos

Para implementar el Menú de Comandos, se optó por utilizar una versión modificada del menú provisto por Leap Motion. Este menú se despliega cuando el usuario orienta la palma de su mano izquierda hacia su cara y se mantiene al lado de su mano izquierda mientras el usuario mantenga dicha orientación. Los comandos disponibles en el Menú de Comandos aparecen en la tabla 4.1.

Todos estos comandos (excepto el Scroller) tienen un botón asignado. No todos los botones están disponibles en todas las vistas; por ejemplo, no tiene mucho sentido tener el botón Top disponible en la Vista Superior. La disponibilidad de cada botón puede verse en la tabla 4.1. Al igual que en en Menú Principal, los botones del Menú de Comandos cambian de color según su estado de interacción.

Debido a la estructura del proyecto en Unity, se requiere implementar el Menú de Comandos de tal forma que sea posible incorporarlo a cada una de las escenas sin modificar su

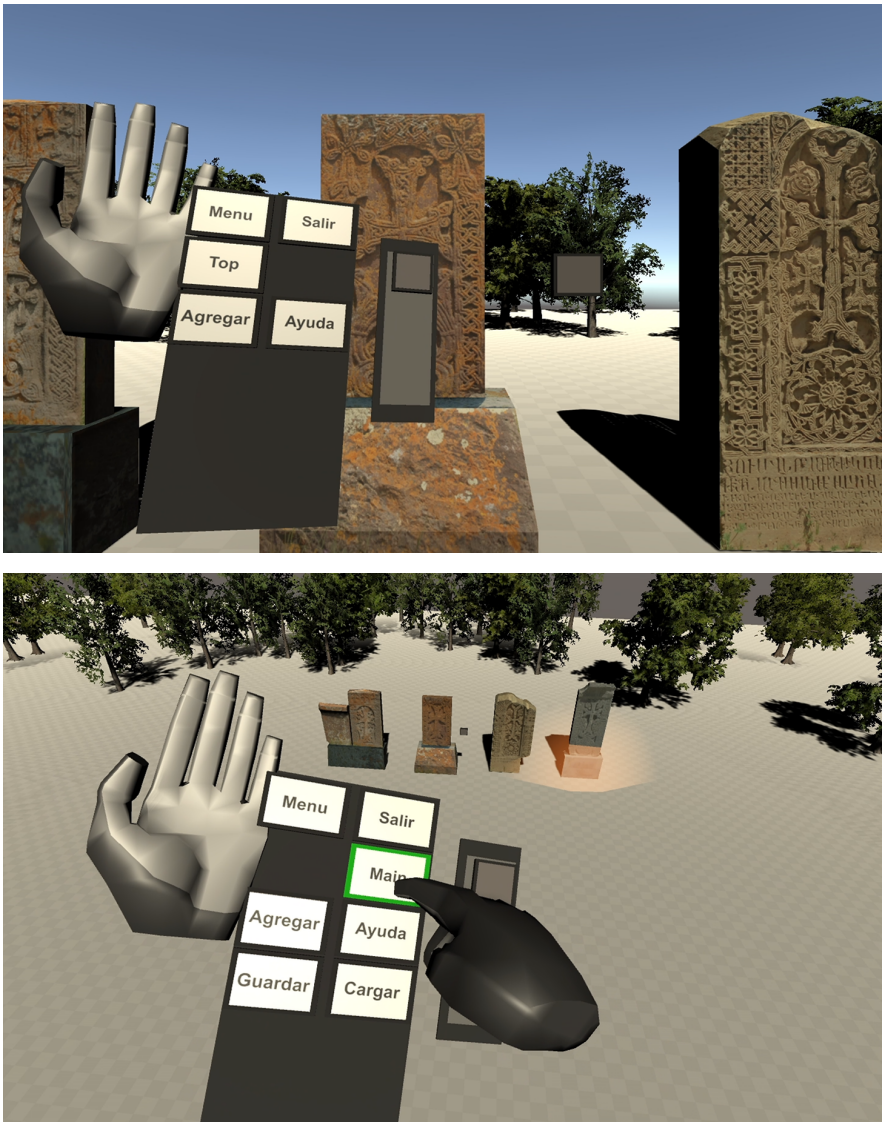

Figura 4.2: Menú de Comandos en la Vista Principal y la Vista Superior

funcionamiento. Por lo tanto, se decidió crear este Menú en un *prefab*, y usar este prefab en todas las salas. Esto además permite que cualquier modificación al prefab (por ejemplo, si se desea agregar un nuevo botón) se vea reflejada en todas las salas.

El nuevo módulo `HandMenu` contiene el código que maneja la disponibilidad y las acciones de los botones del Menú de Comandos.

4.1.3. Ir a Vista Superior

La nueva funcionalidad para pasar de la Vista Principal a la Vista Superior fue implementada en el módulo `CameraStateChanger`, ya que éste es el módulo encargado de hacer el cambio entre cámaras, por lo que ya contiene algunos de los elementos necesarios para efectuar dicho cambio.

Al presionar el botón **Top** del Menú de Comandos, el cambio cumple con los siguientes pasos:

1. Se desactiva el movimiento del usuario, de forma de que no pueda cambiar su posición en la Vista Principal.
2. Se desactiva la Cámara Principal y se activa la Cámara Superior.
3. Se activa el módulo `SelectionManager` que permite manipular los Khatchkars.
4. El LeapHandController cambia de posición, de forma que las manos queden frente a la Cámara Superior. Inicialmente esto se lograba simplemente desplazando el LeapHandController a su nueva posición, pero esto trajo problemas al implementar el movimiento de la Cámara Superior, ya que al moverla también era necesario considerar el desplazamiento del LeapHandController. Para corregir esto se decidió que el LeapHandController debía pasar a ser hijo de la Cámara Superior, de forma que si la cámara se mueve, el LeapHandController automáticamente se mueve con ella.
5. Se cambia la cámara que activa el Menú de Comandos. Ahora la palma de la mano debe orientarse hacia la Cámara Superior.
6. Se activan los detectores de agarre de Khatchkars (sección 4.2.1) y de movimiento de la cámara (sección 4.3.3).
7. Se activa la Zona de Eliminación (sección 4.3.2)

4.1.4. Obtener Ayuda

Para informar al usuario sobre los distintos gestos que puede hacer para interactuar con el Museo, se agregó un botón etiquetado "Ayuda" en el Menú de Comandos. Al presionarlo, se despliega un mensaje de ayuda cuyo contenido depende de la Vista en la que se encuentra el usuario (Principal, Superior, Información Adicional). En la figura 4.3 puede verse uno de estos mensajes.

4.1.5. Agregar Khatchkar

Para permitir al usuario agregar Khatchkars a la sala, se decidió implementar un panel adicional para el Menú de Comandos. Este panel contiene botones que corresponden a cada uno de los Khatchkars disponibles; cuando el usuario presiona uno de estos botones, el Khatchkar apropiado aparece en la sala, en la ubicación correspondiente al centro de la pantalla.

Debido a que la cantidad de Khatchkars disponibles (46 en esta versión del Museo) es muy grande para mostrar todos los botones a la vez, se optó por un formato de "páginas": solo se muestran cuatro botones a la vez. Al avanzar (o retroceder) se muestran los siguientes (o los anteriores) cuatro Khatchkars. El panel cuenta con botones para avanzar y retroceder, como

Figura 4.3: Uno de los mensajes de ayuda

se ve en la figura 4.4. Además, todos los botones presentes utilizan SimpleInteractionGlow para entregar retroalimentación visual al usuario.

Se eligió tener cuatro botones por página porque se encontró que otorgaban un buen balance entre el tamaño del menú y el tamaño de los botones. Un mayor número de botones requiere que (1) se aumente el tamaño del panel, o (2) se reduzca el tamaño de los botones, lo que los vuelve más difíciles de manipular. Por otro lado, un número de botones por página muy pequeño tiene como consecuencia que el usuario debe realizar muchos cambios de página para encontrar el Khatchkar que desea agregar.

Para activar y desactivar el panel, basta con presionar el botón “Agregar” del Menú de Comandos.

4.1.6. Guardar Configuración

Guardar una configuración requiere que el usuario ingrese el nombre que le quiere dar a la configuración. Para poder hacer esto sin usar el teclado del PC, se decidió crear un panel con el que el usuario puede interactuar usando sus manos. El panel contiene un campo de texto, teclas que el usuario puede presionar para ingresar el nombre y botones para confirmar y cancelar la operación. Se optó por organizar las teclas de forma similar a las teclas de un teclado QWERTY estándar, para que la ubicación de las teclas sea familiar para una usuario normal. El panel puede verse en la figura 4.5.

Al presionar las teclas, éstas envían el evento de teclado apropiado al campo de texto, que se encarga de procesar los eventos y cambiar su contenido según corresponda. Esta forma de implementar las teclas permite que en el futuro se puedan agregar más teclas de ser necesario.

Si el usuario intenta guardar una configuración que ya existe, el panel pregunta si el usuario desea sobrescribir dicha configuración. En esta situación, el usuario puede volver a

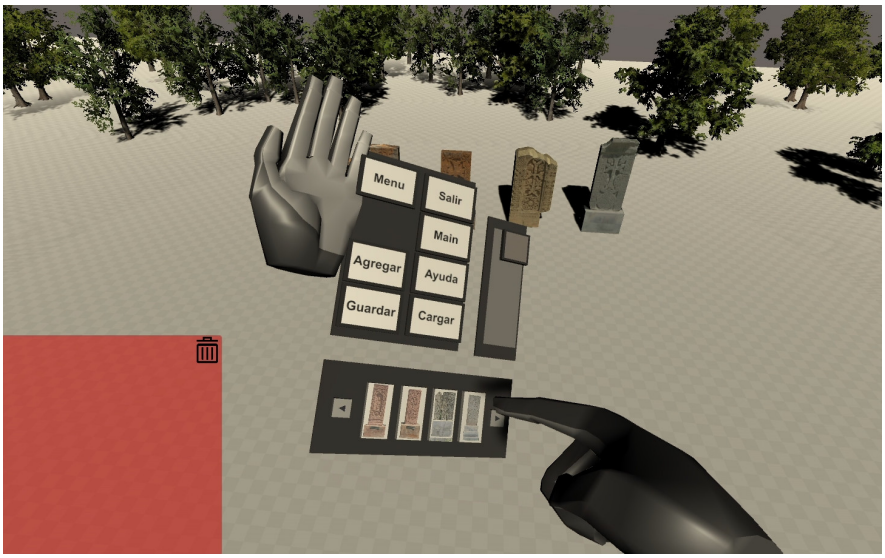

Figura 4.4: Panel para agregar Khatchkars

Figura 4.5: El nuevo panel para guardar configuraciones

presionar el botón verde para confirmar la sobrescritura o el botón rojo para cancelarla.

Almacenar una configuración requiere obtener la información necesaria para poder reproducirla. Esta información incluye:

- Cuáles Khatchkars están en la Sala.
- Las ubicaciones de los Khatchkars.
- Las orientaciones de los Khatchkars.
- El nombre de la Sala.

Este último dato es necesario para que el usuario sólo pueda cargar las configuraciones correspondientes a la sala en la que se encuentra. Un ejemplo de cómo se almacena una configuración puede verse en el anexo B. En teoría es posible compartir las configuraciones almacenadas con otros usuarios que ejecutan la aplicación en sus propios PCs, pero esto no fue testeado.

Durante la implementación del panel, fue necesario considerar que el panel se encuentra en la misma Vista Superior, tapando los demás elementos mientras se encuentra activo. En otras palabras, la presencia del panel por sí misma no impide que el usuario pueda agarrar y mover un Khatchkar, mover la cámara, etc. Por lo tanto, se decidió desactivar dichas funciones mientras el panel se encuentre abierto.

Todas las teclas y botones del panel utilizan SimpleInteractionGlow para entregar retroalimentación visual al usuario.

Para abrir el panel, basta con presionar el botón "Guardar" del Menú de Comandos.

4.1.7. Cargar Configuración

Al igual que en el caso del panel de guardado, se decidió implementar un panel de carga que permite al usuario elegir una configuración para cargarla. El panel contiene una lista de botones, donde cada uno corresponde a una de las configuraciones guardadas que corresponden a la sala actual. La configuración será cargada al presionar uno de los botones. Dado que en principio no hay límite para el número de configuraciones guardadas, se decidió organizar los botones de carga en forma de "páginas": el panel sólo muestra cuatro configuraciones, con botones que permiten al usuario "avanzar" (ver las cuatro siguientes) o "retroceder" (ver las cuatro anteriores). Al igual que en el panel para agregar Khatchkars, se decidió que cuatro botones por página era un número apropiado. El panel puede verse en la figura 4.6.

El panel de carga también cuenta con un botón para cerrarlo. Al igual que con el panel de guardado, el Menú de Comandos estará desactivado mientras el panel de carga esté abierto. Para abrir el panel, basta con presionar el botón "Cargar" del Menú de Comandos.

La aplicación lee las configuraciones almacenadas cada vez que se abre el panel. Esto se hace por dos razones: en primer lugar, es necesario para poder mostrar únicamente las configuraciones que corresponden a la Sala actual. Por otro lado, las configuraciones guardadas

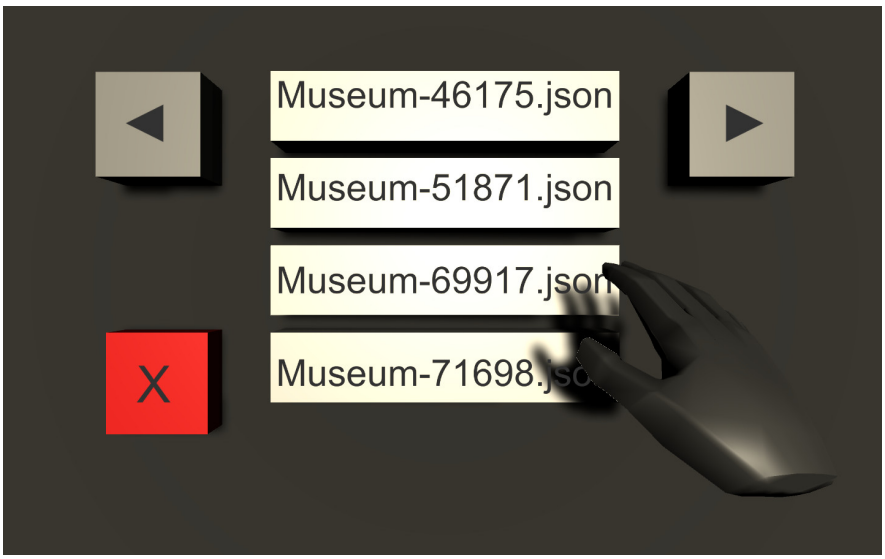

Figura 4.6: El nuevo panel para cargar configuraciones

podrían cambiar desde la última vez que se abrió el panel. Para reflejar esos cambios, es necesario leer las configuraciones en cada apertura.

4.1.8. Volver a la Vista Principal

La implementación de la funcionalidad para volver a la Vista Principal requiere considerar dos casos:

1. Volver a la Vista Principal desde la Vista Superior.
2. Volver a la Vista Principal desde la Vista de Información Adicional.

En ambos casos, el usuario puede volver a la Vista Principal presionando el botón "Main" del Menú de Comandos, pero el procedimiento para efectivamente volver a la Vista Principal es distinto.

En el caso 1, ambas Vistas utilizan la misma escena. Para volver a la Vista Principal, básicamente se revierten los pasos mencionados en la sección 4.1.3, es decir:

1. Se desactiva la Zona de Eliminación.
2. Se desactivan los detectores de agarre de Khatchkars y de movimiento de la Cámara Superior.
3. Se desactiva el módulo `SelectionManager`.
4. Se desactiva la Cámara Superior y se activa la Cámara Principal.
5. El LeapHandController cambia de posición y ahora sigue la posición de la Cámara Principal.

6. Se cambia la cámara que activa el Menú de Comandos. Ahora la palma de la mano debe orientarse hacia la Cámara Superior.
7. Se activa el movimiento del usuario dentro de la Vista Principal.

En el caso 2, las vistas utilizan escenas distintas. Para volver a la Vista Principal, se necesita información sobre dónde estaba el usuario antes de ingresar a la Vista de Información Adicional. Para lograr esto, la posición del usuario y el nombre de la escena son almacenados en StaticValues cada vez que el usuario ingresa a la Vista de Información Adicional. Con esta información, es posible colocar al usuario en el lugar apropiado de la escena apropiada al presionar el botón "Main" del Menú de Comandos.

4.2. Interacción mediante Manipulación de Objetos

4.2.1. Mover Khatchkars

Para mover Khatchkars, se decidió que sólo la mano derecha pudiera moverlos. Esto deja la mano izquierda libre para realizar otras operaciones, como mover la vista. Debido a que las manos no estarán en contacto directo con los Khatchkars, sino que estarán a una distancia que no permite manipularlos directamente, el problema de mover Khatchkars usando las manos tiene dos partes:

1. Detectar si las manos están "encima" de los Khatchkars, de forma similar al puntero de un mouse.
2. Agarrar los Khatchkars y utilizar la posición de la mano para modificar la posición del Khatchkar agarrado.

Para la parte 1, se decidió utilizar un `Raycast` de Unity para detectar si la mano derecha está sobre un Khatchkar: mientras la mano derecha esté en la pantalla, se lanza un rayo con origen en la cámara y con dirección hacia la mano. Si el rayo choca con un Khatchkar, quiere decir que la mano está sobre dicho Khatchkar, que se iluminará para indicar que la mano está sobre él. La figura 4.7 muestra cómo funciona esta detección.

Para la parte 2, inicialmente se implementó un módulo nuevo llamado `StoneGrasper`, cuyo propósito es detectar cuando el usuario está haciendo el gesto de agarrar con su mano derecha. El StoneGrasper consiste en un objeto invisible que siempre se encuentra en frente de la palma de la mano derecha del usuario, de forma que cuando el usuario cierra su mano StoneGrasper detecta que el usuario está haciendo el gesto de agarrar. Si la mano del usuario está sobre un Khatchkar cuando se inicia el agarre, el Khatchkar se moverá siguiendo a la mano, pero manteniéndose en el piso. Al dejar de hacer el gesto de agarre (i.e. al soltar el StoneGrasper), el Khatchkar dejará de moverse.

Sin embargo, al hacer pruebas básicas de uso, se determinó que la detección del gesto de agarre era inconsistente: frecuentemente había que hacer múltiples intentos antes de que el

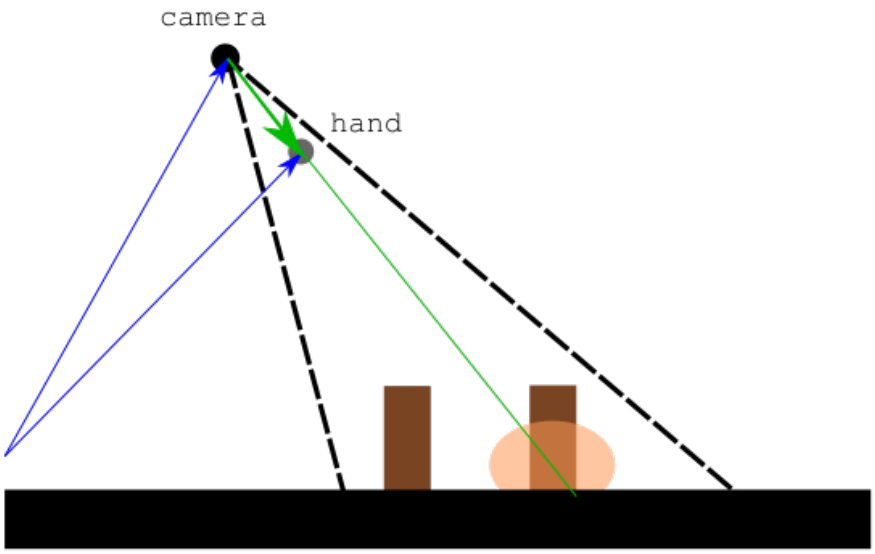

Figura 4.7: Mecanismo para detectar si la mano está sobre un Khatchkar

Khatchkar fuera agarrado. Esta situación, más el éxito obtenido con el sistema de detección de agarre para mover la Cámara Superior (ver sección 4.3.3), llevó a la decisión de utilizar un mecanismo de detección similar. Al igual que con el movimiento de la cámara, el nuevo sistema mide constantemente qué tan "cerrada" está la mano (mediante el método `GetFistStrength` de la API de Leap Motion) para detectar el agarre.

Finalmente, se decidió corregir un problema presente en la versión original del Museo: nada impide que existan superposiciones entre Khatchkars. Para evitar superposiciones entre los Khatchkars, se les agregó un componente de tipo Rigidbody a cada uno de ellos, de forma que actúen como cuerpos rígidos para el propósito del desplazamiento.

4.2.2. Orientar Khatchkars

Una vez implementado el sistema que permite desplazar los Khatchkars, permitir al usuario cambiar la orientación de estos fue relativamente simple: mientras el Khatchkar está agarrado, el usuario puede rotar su mano con respecto a su eje z (en Unity, el eje z es el vector que apunta hacia "adelante"). Esta rotación de la mano genera una rotación del Khatchkar con respecto a su vertical, cambiando así su orientación. Se

4.2.3. Examinar Khatchkars

Para examinar los Khatchkars en la Vista de Información Adicional, se decidió permitir al usuario manipular el Khatchkar con sus manos. Para lograr esto fue necesario completar dos pasos:

1. Modificar el Khatchkar presente en la Vista de Información Adicional, de forma que permita ser manipulado por el usuario.

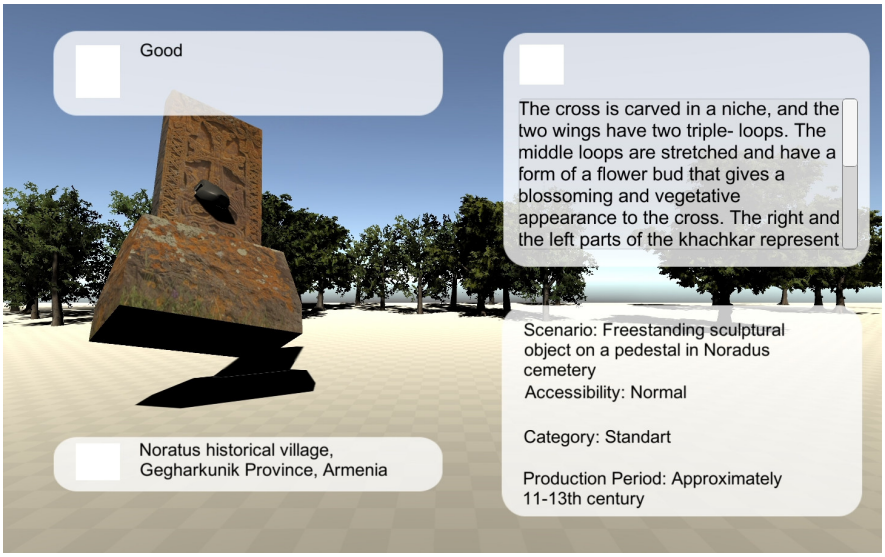

Figura 4.8: Manipulación de Khatchkar en la Vista de Información Adicional

2. Modificar la posición del LeapHandController, de forma que el Khatchkar quede al alcance de las manos del usuario.

Para lograr el paso 1, simplemente se hizo cambios en el proceso de instanciación del Khatchkar: uno de los pasos consiste en agregarle un componente de tipo `InteractionBehaviour`. Este componente es provisto por Leap Motion y permite convertir cualquier objeto en un objeto manipulable por el usuario.

Para lograr el paso 2, fue necesario considerar que la nueva posición del LeapHandController significa que es más difícil leer y presionar los botones del Menú de Comandos, ya que se encuentra más lejos de la cámara. Para compensar, se decidió agrandar el Menú de Comandos cuando el usuario se encuentra en la Vista de Información Adicional.

El resultado puede verse en la figura 4.8.

4.3. Otras Interacciones

4.3.1. Movimiento del Usuario

Para permitir el movimiento del usuario en la Vista Principal, la idea inicial era la siguiente: implementar un nuevo módulo, llamado `MovePointer`, que detectara si el usuario está apuntando con su mano derecha y moverlo en esa dirección mientras esté apuntando. La API del sensor Leap Motion permite detectar cuáles dedos de la mano están extendidos, así como la dirección hacia la que apuntan, por lo que se decidió lo siguiente: *apuntar* significa tener únicamente el dedo índice de la mano derecha extendido. El usuario será desplazado según la dirección hacia la que esté apuntando el dedo índice. El movimiento del usuario ya está limitado al plano horizontal gracias a FPSController, lo que permite utilizar directamente la dirección del dedo índice, incluso cuando tiene una componente vertical.

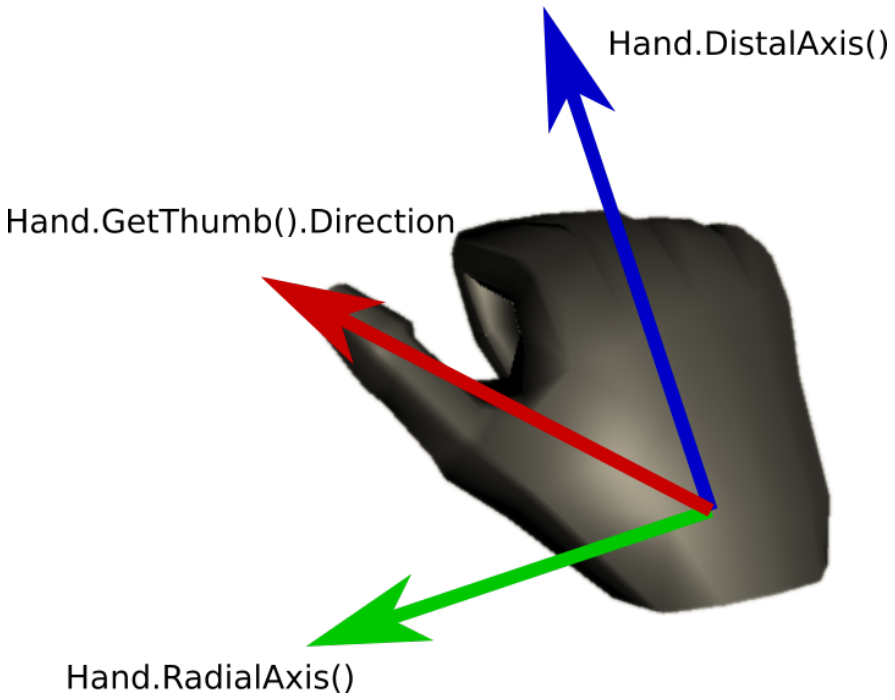

Figura 4.9: Vectores entregados por `Hand.GetThumb().Direction` y `Hand.RadialAxis()`

Esta solución tiene una limitación importante: es difícil para el usuario apuntar hacia atrás o hacia la derecha. Para solucionar esto se decidió incluir al pulgar de la mano derecha: si el usuario tiene únicamente el índice o únicamente el pulgar extendido (pero no ambos), será desplazado según la dirección del dedo que tenga extendido.

Al hacer esto, surgió otro problema: el desplazamiento usando el pulgar ocurre en una dirección distinta a la esperada. Se espera que el movimiento ocurra en una dirección perpendicular a los demás dedos de la mano, pero la dirección del movimiento real es notoriamente desviada con respecto a dicha expectativa. Como se ve en la figura 4.9, la dirección del pulgar extendido no es perpendicular a los demás dedos, ni siquiera cuando el pulgar se encuentra completamente extendido. Para solucionar esto, se decidió no utilizar la dirección del pulgar extendido, sino la dirección correspondiente al eje radial de la mano, también provista por la API de Leap Motion.

Finalmente, se decidió que el usuario no se moverá mientras el Menú de Comandos esté abierto. Se tomó esta decisión porque para presionar los botones del Menú de Comandos, el usuario frecuentemente utiliza su dedo índice. Esto llevaría al usuario a moverse en forma accidental o no deseada; bloquear el movimiento del usuario evita este problema.

4.3.2. Eliminar Khatchkar

Para eliminar Khatchkars desde la Vista Superior se consideraron dos opciones.

La primera opción consistía en “apretar” el Khatchkar que se desea eliminar entre los dedos pulgar e índice de la mano derecha. Esta opción tenía a su favor el hecho de que la

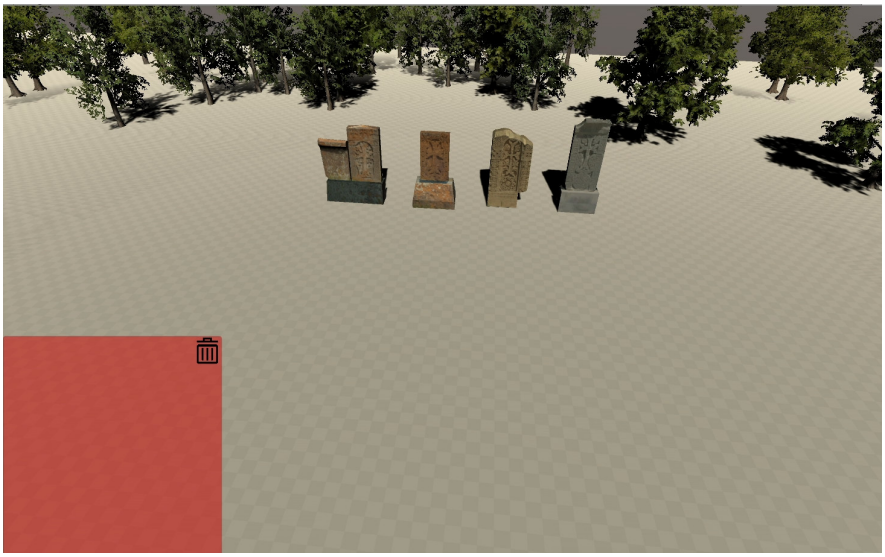

Figura 4.10: Zona de eliminación en la parte inferior izquierda de la pantalla.

API de Leap Motion permite detectar fácilmente dicho gesto, e incluso medir la separación entre los dedos. Sin embargo, los primeros intentos por implementar esta opción se toparon con un obstáculo: la dificultad para distinguir entre el gesto para eliminar (juntar índice y pulgar) y el gesto para agarrar y mover un Khatchkar (cerrar la mano).

Ante estas dificultades, se tomó la decisión de implementar la segunda opción: habilitar una "zona de eliminación" en la pantalla, con forma rectangular, tal que los Khatchkars son eliminados cuando el usuario los suelta en dicha zona. Esta zona fue implementada en el nuevo módulo `DeleteZone`. Dado que la zona de eliminación no corresponde a un espacio físico, sino a un espacio en la pantalla, fue necesario implementar un módulo adicional llamado `StoneRect`. Cuando el usuario agarra un Khatchkar, StoneRect se encarga de calcular el espacio rectangular que dicho Khatchkar ocupa en la pantalla, así como el porcentaje de dicho rectángulo que se encuentra dentro de la zona de eliminación. Si el usuario suelta el Khatchkar cuando este porcentaje se encuentra por sobre un umbral determinado, el Khatchkar es eliminado de la Sala. Cuando el umbral de eliminación es superado, la zona cambia de color para dar a entender al usuario que el Khatchkar será eliminado si lo suelta.

4.3.3. Desplazar Vista

Para mover la cámara en la Vista Superior, inicialmente se consideró la posibilidad de utilizar un sistema similar al descrito en la sección 4.3.1. Esta solución cuenta con dos limitaciones importantes.

En primer lugar, la mano derecha ya realiza múltiples operaciones. Además, puede que el usuario requiera mover la cámara mientras realiza alguna de ellas (por ejemplo, mover la cámara mientras mueve uno de los Khatchkars, en caso de que desee colocarlo en una ubicación lejana a la original). Para evitar este problema, se decidió que la cámara en la Vista Superior será controlada con la mano izquierda.

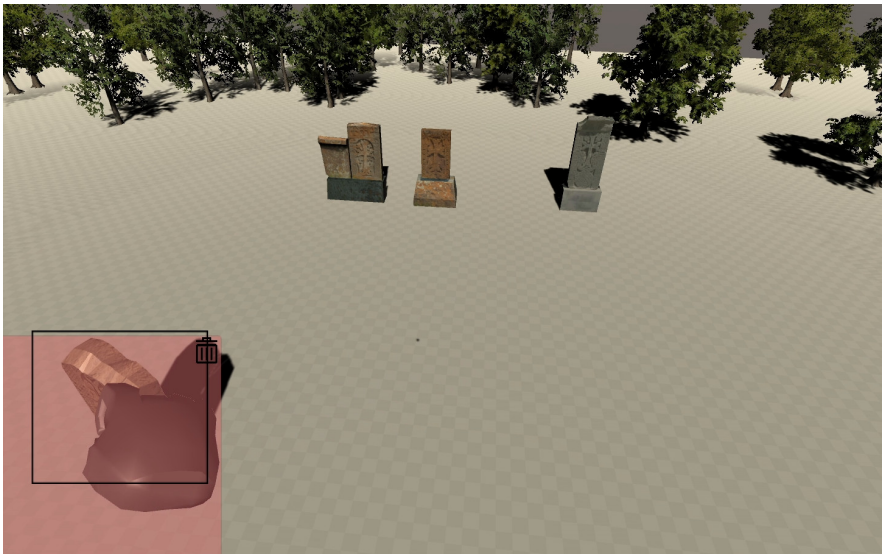

Figura 4.11: Khatchkar a punto de ser eliminado.

En segundo lugar, el movimiento del usuario en la Vista Principal se realiza únicamente en el plano horizontal, e incluso en esas condiciones la mano derecha no permite apuntar en todas las direcciones posibles. En la Vista Superior, el movimiento es en tres dimensiones, las limitaciones de la mano son más evidentes y es más difícil detectar cuando el usuario está apuntando hacia arriba, debido a las limitaciones del sensor Leap Motion (básicamente, cuando el sensor está en modo Desktop, el dedo que apunta hacia arriba está tapado por el resto de la mano).

Una solución que evita los problemas de la segunda limitación consiste en utilizar un sistema similar al de "tocar y arrastrar" utilizado en dispositivos con pantalla táctil. En el caso del Museo, el sistema consiste en "agarrar y arrastrar" con la mano izquierda, implementado en un nuevo módulo llamado `CameraPanner`. Cuando CameraPanner detecta que el usuario está agarrando con su mano izquierda, determina cuánto se desplazó la mano con respecto a su posición inicial (cuando se inició el agarre) y utiliza ese desplazamiento para calcular la nueva posición de la cámara.

4.3.4. Scrolling de Texto

El Scrollbar de Unity utiliza un valor de tipo *float* entre 0 y 1 que representa la posición del Scrollbar: 0 significa que está en el inicio, 1 representa el final. Por otro lado, el script de Leap Motion `InteractionSlider` permite que un objeto pueda "moverse" entre 2 valores de tipo *float* cualesquiera. Para aprovechar esta funcionalidad, también cuenta con el parámetro `VerticalSlideEvent` (en el caso de sliders verticales) que permite designar un método que será llamado cada vez que cambie el valor del slider. Dadas estas herramientas provistas por Leap Motion, se decidió implementar un *Scroller*: un objeto de tipo `InteractionSlider` que el usuario puede manipular con su mano derecha. Este Scroller permite modificar la posición del Scrollbar presente en la Vista de Información Adicional.

El Scroller es parte del Menú de Comandos, pero sólo aparece cuando el usuario se encuen-

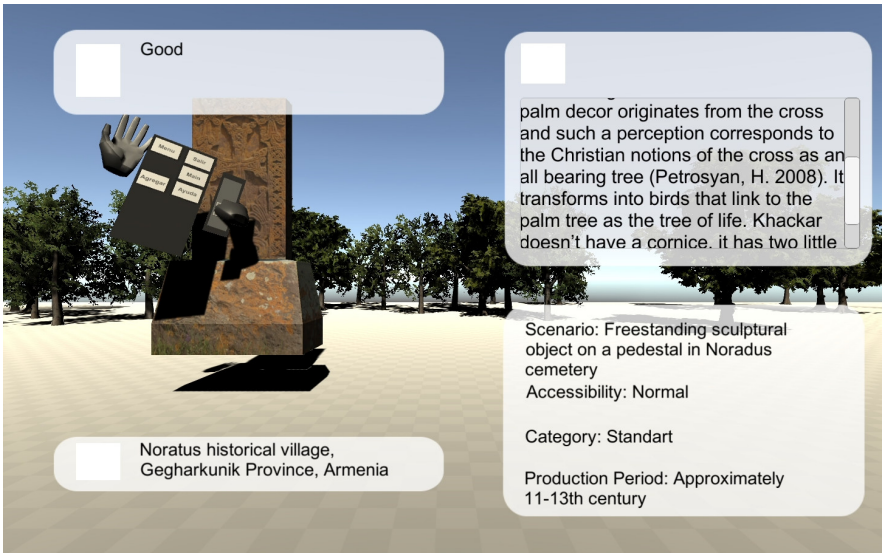

Figura 4.12: El scrollbar y el Interaction Slider asociado

tra en la Vista de Información Adicional. Al deslizar el Scroller hacia arriba y hacia abajo, la posición del scrollbar cambia. Además, mediante el uso de `SimpleInteractionGlow`, el slider cambia de color cuando está siendo manipulado por el usuario.

El scrollbar original tiene la siguiente limitación: no se ajusta al tamaño del texto. En otras palabras, al poner el scrollbar en su posición final, el texto avanza una cantidad fija de líneas, sin tomar en cuenta si queda o no texto. Esto se solucionó agregando un componente de tipo Content Size Fitter al scrollbar.

4.4. Realidad Virtual

Para agregar al proyecto la funcionalidad de Realidad Virtual se optó por utilizar la tecnología Trinus VR. Trinus VR permite utilizar un teléfono celular como visor de Realidad Virtual. Para lograr esto, Trinus VR utiliza dos componentes:

- El *servidor* se instala en la misma máquina donde se ejecuta la aplicación que se desea ver en Realidad Virtual (en este caso, el Museo). El servidor se encarga de capturar la imagen generada por la aplicación, convertirla en una imagen de Realidad Virtual (es decir, una para cada ojo) y enviarla al cliente.
- El *cliente* consiste en una aplicación que se ejecuta en el celular que se utilizará como visor. El cliente tiene dos funciones: en primer lugar, recibe las imágenes enviadas por el servidor y las muestra en la pantalla del celular. En segundo lugar, captura información del movimiento del celular y la envía al servidor para ser interpretada. La interpretación que el servidor da a esta información es configurable.

Para el propósito del Museo, se optó por configurar el servidor de forma que interprete el movimiento del celular como movimientos del mouse. De esta forma, cuando el celular se

encuentra dentro del casco y el usuario gira su cabeza, el servidor moverá el mouse, lo que se traduce en el movimiento esperado dentro de la Sala. La implementación de esta funcionalidad fue una de las últimas en realizarse, lo que tuvo sus consecuencias. Hasta ese punto, se había trabajado con una resolución de imagen con una relación de aspecto arbitraria (en este caso, 16:10). Sin embargo, dado que el cliente necesita mostrar una imagen para cada ojo, utilizando una pantalla con relación de aspecto 16:9 o incluso 2:1, se requiere que la imagen de origen tenga una relación de aspecto de 8:9 o 1:1. En Unity, pasar de 16:10 a 1:1 significa que se deja de mostrar el "exceso" de imagen a los costados, por lo que hubo que realizar los siguientes cambios:

- Alejar la cámara del panel utilizado para el Menú Principal.
- Disminuir el tamaño de los paneles para guardar y cargar, de forma que aparecieran enteros en la pantalla.
- Ajustar los tamaños y posiciones de los elementos de interfaz de usuario en *screen space*, sobre todo los presentes en la Vista de Información Adicional.

Capítulo 5

Evaluación

La evaluación de este proyecto consistió principalmente en verificar el funcionamiento correcto de todas las interacciones implementadas.

Se verificó que es posible realizar todas las interacciones, pero se detectaron algunos problemas.

Ocasionalmente el sensor tiene dificultades al momento de detectar las manos del usuario. Usualmente este problema se manifiesta como la detección de manos que no están presentes en la realidad, lo que lleva a situaciones como:

- Detección ocasional de la mano derecha como si fuera una mano izquierda, lo que impide temporalmente la realización de algunas acciones. Cuando esto ocurre, la solución consiste en sacar la mano derecha del campo de visión del sensor y reintroducirla.
- Aparición ocasional de una mano izquierda "fantasma", con movimientos aparentemente aleatorios y erráticos. Esta mano fantasma puede incluso ser generada fuera del área visible en la pantalla. Además, los movimientos erráticos de la mano fantasma pueden ocasionar que se abra el Menú de Comandos y se termine presionando uno de los botones, generalmente el botón que cierra la aplicación. Esto lleva a que, por ejemplo, ocasionalmente la aplicación se cierre sin intervención por parte del usuario. La mano fantasma tiende a aparecer cuando el usuario no tiene su mano izquierda dentro del campo de visión del sensor, por lo que la solución cuando esto ocurre consiste en introducir la mano izquierda. Además, para evitar este problema cuando la mano fantasma está fuera de la pantalla, se decidió impedir la apertura del Menú de Comandos a menos que la mano izquierda se encuentre dentro de la pantalla.

Otro problema encontrado fue que ocasionalmente, los botones no responden al contacto por parte del usuario. Cuando esto ocurre, el botón se ilumina, es decir, sabe que la mano del usuario está cerca; sin embargo, dicho botón no se mueve al ser presionado por el usuario y el evento que debería ocurrir al ser presionado no ocurre.

Además de esta verificación, se realizaron tests unitarios para evaluar el funcionamiento correcto del código implementado. Unity permite la implementación de tests en modo "Edit"

y en modo “Play”. Los tests en modo Edit se ejecutan en el editor de Unity y se utilizan principalmente para código que no depende del ciclo de ejecución de Unity. Para el código que sí requiere que el ciclo de ejecución de Unity se encuentre activo, se utilizan los tests en modo “Play”, que se ejecutan como corutinas durante la ejecución de la aplicación.

Si bien se buscó cubrir la mayor cantidad posible de líneas de código, medir esta cobertura resultó imposible. Este proyecto fue implementado usando la versión 2018.3 de Unity, debido a que la versión original del Museo fue implementada con esa misma versión y presentaba problemas de compatibilidad con versiones más modernas. Sin embargo, Unity recién incorporó funcionalidad de cobertura de código en su versión 2019.3 y no se encontraron herramientas alternativas que cumplieran esa misma función.

Se consideró en algún momento la posibilidad de realizar pruebas de usuario para evaluar en forma básica la usabilidad de las nuevas interacciones. Esta posibilidad se descartó por dos razones. En primer lugar, desde un comienzo se definió que el alcance del proyecto solo consideraba implementar las interacciones, dejando la evaluación de estas u otras a un eventual proyecto futuro. En segundo lugar, este proyecto fue desarrollado durante la pandemia de COVID-19. El sensor no es un dispositivo que muchas personas posean, lo que limita la posibilidad de realizar pruebas en forma remota. Por otro lado, debido a la pandemia fue difícil encontrar un número suficiente de personas para realizar las pruebas en forma presencial.

Debido a estas complicaciones, se optó por utilizar un método de evaluación llamado *recorrido cognitivo*. Este método se basa en el análisis de los pasos que daría un usuario para lograr un objetivo. En primer lugar, se definieron los siguientes objetivos para el Museo Virtual:

- Ingresar a una Sala
- Agregar Khatchkar
- Mover y orientar Khatchkar
- Eliminar Khatchkar
- Guardar configuración
- Cargar configuración
- Examinar Khatchkar
- Leer información adicional sobre Khatchkar
- Recorrer la Sala

Luego, para cada uno de estos objetivos se determinaron las acciones necesarias para lograr el objetivo. Para cada una de las acciones se buscó responder las siguientes preguntas:

- ¿Entenderá el usuario que la acción es necesaria para lograr el objetivo?
- ¿Notará el usuario que la acción está disponible?

- ¿Asociará el usuario la acción correcta con el efecto deseado?
- ¿Notará el usuario que hubo un avance hacia el objetivo?

El análisis de las respuestas a estas preguntas reveló los siguientes problemas:

- En general, la disponibilidad de las acciones no es inmediatamente visible y requiere leer el mensaje de ayuda. Este problema también existe en la versión original del Museo, donde las acciones asociadas a distintas teclas solo se explican en el mensaje de ayuda.
- La opción de activar el Menú de Comandos no se explica en ningún momento. Un usuario nuevo no tendrá cómo saber que tiene dicha opción a su disposición.
- En el Menú Principal, no queda claro el efecto de presionar uno de los botones hasta que es presionado.
- La rotulación de los botones para moverse entre la Vista Superior y la Vista Principal (botones "Top" y "Main") no es la más adecuada, ya que los rótulos no indican el propósito de dichas vistas (por ejemplo, la Vista Superior existe para poder editar la configuración de la Sala, por lo que el botón debería llamarse "Editar" o algún otro nombre que mejor indique el propósito).
- El gesto para ingresar a la Vista de Información Adicional requiere cierta proximidad al Khatchkar elegido, pero no se indica de ninguna manera al usuario si está suficientemente cerca.

Estos problemas fueron solucionados de la siguiente forma:

- Se agregó un mensaje en el Menú Principal que dice "Presione uno de los botones para entrar a una sala", similar al mensaje existente en la versión original del Museo.
- Al igual que en la versión original del Museo, que en la Vista Principal tenía un mensaje indicando la posibilidad de presionar la tecla "H" para obtener ayuda, se agregó un breve mensaje en la Vista Principal que informa al usuario sobre la posibilidad de orientar la palma de la mano izquierda para abrir el Menú de Comandos.
- Se modificó la etiqueta de los botones para ir a las vistas Superior y Principal, llamándose ahora "Editar" y "Volver a Sala".
- Se implementó detección de proximidad a los Khatchkars: cuando el usuario apunta a un Khatchkar con la mano extendida, si el usuario está suficientemente cerca el Khatchkar se ilumina inmediatamente.

Capítulo 6

Conclusiones

Este trabajo tuvo como objetivo incorporar funcionalidad en un museo virtual que permitiera a los usuarios interactuar con dicho museo mediante gestos de las manos. Para cumplir este objetivo se eligió un dispositivo apropiado, se identificaron las interacciones a reemplazar y se diseñaron e implementaron interacciones nuevas que cumplen el propósito correspondiente, reemplazando así todas las interacciones originales. El testeo de estas nuevas interacciones fue acotado pero suficiente para cumplir con el objetivo establecido.

Para lograr la interacción *touchless*, se utilizó un sensor Leap Motion como dispositivo de captura de gestos de las manos en conjunto con tecnología de realidad virtual. En el caso de la captura de gestos, se utilizó principalmente una combinación de interacciones mediante botones físicos y detección de gestos específicos de las manos. El resultado que se logró es que el museo completo es navegable mediante el uso de las manos y la cabeza del usuario, sin necesidad de usar dispositivos como el mouse, teclado o algún control específico.

Los botones se utilizaron para interacciones acotadas y discretas, muchas de las cuales estaban asociadas a botones en la versión original. Esto puede verse con claridad en la implementación del Menú de Comandos, que contiene muchas de las acciones que en la versión original del Museo se activaban mediante la presión de una tecla. Si bien no se realizaron pruebas de usuario, el simple uso para verificar el funcionamiento sirvió para detectar algunos problemas de confiabilidad de los botones. De todas formas, se decidió mantenerlos debido a la facilidad de implementación: cualquier objeto de Unity puede utilizarse como botón si se le agrega el componente apropiado provisto por Leap Motion.

La detección de gestos se utilizó para interacciones prolongadas y continuas, que en la versión original del Museo estaban asociadas a acciones como el movimiento del mouse o la presión sostenida de una tecla. Inicialmente estos gestos eran detectados mediante el uso de objetos invisibles ("detectores") cuyo único propósito era detectar si habían sido agarrados, tocados, etc. Este método resultó ser difícil de utilizar, debido a que la sensibilidad y confiabilidad de la detección dependían de la posición, forma y tamaño de los detectores, lo que impedía ajustar su funcionamiento en forma consistente. Una vez que se aprendió que la API de Leap Motion permite medir y consultar el grado de flexión de dedos individuales, la transición hacia este nuevo método de detección fue inmediata.

La evaluación de los resultados estuvo limitada por las circunstancias. De haber sido posible, se habrían realizado pruebas de usuario, lo que habría permitido identificar, por ejemplo, cuáles de las interacciones implementadas resultan ser más cómodas para el usuario. De todas formas, el uso de técnicas como el recorrido cognitivo permitió detectar y corregir problemas con la implementación de algunas interacciones.

Hecho este proyecto, el siguiente paso es estudiar las interacciones implementadas. Por un lado, es posible que estas interacciones requieran más ajustes o incluso que no sean las más apropiadas y deban ser reemplazadas por otras. Por otro lado, una de las razones por las que se utilizan interacciones *touchless* es para aumentar el grado de inmersión del usuario. Por lo tanto, sería interesante estudiar si esto ocurre en el caso de este Museo.

Bibliografía

[1] Baloian, Nelson, Wolfram Luther, Daniel Biella, Nare Karapetyan, José A Pino, Tobias Schreck, Andres Ferrada y Nancy Hitschfeld: *Exploring collaboration in the realm of virtual museums*. En *CYTED-RITOS International Workshop on Groupware*, páginas 252–259. Springer, 2017.

[2] Kersten, Thomas P, Felix Tschirschwitz y Simon Deggim: *Development of a virtual museum including a 4D presentation of building history in virtual reality*. The International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences, 42:361, 2017.

[3] *Virtual Museum of Canada*. `http://www.virtualmuseum.ca/`, visitado el 2021-03-12.

[4] *The British Museum (@britishmuseum)*. `https://sketchfab.com/britishmuseum`, visitado el 2021-03-12.

[5] Dal Falco, Federica y Stavros Vassos: *Museum experience design: A modern storytelling methodology*. The Design Journal, 20(sup1):S3975–S3983, 2017.

[6] Suhendi, Andang: *Constructivist learning theory: The contribution to foreign language learning and teaching*. KnE Social Sciences, páginas 87–95, 2018.

[7] *Unity WebGL Player: Interactive Museum*. `http://saduewa.dcc.uchile.cl/museum/`, visitado el 2021-03-12.

[8] *Unity*. `https://unity.com/`, visitado el 2021-03-12.

[9] Jung, Timothy, M Claudia tom Dieck, Hyunae Lee y Namho Chung: *Effects of virtual reality and augmented reality on visitor experiences in museum*. En *Information and communication technologies in tourism 2016*, páginas 621–635. Springer, 2016.

[10] Jung, Timothy Hyungsoo y M Claudia tom Dieck: *Augmented reality, virtual reality and 3D printing for the co-creation of value for the visitor experience at cultural heritage places*. Journal of Place Management and Development, 2017.

[11] Puig, Anna, Inmaculada Rodríguez, Josep Ll Arcos, Juan A Rodríguez-Aguilar, Sergi Cebrián, Anton Bogdanovych, Núria Morera, Antoni Palomo y Raquel Piqué: *Lessons learned from supplementing archaeological museum exhibitions with virtual reality*. Virtual Reality, 24(2):343–358, 2020.

[12] Izzo, Filomena: *Museum customer experience and virtual reality: H. BOSCH exhibition case study.* Modern Economy, 8(4):531–536, 2017.

[13] Parker, Eryn y Michael Saker: *Art museums and the incorporation of virtual reality: Examining the impact of VR on spatial and social norms.* Convergence, 26(5-6):1159–1173, 2020.

[14] Home, MW: *Virtual reality at the British Museum: What is the value of virtual reality environments for learning by children and young people, schools, and families.* En *Proceedings of the Annual Conference of Museums and the Web, Los Angeles, CA, USA*, páginas 6–9, 2016.

[15] Martinez, Jonatan, Daniel Griffiths, Valerio Biscione, Orestis Georgiou y Tom Carter: *Touchless haptic feedback for supernatural vr experiences.* En *2018 IEEE Conference on Virtual Reality and 3D User Interfaces (VR)*, páginas 629–630. IEEE, 2018.

[16] Georgiou, Orestis, Craig Jeffrey, Ziyuan Chen, Bao Xiao Tong, Shing Hei Chan, Boyin Yang, Adam Harwood y Tom Carter: *Touchless haptic feedback for VR rhythm games.* En *2018 IEEE Conference on Virtual Reality and 3D User Interfaces (VR)*, páginas 553–554. IEEE, 2018.

[17] Sorce, Salvatore, Vito Gentile, Debora Oliveto, Rossella Barraco, Alessio Malizia y Antonio Gentile: *Exploring Usability and Accessibility of Avatar-based Touchless Gestural Interfaces for Autistic People.* En *Proceedings of the 7th ACM International Symposium on Pervasive Displays*, páginas 1–2, 2018.

[18] Gentile, Vito, Salvatore Sorce, Alessio Malizia, Dario Pirrello y Antonio Gentile: *Touchless interfaces for public displays: can we deliver interface designers from introducing artificial push button gestures?* En *Proceedings of the International Working Conference on Advanced Visual Interfaces*, páginas 40–43, 2016.

[19] März, Paul, Daniel Schwahlen, Stefan Geisler y Thomas Kopinski: *User expectations on touchless gestures in vehicles.* Mensch und Computer 2016–Workshopband, 2016.

[20] Habibi, Pantea y Debaleena Chattopadhyay: *A Left-Hand Advantage: Motor Asymmetry in Touchless Input.* En *Extended Abstracts of the 2019 CHI Conference on Human Factors in Computing Systems*, páginas 1–6, 2019.

[21] Hatscher, Benjamin, Maria Luz y Christian Hansen: *Foot interaction concepts to support radiological interventions.* i-com, 17(1):3–13, 2018.

[22] *Unity Modules: Interaction Engine.* `https://docs.ultraleap.com/unity-api/unity-user-manual/interaction-engine.html`, visitado el 2021-12-15.

Anexo A: Resultados de cuestionario sobre el museo original

Evaluation of the Implemented Application

Questionnaire

Please mark your affiliation(s)
0 Engineer 0 Curator 4 Lecturer 6 Student 4 Visitor 0 Tourist 2 Expert 0 Other

General questions

1) Your age in years: ___18-68_____
2) I visited the exposition on my computer screen xxxxxxxxxxxxxxx
3) I followed a demonstration xxxxxxxxxxxxx
4) I already know virtual khachkar expositions xx Slide shows xxxx Virtual 3D environment with khachkars xxx.

Questions concerning the exposition

5) I visited more than one scenario	Yes xxxxxxxxxxxxxxx	No ☐
6) I had a look at the **complementary information** to the items	Yes xxxxxxxxxx	No xxxxxx
7) I moved/deleted/added stones	Yes xxxxxxxxxxxxxxx	No ☐

Please mark the appropriate answer (Yes/No) or express your degree of approval (++ complete approval, 0 uncommitted, -- total disapproval).

	Strong agree ++	+	neutral 0	strongly disagree -	- -
8) I managed to orient myself in the museum environment	xxx	xxxxxxxxx	xx	x	☐
9) I could approach the stones enough	xxxxx	xxxxxxxxx	xx	☐	☐
10) I had no problems to change between the scenes	xxxxxxxx	xxxxxxxx	☐	☐	☐
11) I had no problems reading the information concerning the stones	xxx	xxx	xxxxxx	xx	☐
12) I liked the free exploration	xxxxx	xxxxxxxxx	xxx	☐	☐
13) I managed well the navigation within the museum	xxxxxxx	xxx	x	xxxxx	x
14) I changed the place of the items	xxxxxxxxx	xxxxx	xx	☐	☐
15) All in all, I like the virtual exposition	xxxxxxxxx	xxxxxxx	☐	☐	☐
16) I think I can learn more about Khatchkars with the tool	xxxxxxxxx	xxxx	xxx	☐	☐
17) It is easy to interact with the software	xx	xxx	xxxxxx	xxx	x

Question concerning actual and further features: I (would) like

18) A tutorial at the beginning of the museum	Yes xxxxxxxxxx	No xxxxx
19) A short slide show with spoken information	Yes xxxxxxxxx	No xxxxxxx
20) An interactive map of Armenia with the sites as navigation aid	Yes xxxxxxxxxxxxxxx	No ☐
21) To modify or enhance the metadata of the Khachkars	Yes xxxxxxx	No xxxxxxxxx
22) To publish my walk in the museum	Yes xxxxxxxx	No xxxxxxxx
23) More information about the inscriptions	Yes xxxxxxxxxx	No xxxxx

Please rank these six new features from most important one (1) to the least important one (6) for you:

	Rank	Σ
A Multi user access (various users creating an exposition)	4 4 6 1 6 4 5 3 6 6 0 3 2 6 6 5	67
B Grouping khachkars in a place using common metadata: e.g., master, age, motif for erection, cross style, ornaments, material, etc.	4 5 2 2 4 1 3 6 5 5 0 5 1 5 4 2	54
C For each khachkar recommending similar khachkars	4 6 4 3 5 3 2 5 3 4 0 6 3 4 5 4	61
D Tour planning	4 3 5 6 3 1 2 2 4 2 0 4 6 4 4 3	53
E Khachkar workbench (creating your own khatchkar)	6 2 1 5 1 1 3 1 1 3 0 2 5 1 1 2	35
F Storytelling based on khachkars' history	6 1 1 3 4 2 1 2 4 2 1 1 4 1 1 2	36

The results needs a comment: People like obviously storytelling, which greatly benefits from enhanced (18.), tour planning (D, 19) and publishing as well as B and information on the inscriptions, features not really positive rated by the majority. People find it not very easy to interact with the software, orient themselves and navigate

Anexo B: Ejemplo de almacenamiento de configuración


```
{
    "sceneName": "Museum",
    "stoneList": [
        {
            "id": 14,
            "position": {
                "x": 120.13999938964844,
                "y": 0.049999237060546878,
                "z": 95.3499984741211
            },
            "rotation": {
                "x": -0.7070798873901367,
                "y": -0.006170541979372501,
                "z": -0.006170541979372501,
                "w": 0.7070798873901367
            }
        },
        {
            "id": 16,
            "position": {
                "x": 130.18173217773438,
                "y": 0.04000091552734375,
                "z": 95.09683227539063
            },
            "rotation": {
                "x": -0.7071068286895752,
                "y": 0.0,
                "z": 0.0,
                "w": 0.7071068286895752
            }
        },
        {
            "id": 18,
            "position": {
                "x": 101.68000030517578,
                "y": -0.090000152587890625,
                "z": 94.04000091552735
            },
            "rotation": {
                "x": -0.7071068286895752,
                "y": 0.0,
```

```
                "z": 0.0,
                "w": 0.7071068286895752
            }
        },
        {
            "id": 26,
            "position": {
                "x": 110.33173370361328,
                "y": 1.2900009155273438,
                "z": 92.61683654785156
            },
            "rotation": {
                "x": -0.7071068286895752,
                "y": 0.0,
                "z": 0.0,
                "w": 0.7071068286895752
            }
        }
    ]
}
```

Anexo C: Estructuras utilizadas para el almacenamiento de configuraciones

```
1 [Serializable]
2 public class SaveStone
3 {
4     public int id;
5     public Vector3 position;
6     public Quaternion rotation;
7 }
8
9 [Serializable]
10 public class SaveStoneList
11 {
12     public string sceneName;
13     public List<SaveStone> stoneList;
14 }
```