Componentes para simplificar el desarrollo de aplicaciones de Realidad Virtual Web

Federico Marino

Facultad de Ingeniería, UBA, Argentina
fmarino@fi.uba.ar

Resumen. La realidad virtual (RV) ha ampliado sus aplicaciones gracias a la plataforma web y dispositivos RV más accesibles. La RV, más que visualización avanzada, es una poderosa herramienta ligada al modelado y simulación computacional, intuitiva y atractiva para interactuar y visualizar sistemas complejos en diversas industrias y ciencias. La plataforma web permite incluso, simulación distribuida con visualización y simulación separadas, sincronizadas en red. Durante el desarrollo de múltiples aplicaciones de RV web, se suele observar una brecha entre las capacidades provistas por las bibliotecas y los mecanismos básicos necesarios en una aplicación típica. Por lo tanto, en este trabajo se proponen un conjunto de componentes de software de alto nivel para facilitar la creación de aplicaciones de RV web. La solución propuesta se enfoca en resolver los desafíos comunes que enfrentan los desarrolladores, como por ejemplo: la gestión de los controladores XR (manejo de eventos, representación 3D, identificación de lateralidad, emisión de rayos para la detección de intersección con superficies, etc.), la creación de interfaces de usuario dinámicas (menús), la navegación de la escena (teleportación y modo de vuelo), la selección y manipulación de objetos virtuales y la provisión de herramientas para depuración. Así, se proponen clases y módulos reutilizables, de código abierto, que se encuentran disponibles en línea. También se presentan un conjunto de ejemplos didácticos que sirven para hacer una evaluación funcional de los módulos, validar los componentes y proporcionar puntos de partida prácticos para el desarrollo de aplicaciones de RV basadas en tecnología web.

Palabras clave: realidad virtual, WebXR, three.js, web inmersiva, interfaz de usuario

Components to simplify the development of Virtual Reality Web applications

Federico Marino

Faculty of Engineering, University of Buenos Aires
fmarino@fi.uba.ar

Abstract. Virtual Reality (VR) has expanded its applications thanks to the web platform and more devices. VR, more than just advanced visualization, is a powerful tool linked to computational modeling and simulation, intuitive and engaging for interacting with and visualizing complex systems across various industries and sciences. The web platform even enables distributed simulation with separate, network-synchronized visualization and simulation. During the development of multiple web VR applications, a gap is often observed between the capabilities provided by libraries and the basic mechanisms needed in a typical application. Therefore, this paper proposes a set of high-level software components to facilitate the creation of web VR applications. The proposed solution focuses on addressing common challenges faced by developers, such as: XR controller management (event handling, 3D representation, laterality identification, Raycasting for surface intersection detection, etc.), the creation of dynamic user interfaces (menus), scene navigation (teleportation and flight mode), the selection and manipulation of virtual objects, and the provision of debugging tools. Thus, reusable, open-source classes and modules are proposed and made available online. A set of didactic examples is also presented to provide a functional evaluation of the modules, validate the components, and offer practical starting points for the development of web-based VR applications.

Keywords: virtual reality, WebXR, three.js, immersive web, user interface

1 Introducción

La realidad virtual (RV) web ha emergido como un campo de rápido crecimiento, democratizando el desarrollo de experiencias inmersivas y ha contribuido a mejorar el aprendizaje según diversos estudios (Meccawy, 2022), gracias a la convergencia de hardware más accesible y herramientas de software poderosas, de código abierto. La reciente proliferación de dispositivos RV autónomos y asequibles, como Meta Quest 3, combinada con la madurez de tecnologías web estandarizadas como WebGL, WebXR (W3C, s.f.) y JavaScript, ha abierto un abanico de oportunidades sin precedentes para crear y distribuir aplicaciones RV directamente a través de la web. Este nuevo paradigma elimina barreras tradicionales, simplificando la publicación y reduciendo los costos de entrada, haciendo que la creación de experiencias RV sea más accesible.

La RV ofrece una plataforma excepcionalmente intuitiva y atractiva para interactuar y visualizar sistemas complejos que han sido modelados y simulados computacionalmente. Al permitir a los usuarios sumergirse en entornos tridimensionales generados por computadora, la RV va más allá de las interfaces tradicionales bidimensionales, facilitando una comprensión más profunda y una interacción más natural con los datos y los modelos subyacentes. La sensación conocida como telepresencia, se logra mediante el uso de sensores de movimiento que rastrean los movimientos del usuario y ajustan la perspectiva visual en tiempo real. Esta capacidad de simular experiencias sensoriales, incluyendo la vista, el oído, transporta a los usuarios a mundos virtuales meticulosamente diseñados, desafiando la percepción de la realidad y abriendo nuevas vías para la exploración y el entendimiento. La naturaleza inmersiva de la RV la convierte en un medio ideal para experimentar los resultados de simulaciones computacionales, obteniendo una comprensión más visceral (ver ejemplos en la figura 1).

Fig. 1. Dos aplicaciones RV Web, “Laboratorio de microbiología” (izquierda) y “Interpretación de Cartas topográfica” (derecha) desarrolladas con Three.js para CITEP UBA, Año 2022.

Existen numerosas industrias que se benefician de la integración de la RV con la simulación. Según un relevamiento (Berg & Vance, 2016) los ejemplos abarcan industrias como la automotriz, agrícola, construcción, energía y hasta la militar.

En conclusión, la RV, el modelado y la simulación computacional mantienen una relación simbiótica fundamental. La RV no solo proporciona una plataforma inmersiva para visualizar los resultados de las simulaciones, sino que también sirve como un entorno para la creación e interacción con modelos de sistemas dinámicos.

1.1 Problema y motivación

Si bien el panorama del desarrollo de RV web se presenta cada vez más accesible y atractivo, la realidad práctica para los desarrolladores a menudo revela una curva de aprendizaje pronunciada en tareas fundamentales (Nebeling & Speicher, 2018) (Meccawy, 2022). Bibliotecas gráficas como Three.js (Three.js, s.f.), aunque potentes y ampliamente utilizadas, ofrecen una interfaz de programación para la interacción con dispositivos RV que se sitúa en un nivel relativamente bajo. Esto implica que, incluso para implementar funcionalidades básicas y ubicuas en experiencias RV, como la detección de eventos de controladores XR (joysticks), la navegación de la escena o la

creación de interfaces de usuario interactivas, los desarrolladores deben escribir código extenso y repetitivo en cada nuevo proyecto.

Esta necesidad de reinventar la rueda en cada aplicación se agudiza por la intrínseca diversidad de las experiencias de RV (Chen et al., 2024). La vastedad de casos de uso, estilos de interacción y objetivos de diseño dificulta la creación de componentes genéricos y universales que puedan satisfacer todas las necesidades. Por ejemplo, en cuanto a la navegación de la escena en RV, muchos sistemas combinan varias metáforas como teleportación libre, caminar o volar, para ofrecer la mejor experiencia (Ahmed & Andujar, 2022). Esta atomización de soluciones y la ausencia de bibliotecas de componentes de alto nivel para RV web contrastan con la creciente madurez del ecosistema, donde se esperaría una mayor disponibilidad de herramientas.

La motivación detrás de este trabajo surge de la experiencia práctica y recurrente en el desarrollo de múltiples aplicaciones de RV que evidenció un patrón de necesidades comunes que requerían una atención inicial en cada nuevo desarrollo. Una de estas necesidades reside en la gestión eficiente y representación 3D de los controladores XR. La API de Three.js, por ejemplo, proporciona información detallada sobre el estado de las palancas (joysticks) y botones, pero carece de abstracciones de más alto nivel que faciliten la interpretación de acciones comunes y significativas para el usuario.

Ilustrativamente, la detección de un simple gesto como sostener la palanca hacia un lado por un breve lapso, no se traduce directamente de los eventos básicos provistos por la API. Implementar una acción así requiere sintetizar nuevos tipos de eventos a partir de los datos de bajo nivel. La implementación de mecanismos de teleportación, esenciales para la movilidad en entornos virtuales, implica resolver cálculos geométricos complejos entre diferentes sistemas de coordenadas, haciendo deseable encapsular esta lógica en un componente reutilizable.

En esencia, el objetivo de este trabajo es proponer una solución basada en componentes de software reutilizables que simplifiquen y aceleren el desarrollo de aplicaciones de RV web. Además, de proporcionar ejemplos prácticos, que pueden utilizarse como puntos de partida para desarrollos de casos de uso más complejos.

1.2 Antecedentes y trabajos relacionados

Si bien, no existen un conjunto de componentes equivalentes contra los cuales realizar una comparación en el contexto de aplicaciones RV web, es posible identificar esfuerzos por crear herramientas con el objetivo común de facilitar su desarrollo.

En un trabajo reciente (Nebeling & Speicher, 2018) se analizan las herramientas de creación de realidad aumentada (RA) y RV existentes, se identifican cinco clases de herramientas y se describen los problemas principales que enfrentan los diseñadores, especialmente aquellos sin conocimientos técnicos. Se destaca la complejidad del panorama de herramientas, la necesidad frecuente de usar múltiples clases para un solo proyecto y las significativas carencias entre y dentro de estas clases.

En otro trabajo (Yigitbas et al., 2022) se presenta una herramienta de desarrollo llamada VREUD, haciendo énfasis en capacitar a los usuarios finales que carecen de conocimientos de programación. VREUD combina métodos basados en componentes y

asistentes paso a paso. Este enfoque, aunque simple de aprender, tiene un costo en términos de flexibilidad y capacidad de crear aplicaciones diversas y complejas.

Las herramientas de desarrollo RV se pueden dividir en tres categorías (Meccawy, 2022). Primero, plataformas de programación, como Unity, Unreal Engine y WebXR, dirigidas a desarrolladores con experiencia técnica. Luego, plataformas de codificación mínima (PlugXR, Adobe Aero, etc.) diseñadas para usuarios con habilidades de programación mínimas o nulas. Por último, las soluciones XR completas, (ClassVR, zSpace y VictoryXR) que proporcionan paquetes con contenido educativo prediseñado, que pueden ser costosas, poco escalables y para una temática específica.

Landmarks (Starrett et al., 2021) es un paquete de herramientas para el motor de videojuegos Unity, diseñado para facilitar la creación de experimentos de navegación espacial y memoria en RV. Su objetivo es simplificar este proceso al ofrecer una interfaz intuitiva de "arrastrar y soltar", sin necesidad de escribir código, aunque es aplicable a un tipo de aplicación específico.

En cuanto a la plataforma web, si bien existen alternativas a la biblioteca Three.js, (como Babylon.js) todas ellas están relegadas a un lugar secundario en cuanto a su nivel de popularidad, versatilidad, madurez, flexibilidad y frecuencia de actualización del proyecto. Distintas métricas como la actividad en el repositorio de Github o en foros de discusión de cada proyecto, sugieren que Three.js está muy por delante del resto de las bibliotecas de visualización 3D web.

1.3 Plataforma de hardware RV utilizada en el proyecto

En el desarrollo de este proyecto se utilizó un dispositivo RV Meta Quest 3. Este posee la capacidad de ejecutar aplicaciones WebXR directamente en el casco mediante un navegador web que forma parte del sistema operativo Meta Horizon OS (del dispositivo). Además, el casco permite la ejecución en modo PCVR en donde un navegador corriendo en una PC con Windows 10 u 11, ejecuta la aplicación WebXR y transmite vía WIFI (Meta Quest Link) las imágenes de cada ojo, al casco en tiempo real. En este modo, es la GPU de la PC la que se encarga del renderizado, permitiendo la visualización de escenas mucho más complejas. Ambos modos son compatibles con los ejemplos del proyecto aquí propuesto.

2 Desafíos y soluciones propuestas

La arquitectura propuesta en este proyecto está basada en componentes JavaScript individuales agrupados en módulos (ver figura 2). Cada uno es responsable de una funcionalidad específica, utilizando los servicios provistos por la biblioteca Three.js. Se buscó abstraer la complejidad de las APIs subyacentes, proporcionando interfaces más intuitivas, asumiendo precondiciones como el uso de dos controladores (izquierdo y derecho) con las características de los dispositivos Meta Quest.

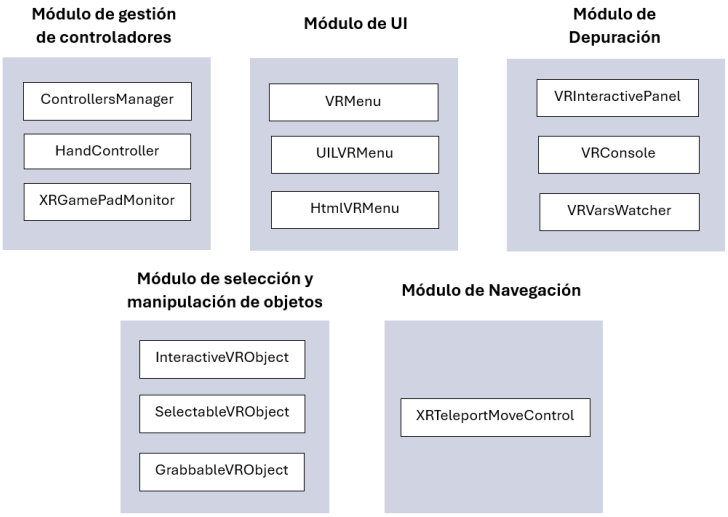

Fig. 2. Esquema de clases y módulos propuestos.

2.1 Módulo de gestión de controladores

La API WebXR contempla la posibilidad de que existan múltiples controladores XR conectados durante una sesión XR, con diversas formas, características. Se mantiene una lista dinámica de controladores (pueden conectarse y desconectarse durante la sesión). Incluso pueden existir más de dos, que pueden estar identificados con la mano derecha, izquierda o ser neutrales. Al desarrollar una aplicación RV, necesitamos encapsular toda esta complejidad de gestión, para poder acceder directamente al control derecho o izquierdo desde la aplicación. Se desea poder registrar llamados para la escucha de eventos, sin importar su estado de conexión. Por otro lado, es necesario sintetizar nuevos tipos eventos de alto nivel relacionados con la emisión de rayos (Raycasting) que son cruciales para implementar interacciones como activar, sujetar y arrastrar objetos 3D o la detección de puntos de destino para la navegación por teleportación. En este proyecto se definió un subsistema basado tres clases interrelacionadas.

Primero, se diseñó la clase XRGamepadMonitor que monitorea el estado de los botones y palancas del controlador XR. Utilizando la Gamepad API (accesible desde Javascript), se detecta cuándo se presionan o sueltan los botones, o cuando se mueven las palancas, para emitir eventos en consecuencia. En el caso de las palancas, genera además eventos sintéticos cuando estas, se mueven rápidamente ida y vuelta en las 4 direcciones (izquierda, derecha arriba y abajo).

Luego se definió, la clase HandController que representa un controlador RV individual. Esta computa el valor del rayo del controlador (un vector dirección, saliendo del frente), gestiona la conexión y desconexión del dispositivo, la representación 3D en la escena virtual y la emisión de eventos de alto nivel para la detección de intersecciones entre el rayo del controlador y los objetos de la escena. Cuando existen en la escena

múltiples objetos interactivos en la línea del rayo saliente del controlador, necesitamos garantizar que solo el objeto más cercano, responda a la interacción. Estos eventos de alto nivel implementan un mecanismo para detener su propagación, evitando la necesidad de programar lógicas de control centralizadas.

Finalmente, se implementó la clase ControllersManager que centraliza la gestión de ambos controladores. Permite acceder a ellos por su lateralidad (obtener el izquierdo o el derecho) o por su rol (podemos definirlos como “mano hábil” o “mano no hábil”). También gestiona y emite eventos que involucran la participación simultanea de ambos controladores, como el gesto "double squeeze" (acercar o alejar ambos).

2.2 Módulo de navegación

El desarrollo de mecanismos de navegación en RV es un campo de investigación activo. Hay trabajos recientes (Prithul et al., 2021) que comparan la teleportación con otros métodos de locomoción en RV, destacando sus ventajas y desventajas.

Para solucionar esta necesidad, se definió la clase XRTeleportMoveControl que combina dos mecanismos de navegación de la escena, la teleportación para desplazamientos rápidos y el modo de vuelo para el ajuste fino de la ubicación. Para teleportarse, el usuario presiona el gatillo del controlador y apunta a una superficie habilitada para teleportarse (ver figura 3 derecha).

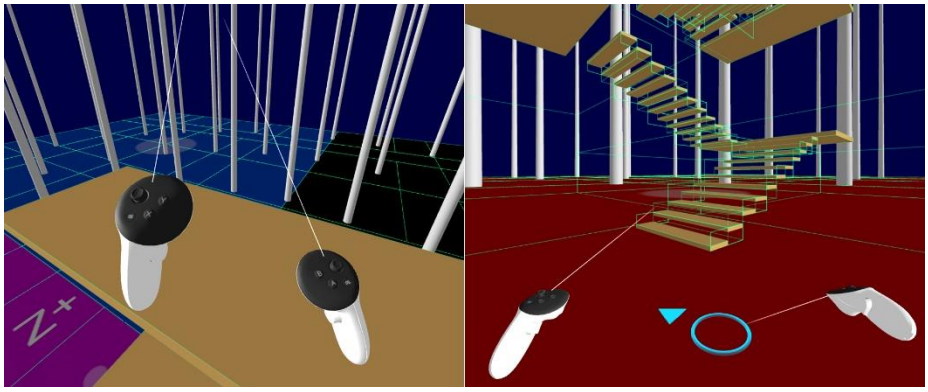

Fig. 3. Ejemplo de uso del módulo de navegación, representación de los controladores XR (izquierda) y uso del mecanismo de teleportación (derecha).

El modo de vuelo permite al usuario volar por la escena en la dirección apuntada por el controlador, con una velocidad variable. Además, le permite variar su orientación en la escena en incrementos fijos utilizando los eventos sintéticos de la palanca. La clase ofrece opciones de configuración para elegir la mano habilitada para la teleportación, activar o desactivar el modo de vuelo, etc. El mecanismo de teleportación requiere que se definan de antemano las superficies hacia donde es válido teleportarse. Se utiliza la biblioteca Three-Mesh-Bvh (Johnson, s.f.) para acelerar la detección de intersecciones de superficie con el rayo del controlador, mediante una estructura del tipo “Bounding Volume Hierachy” (Rubin & Whited, 1980).

En resumen, XRTeleportMoveControl, proporciona una forma intuitiva de navegar por la escena 3D en RV, utilizando teleportación para movimientos rápidos y vuelo para ajustes finos. Además, cuando la escena contiene múltiples objetos interactivos, en la línea del rayo del controlador, el sistema de eventos permite manejar prioridades, permitiendo así que la intersección del rayo con un menú interactivo tenga prioridad sobre el mecanismo de teleportación.

2.3 Módulo UI

Si bien la investigación sobre interfaces de usuario (UI) para RV explora cada vez más las interfaces 3D y los modos de interacción espacial (Kharoub et al., 2019), las interfaces 2D tradicionales siguen siendo relevantes y efectivas para ciertas tareas, especialmente aquellas que requieren precisión y familiaridad, como la interacción con menús y paneles de control.

Considerando además a la flexibilidad que brindan las bibliotecas JavaScript basadas en HTML/CSS para crear menús dinámicamente, se decidió implementar en este proyecto, una solución basada en menús bidimensionales.

Para ello, se diseñó la clase abstracta VRMenu que representa un panel rectangular que utiliza un mapa de texturas actualizado dinámicamente. Este panel, recibe y traduce los eventos generados por los controladores XR, en eventos de puntero (similar a los de un mouse) que luego son transmitidos al componente JavaScript subyacente, encargado de responder a la interacción. Además, ofrece dos modos de visualización. En el modo "panel", el menú se muestra como un panel inclinado, que se ubica delante de la posición actual del usuario. En el modo "swatch", el menú acopla al controlador XR (definido como “no hábil”) en la posición de un reloj de muñeca.

Luego se definieron las dos subclases UILVRMenu y HtmlVRMenu (ver figura 4) que implementan dos mecanismos diferentes para actualizar el mapa de texturas.

La clase HtmlVRMenu utiliza la biblioteca lil-gui (lil-gui, s.f.) para generar menús basados en una estructura HTML. Luego dicha estructura es convertida a un objeto canvas mediante un proceso que itera sobre todas las etiquetas HTML para dibujarlas en un mapa de texturas.

La clase UILVRMenu, en cambio utiliza una versión modificada de biblioteca UIL.js que permite crear menús directamente sobre un mapa de texturas, evitando la conversión desde HTML.

Estas clases sirven de ejemplo para implementaciones más complejas, que pueden basarse en HTML/CSS y ser programadas con bibliotecas como React.js o Vue.js. Cabe destacar que la conversión de estructuras HTML/CSS a elementos canvas tiene varias limitaciones. No todas las etiquetas HTML y atributos son convertidos de manera correcta a su representación en mapas de bits. Bibliotecas como Html2Canvas (Html2canvas, s.f.), son habitualmente utilizadas para realizar este proceso de conversión.

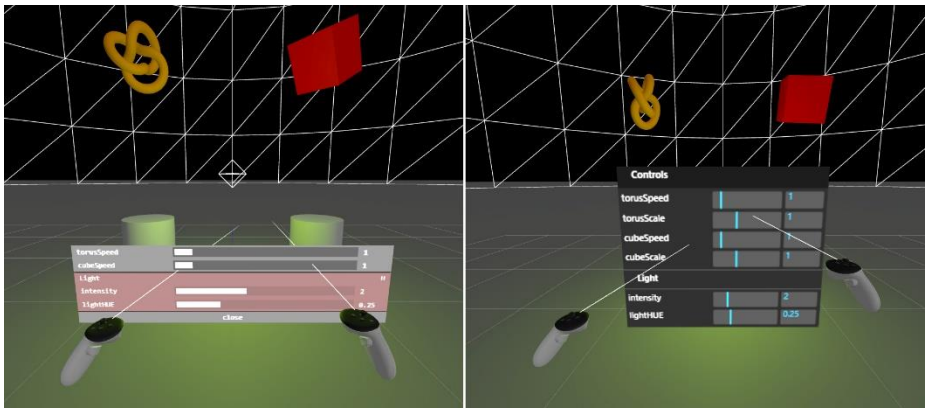

Fig. 4. Ejemplo de uso de clase UILVRMenu (izquierda) y de HtmlVRMenu (derecha).

2.4 Módulo de depuración

La depuración de aplicaciones de RV basadas en web introduce desafíos inherentes a la naturaleza inmersiva y al hardware específico de estos entornos. El ciclo de depuración tradicional se ve interrumpido por la necesidad de alternar repetidamente entre el entorno de desarrollo en el equipo PC y la experiencia inmersiva en el dispositivo RV. Este proceso iterativo, que requiere colocarse y retirarse el casco RV para cada prueba, consume tiempo y fragmenta el flujo de trabajo del desarrollador (Bose & Brown, 2024). Las herramientas de depuración convencionales solo están disponibles en el monitor de la computadora, cuando se trabaja en modo PCVR. En este modo la PC ejecuta la aplicación en un navegador, mientras que el casco RV solo se encarga de mostrar las imágenes resultantes y enviar la información de los sensores.

La interrupción de la renderización durante la inspección de variables o la ejecución paso a paso del código fuente, congela la experiencia para cualquier usuario en la sesión RV, complicando la depuración.

Finalmente, la naturaleza espacial de las aplicaciones RV introduce la dificultad de reproducir y aislar errores que a menudo dependen de secuencias de movimientos y acciones tridimensionales complejas, limitando la efectividad de emuladores, como Inmersive Web Emulator (Meta, s.f.) que no capturan la riqueza y la fidelidad de la interacción real con los controladores XR (Bose, 2024). Para ayudar a solucionar estas dificultades diseñe dos clases.

Se definió la clase VRConsole que permite crear un plano en donde se proyecta la salida de la consola de depuración del navegador en tiempo real. Esto permite al usuario tener una visión inmediata de errores, excepciones o mensajes de depuración.

Luego se implementó, la clase VRVarsWatcher, que permite asociarle un conjunto de variables u objetos a monitorear. Estos se desplegarán y actualizarán dinámicamente durante el ciclo de renderizado, con un formato de tabla (con dos columnas nombre y valor).

En ambos casos los planos se despliegan en una posición fija en la escena virtual.

2.5 Módulo de selección y manipulación de objetos

Existen formas muy variadas de seleccionar y manipular objetos en RV según se detalla en (Yu et al., 2024). Es parte del diseño de cada experiencia encontrar la metáfora más adecuada y resulta difícil definir elementos reutilizables.

En este caso se diseñó un subsistema que permite asociar comportamientos a objetos 3D que hacen posible seleccionarlos, sujetarlos, arrastrarlos o soltarlos, utilizando los controladores XR.

Se definió la clase SelectableVrObject (ver figura 5 izquierda) que escucha eventos generados por los controladores XR, para emitir nuevos eventos cuando: se concreta la selección, cuando el rayo pasa por encima del objeto (para poder resaltar su color).

También se creó la clase GrabbableVrObject (ver figura 5 derecha) que dota a los objetos de la capacidad de ser sujetados mediante el botón secundario del controlador XR. Genera dos tipos de eventos al sujetar y soltar. Mediante opciones de configuración se pueden definir varios modos de funcionamiento.

El modo “remoteGrabbing”, permite tomar un objeto que está fuera del alcance, pero en línea con el rayo del controlador. Al iniciar la acción, el objeto “vuela” hacia el controlador y se mantiene sujeto al mismo hasta concluir la acción. Las opciones de configuración permiten definir la velocidad de acercamiento y la devolución del objeto a su posición original al cancelar el gesto de sujeción.

El modo “contactGrabbing”, permite tomar objetos solo si están en contacto directo con el controlador. Para evaluar el contacto se utiliza el “bounding box” del objeto 3D junto con un umbral de distancia configurable.

Para asociar ambos comportamientos simplemente hay que instanciar la clase e implementar las funciones que escucharan los eventos generados.

3 Evaluación funcional mediante ejemplos de uso

Dada la naturaleza del proyecto, la evaluación de los resultados se centra en la demostración práctica en RV, de la funcionalidad de estos componentes a través de ejemplos de uso concretos, utilizando como métrica de éxito el cumplimiento de los objetivos propuestos en cada módulo. Además, se realizó una estimación del ahorro en horas de programación, mediante una técnica simplificada, inspirada en la métrica de punto función (Albrecht, 1979), cuyos resultados se presentan al final de este capítulo.

Los ejemplos sirven para la evaluación de los módulos y además como guía didáctica o tutorial para aprender sobre su utilización. El código fuente se encuentra disponible al público en este repositorio: https://fmarinofiuba.github.io/vrComponentsKit/.

3.1 Evaluación del módulo de gestión de controladores

En el ejemplo controllersExample.html (figura 3 izquierda) se evaluó el correcto funcionamiento de módulo. Se verificó la correcta representación 3D de los controladores y el correcto funcionamiento de las funciones para escuchar eventos (imprime los eventos en la consola de navegador).

3.2 Evaluación del módulo de navegación

En el ejemplo navigationExample.html, (ver figura 3 derecha) se diseñó una escena que incluye un plano de suelo, con sus 4 cuadrantes en diferentes colores. Se verifico el correcto funcionamiento del modo "vuelo" (el usuario apunta el controlador y se traslada en dicha dirección). Se validó el mecanismo de teleportación utilizando los varios círculos blancos dibujados en el suelo para verificar que la nueva ubicación (luego de la traslación) efectivamente se encuentra en el centro de dichos círculos.

Por último, se utilizó un modelo 3D de una escalera con varios tramos ascendentes, para probar la teleportación en superficies a diferentes alturas (ver figura 3 derecha). Además, se utilizaron los cuadrantes coloreados para verificar que los cambios de orientación incremental se realizaron correctamente sin alterar la posición del usuario (la rotación debe ejecutarse sobre el eje vertical Y sin alterar la posición en la escena).

3.3 Evaluación del módulo UI

En el ejemplo UILVRMenuExample.html (ver figura 4) se diseñó una escena con dos objetos y una fuente de luz puntual. Se definió un menú (basado en la clase UILVR-Menu) que permite controlar las velocidades de rotación de ambos objetos, además del color y la intensidad de la fuente de luz. Este ejemplo permitió evaluar el correcto funcionamiento del menú en conjunto con el módulo de navegación. Cuando el usuario interactúa con el menú, el rayo saliente del controlador solo debe activar acciones sobre la UI y no ejecutar la acción de teleportación. El usuario puede mostrar/ocultar el menú mediante funciones de llamada vinculadas a los botones del controlador XR. Por último, en el ejemplo HtmlVRMenuExample.html, se realizaron las mismas evaluaciones, pero sobre un menú basado en HTML.

3.4 Evaluación del módulo de depuración

En el ejemplo debugToolsExample.html se creó una escena con un modelo 3D de una tetera que cambia de posición siguiendo una onda senoidal. En la escena se agregó un panel VrVarsWatcher que muestra en tiempo real los vectores posición, rotación y escala de la tetera. Además, se agregó un panel VrConsole y se generaron diferentes mensajes simulados a la consola del navegador, para verificar su correcta visualización en RV. Se pudo verificar que la conversión de HTML a mapa de texturas no muestra correctamente los tipos de datos complejos como objetos JavaScript.

3.5 Evaluación del de manipulación y selección de objetos

En el ejemplo grabableObjectsExample.html (figura 5) se diseñó una escena con un conjunto de objetos móviles (montados sobre una plataforma giratoria) y otro conjunto de objetos estáticos. El primer grupo fue configurado para ser sujetado por contacto, en cambio el según fue configurado para utilizar el modo "remoteGrabbing". El ejemplo permitió evaluar el correcto funcionamiento, verificando especialmente el caso en el

que múltiples objetos están en la línea del rayo del controlador, en cuyo caso solo el objeto más cercano debe ser sujetado.

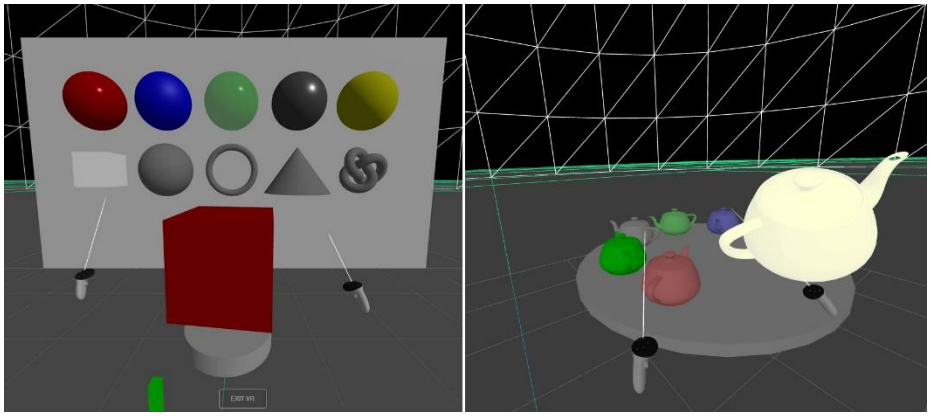

Fig. 5. Ejemplos de uso de SelectableVrObject (izquierda) GrabbableVrObject (derecha)

En el ejemplo selectableObjectsExample.html (figura 5) se creó una escena con 5 objetos que representan diferentes materiales, y 5 objetos que representan variadas formas geométricas. En el centro de la escena hay un pedestal cilíndrico y el usuario puede seleccionar un material o una forma que luego modificará el objeto central sobre el pedestal de forma acorde a la selección. Se verificó el objetivo propuesto.

3.6 Ahorro en horas de programación

Para estimar el impacto del uso de los componentes en términos de costo en horas, se analizaron cuáles son las tareas centrales que realiza cada una de las clases, clasificándolas según su complejidad en 3 niveles (baja: 4 puntos, media: 7 puntos y alta: 15 puntos). Luego, para cada módulo se totalizaron los puntos acumulados como "costo de implementación" (CI). Además, en base a los ejemplos de uso, se estimó el "costo de uso" de utilizar los módulos en una aplicación típica (medio en puntos de complejidad). Se definió un factor de conversión horas/puntos (FH) cuyo valor de estableció en 0.5 horas/puntos. Finalmente se calculó el ahorro (AH) con la siguiente formula:

$$AH=FH * (CI -CU)$$

Tabla 1. Estimación del ahorro en horas de programación

Módulo	CI (puntos)	CU (puntos)	AH (horas)
Gestión de controladores	135	8	63,5
UI	98	6	46
Navegación	58	6	26
Depuración	75	4	35,5
Selección y manipulación de objetos	83	6	38,5

4 Conclusiones y direcciones futuras

Frente el objetivo de facilitar el desarrollo de aplicaciones RV, la solución propuesta da una mayor flexibilidad para diseñar aplicaciones complejas, que otras soluciones pensadas para usuarios finales sin conocimientos de programación, aunque, el público objetivo son programadores con cierta experiencia en Three.js.

Los resultados de la tabla 1 evidencian un ahorro significativo en tiempo de programación y potencialmente en la reducción en las tasas de errores, ya que la organización en módulos permite probar los mecanismos en forma aislada del resto de la aplicación. Por último, el uso de nuevos tipos de eventos de alto nivel reduce el nivel de acoplamiento de la aplicación.

La diversidad de necesidades y características de las aplicaciones a desarrollar hace difícil y subjetiva la definición y justificación para crear componentes reutilizables, corriendo el riesgo de sobre diseñarlos. Solo la experiencia de desarrollar múltiples aplicaciones y la colaboración entre desarrolladores permite evaluar cuando ciertas abstracciones valen la pena. Uno de los desafíos futuros es consolidar las directrices para el diseño de interfaces 3D de usuario en RV (Yeo et al., 2024).

La depuración de aplicaciones RV sigue siendo un desafío, y las herramientas propuestas podrían mejorase significativamente, si se incluyeran mecanismos de entrada como teclados virtuales que permitan hacer cambios en tiempo de ejecución o el desarrollo de paneles con controles interactivos que permitan desplegar información más compleja, que variables simples.

También resulta imperativo realizar una evaluación extensiva de los módulos y sus componentes para analizar aspectos como la compatibilidad con dispositivos alternativos al Meta Quest 3.

Si bien el módulo UI propuesto se centra en una interfaz 2D para simplificar la implementación, futuras versiones podrían explorar la integración de elementos de UI 3D y diferentes modos de interacción (Kharoub et al., 2019).

Este proyecto pretende contribuir a la comunidad de software abierto con un conjunto de ejemplos de código que aceleraren el desarrollo de aplicaciones RV y que eventualmente sirvan para construir mejores componentes reusables que puedan integrarse en bibliotecas.

Bibliografía

(s.f.). Three.js: https://threejs.org/

Ahmed, K., & Andujar, C. (2022). Designing, testing and adapting navigation techniques for the immersive web. *Computers & Graphics, 106*, 66-76. https://doi.org/10.1016/j.cag.2022.05.015

Albrecht, A. (1979). Measuring application development productivity. New York: 83-92.

Berg, L. P., & Vance, J. M. (2016). Industry use of virtual reality in product design and manufacturing: a survey. *Virtual Reality, 21*(1). https://doi.org/10.1007/s10055-016-0293-9

Bose, D., & Brown, C. (2024). An Empirical Study on Current Practices and Challenges of Core AR/VR Developers. *2024 39th IEEE/ACM International Conference on Automated Software Engineering Workshops (ASEW)*, (págs. 233-238). Sacramento, CA, USA.

Chen, M.-X., Hu, H., Yao, R., Qiu , L., & Li, D. (2024). A Survey on the Design of Virtual Reality Interaction Interfaces. *Sensors, 24*(19), 6204. https://doi.org/10.3390/s24196204

Html2canvas. (s.f.). https://html2canvas.hertzen.com/

Johnson, G. (s.f.). *three-mesh-bvh*. https://github.com/gkjohnson/three-mesh-bvh

Kharoub, H., Lataifeh, M., & Ahmed, N. (2019). 3D User Interface Design and Usability for Immersive VR. *Applied Sciences, 9*(22), 4861. https://doi.org/10.3390/app9224861

lil-gui. (s.f.). https://github.com/georgealways/lil-gui

Meccawy, M. (2022). Creating an Immersive XR Learning Experience: A Roadmap for Educators. *Electronics*, 11(21), 3547. https://doi.org/https://doi.org/10.3390/electronics11213547

Meta. (s.f.). *Inmersive Web Emulator*. meta.com: https://developers.meta.com/horizon/blog/webxr-development-immersive-web-emulator/

Nebeling, M., & Speicher, M. (2018). The Trouble with Augmented Reality/Virtual Reality Authoring Tools. *2018 IEEE International Symposium on Mixed and Augmented Reality Adjunct (ISMAR-Adjunct)*, 333-337. https://doi.org/10.1109/ISMAR-Adjunct.2018.00098

Prithul, A., Adhanom, I., & Folmer, E. (2021). Teleportation in Virtual Reality; A Mini-Review. *Frontiers in Virtual Reality, 2*. https://doi.org/10.3389/frvir.2021.730792

Rubin, S. M., & Whited, T. (1980). A 3-dimensional representation for fast rendering of complex scenes. *SIGGRAPH Comput. Graph., 14*(3), 110-116. https://doi.org/10.1145/965105.807479

Starrett, M., McAvan, A. S., Huffman, D., Stokes, J. D., Kyle, C. T., Smuda, D. N., . . . Ekstrom, A. D. (2021). Landmarks: A solution for spatial navigation and memory experiments in virtual reality. *Behavior Research Methods, 53*. https://doi.org/10.3758/s13428-020-01481-6

W3C. (s.f.). *WebXR Device API*. https://www.w3.org/TR/webxr/

Yeo , A., Kwok, B., Joshna, A., Chen, K., & Lee, J. S. (2024). Entering the Next Dimension: A Review of 3D User Interfaces for Virtual Reality. *Electronics, 13*(3), 600. https://doi.org/10.3390/electronics13030600

Yigitbas, E., Klauke, J., Gottschalk, S., & Engels, G. (2022). End-user development for interactive web-based virtual reality scenes. *Journal of Computer Languages, 74*, 101187. https://doi.org/10.1016/j.cola.2022.101187

Yu, D., Dingler, T., Velloso, E., & Goncalves, J. (2024). Object Selection and Manipulation in VR Headsets: Research Challenges, Solutions, and Success Measurements. *ACM Comput. Surv., 57*(4), 34. https://doi.org/10.1145/3706417