Web Wide Worlds: A Modular Framework for Persistent, Interoperable Digital Environments

Dylan Z. Baker

Five Squared Interactive, LLC

Version 1.0, 28 December 2025

Table of Contents

1.0 Introduction

1.1 Abstract

The field of spatial computing is at a critical juncture. While technological innovation has enabled the creation of increasingly complex shared digital environments, the landscape is dominated by proprietary, siloed ecosystems. This fragmentation presents significant barriers to entry, forcing creators into platform-specific workflows and limiting the potential for a truly interoperable and open spatial web. The strategic imperative is clear: to develop web-native, open-source frameworks that empower creators of all skill levels, prioritize interoperability, and build upon the foundational principles that made the World Wide Web a transformative success.

Web Wide Worlds is a modular, open-source framework designed to make building 3D environments on the web accessible to creators across a range of technical backgrounds. Built atop declarative markup logic (VEML [1], X3D [2], and metaverse-extended GLTF [3]) and extensible UI shells, the platform enables drag-and-drop construction of spatial environments using familiar web technologies. This paper introduces the architecture and design principles behind Web Wide Worlds, with a focus on onboarding workflows, modular extensibility, and standards alignment. By integrating protocols such as HTTP and MQTT, and supporting formats like GLTF and USD [4], the framework facilitates scalable, synchronized environments that operate across devices and platforms. We present implementation details and a variety of use cases. The system demonstrates how declarative logic and modular design can reduce development overhead while supporting rich, persistent virtual spaces.

This whitepaper provides a comprehensive architectural and technical overview of the Web Wide Worlds framework for potential collaborators and partners. We will explore the system's core components, including its declarative markup, modular extensibility, and reliance on open standards, and demonstrate how they combine to form a robust foundation for the next generation of web-native worldbuilding.

1.2 The Challenge: Overcoming Fragmentation in Digital World Creation

The creation and deployment of interactive 3D environments has traditionally relied on powerful but proprietary game engines and specialized tooling. While these systems offer high-fidelity rendering and complex capabilities, they present substantial barriers to entry, requiring deep technical expertise and forcing creators into platform-specific workflows that hinder interoperability and accessibility.

The key barriers preventing the widespread, democratized creation of digital worlds can be summarized as follows:

- **Fragmented Ecosystem:** The current landscape is composed of siloed platforms and closed ecosystems. This forces creators to constantly reinvent core functionalities like synchronization and user presence for each new project. For consumers, this fragmentation creates artificial barriers to entry, preventing seamless movement and interaction between different digital experiences.
- **High Skill Level Required:** The development of interactive 3D environments is often gated by significant technical expertise in programming, 3D modeling, and engine-specific logic. This high barrier to entry limits the pool of creators and makes the development of dynamic, evolvable environments infeasible for many individuals, educators, and small teams.
- **Poor Scalability:** As the complexity of a digital world increases, development costs and timelines tend to skyrocket. This model of development is untenable for the vast majority of potential applications and creators.

These challenges highlight a clear market need for a new approach, one that prioritizes accessibility, interoperability, and natural scalability by leveraging the proven principles of the open web.

1.3 The Solution: The Web Wide Worlds Approach

Web Wide Worlds is a direct response to the challenges of fragmentation and complexity. It is positioned not as another proprietary engine, but as a foundational layer for building persistent, interoperable digital worlds using the tools and standards of the open web. The framework is inspired by the principles that made the World Wide Web a transformative, participatory medium.

The core principles of the Web Wide Worlds approach are to:

- Use existing web tools, technologies, and protocols.
- Augment, not replace, the web and established platforms, meeting users where they are.
- Leverage what users and devices already have, enabling worlds to run on billions of existing devices.
- Prioritize speed, accessibility, and natural scalability.
- Design modular and shareable systems that empower community-driven innovation.

This approach forms the architectural basis of the framework, which we will now explore in detail.

2.0 System Architecture: A Layered, Modular Design

Web Wide Worlds is organized as a modular stack comprising five interoperable layers. This strategic design promotes composability, allowing components to be used independently or in concert. It also ensures extensibility and platform-agnostic deployment, enabling the scalable construction of and interaction with persistent 3D environments across a diverse range of devices and platforms.

2.1 The Scene Viewer Layer

The Scene Viewer is the rendering interface through which users access and interact with virtual environments. To ensure maximum accessibility, software applications at this layer (referred to as World Browsers) are designed to abstract rendering complexity while providing a consistent experience across a multitude of platforms.

The reference implementation, WebVerse [5], is a direct implementation of the web's foundational principle of hyperlink-based access, ensuring frictionless entry in stark contrast to the app-store-gated models of closed platforms. Powered by the Straight Four World Engine, its key features include:

- **URL-Based Access:** Worlds can be accessed via a simple URL, eliminating installation barriers and enabling seamless entry, similar to clicking a link to a website.
- **Cross-Platform Rendering:** WebVerse supports WebGL/WebGPU for in-browser rendering and also provides platform-native Windows, Mac, Android, iOS, and Web runtimes for optimized performance.
- **High-Fidelity Clients:** The architecture supports full-featured clients with capabilities for raytracing, volumetric effects, and XR input, enabling high-quality immersive experiences.

The strategic function of this layer is to decouple the user experience from specific hardware, providing flexibility to support diverse input modalities from mouse and keyboard to touchscreen and full XR controllers/hands.

2.2 The World Definition Layer

The World Definition Layer specifies the structure, behavior, and logic of a virtual world. Its design enforces a clear separation of concerns between static scene definition and dynamic runtime behavior, empowering both designers and developers.

- **Declarative Markup (VEML, X3D, and OMI-extended GLTF):** The Virtual Environment Markup Language (VEML) [1] is an XML-based schema used to define the core blueprint of a world. VEML specifies entities, their spatial relationships (position, rotation, scale), references to external assets like GLTF models, and high-level control schemes. Its human-readable syntax allows designers to choreograph spatial logic without writing imperative code. The World Definition Layer also supports additional data formats, such as Extensible 3D (X3D) [2] and the Open Metaverse Interoperability Group (OMI)'s GL Transmission Format (glTF) extensions [3].
- **Modular Scripting:** Standard, client-side JavaScript is used to add dynamic and interactive behaviors. This includes defining triggers, complex interactions, ambient behaviors, and presence-based events. Developers can extend functionality using familiar web paradigms and a set of provided World APIs [6]. The JavaScript logic is executed inside of a JavaScript interpreter's isolated virtual machine, preventing injection-type attacks. Other programming languages can be securely supported as well, through the use of WebAssembly.

This dual-component architecture provides a powerful and accessible workflow. Declarative markup establishes the static scene and its core properties, while modular scripting brings it to life with dynamic, event-driven logic, creating a readable blueprint with a fully extensible runtime.

2.3 The Networking Layer

To provide maximum flexibility for diverse real-time evolving multiplayer architectures, the Networking Layer is implemented as a protocol-agnostic stack that enables synchronized presence and real-time interaction between multiple clients. It employs a layered set of open protocols to provide efficient communication.

Protocol Type	Example Protocols	Purpose
Transport Protocols	TCP, UDP, WebSocket, HTTP, REST	Provide foundational communication channels for data exchange.
Messaging Protocols	MQTT, AMQP, DDS	Enables lightweight, efficient publish/subscribe messaging for real-time updates.
Synchronization	WorldSync, Mirror, Photon	Manages real-time state replication, object ownership, and interpolation.

The primary lightweight engine, WorldSync [7], handles most real-time state replication needs. For use cases requiring robust, deterministic simulation, the framework supports the optional integration of Mirror Networking for authoritative server logic and rollback capabilities. This layered approach allows the framework to support both peer-to-peer and server-authoritative configurations. Additional synchronization protocols can be adapted/developed into this networking stack.

2.4 The Tooling & APIs Layer

The Tooling & APIs Layer provides the extensibility and developer-facing interfaces necessary for customization, integration, and content creation. The tooling is intentionally modular and non-prescriptive, empowering creators to adopt workflows suited to their specific goals.

Key components of this layer include:

- **WorldOS:** A NodeJS-based back-end framework that serves as a scaffold for persistent server-side extensions. It invokes modular, reusable apps on startup or in response to events and uses a software bus for inter-app communication. Apps can be written in Python, NodeJS, Unity, Godot, Blender, or any other language or ecosystem.
- **Open API Ecosystem:** A suite of developer interfaces including RESTful endpoints, WebSocket streams, and SDKs for integration with external systems and services.
- **Development Tools:** A collection of creator-facing tools, including live editors, prefab inspectors, spatial debuggers, and both command-line and browser-native deployment interfaces.

This layer empowers developers to move seamlessly from rapid prototyping to studio-scale integration by providing a robust but adaptable set of tools that augment, rather than dictate, their creative process.

2.5 The Publishing Layer

The Publishing Layer governs the distribution and hosting of worlds, offering flexible ownership models that support a wide range of use cases. Creators can choose the deployment model that best fits their needs for control, scalability, and accessibility.

Self-Hosted Deployment	Third-Party Hosted Services
Grants full control over infrastructure.	Provides cloud-hosted environments, such as WorldHub [8].
Ideal for private simulations, experimental forks, and custom integration pipelines.	Includes features like point-and-click tooling, discoverability, versioning, and user management.

This flexibility supports diverse ownership models and allows creators to publish both persistent, long-lived worlds and ephemeral, temporary experiences according to their needs.

3.0 Core Technologies and Implementation

Moving from the high-level architecture, we now examine the core technologies that power the Web Wide Worlds framework. The implementation combines declarative markup, client-side scripting, and lightweight networking to deliver a scalable and accessible system for creating 3D worlds.

3.1 World Definition with Declarative Markup and JavaScript

The cornerstone of world definition is the Virtual Environment Markup Language (VEML) [1]. As an XML-based schema, VEML provides a human-readable syntax for specifying the spatial layout of a world. A VEML document defines entities and their properties—such as position, rotation, and scale—and includes tags for meshes, terrains, UI elements, and locomotion controls. It also manages references to external assets, such as GLTF models, textures, and audio files, via simple URLs.

These VEML definitions are parsed and executed by a client-side JavaScript runtime. This runtime exposes a set of World APIs [6] that enable scripts to dynamically interact with the scene. These APIs facilitate dynamic entity manipulation (movement, animation), event-

driven interaction (proximity triggers, click handlers), and updates to the scene based on user input or device type. This allows developers to extend a world's functionality with custom behaviors using standard JavaScript modules.

World definition can also be accomplished using other file formats. The Open Metaverse Interoperability (OMI) Group's glTF extensions [3] support similar types of entities as VEML. The Extensible 3D (X3D) [2] XML schema also serves a similar purpose. These formats can be supported by a world browser, with conversions between these formats even being possible.

3.2 Real-Time Synchronization

Real-time state replication is managed by WorldSync [7], a lightweight engine designed to abstract the complexities of networking. Built on WebSocket and MQTT, WorldSync enables multi-user presence with minimal configuration. Synchronization is implemented declaratively; a creator simply adds a synchronize attribute to an entity's definition in the VEML file.

Once enabled, WorldSync automatically handles several critical tasks:

- Replication of an entity's state (e.g., position, rotation) across all connected clients.
- Resolution of object ownership and authority.
- Management of users/clients in a session.

For use cases requiring more robust, server-authoritative simulation, such as competitive multiplayer games, the framework supports the optional integration of Mirror Networking [9]. This provides support for deterministic rollback and authoritative logic, offering developers greater control over the simulation state.

3.3 Server-Side Logic and Deployment with WorldOS

WorldOS [10] is the modular, NodeJS-based server framework used to deploy and manage persistent environments. It provides the back-end infrastructure needed for dynamic, long-lived worlds by invoking reusable apps and facilitating their communication via a software bus.

Key capabilities of WorldOS include:

- Asynchronous messaging via MQTT for inter-app communication.
- Support for the procedural generation and propagation of terrains, structures, and other dynamic assets.

- Persistent state storage for long-lived sessions and user-generated content.

WorldOS is designed to integrate with standard asset pipelines via formats like GLTF and USD. It supports both self-hosted deployment on standard web infrastructure and cloud-based options, providing a flexible back end for everything from small-scale experiments to planet-scale simulations.

3.4 User Access and Interaction

WebVerse [5] serves as the primary user-facing client, or "World Browser." Its most critical feature is its URL-based entry mechanism, which allows users to join a world with a simple click, eliminating the need for downloads or installations. WebVerse adapts its rendering pipeline and input mapping to a wide array of devices and modalities.

Complementing WebVerse is WorldHub [8], a companion web service that provides a suite of tools for creators and users. Its primary functions include:

- Providing hosting and deployment tools for creators.
- A search engine for world discovery, allowing users to find and share worlds.
- Point-and-click editors that enable low-code world creation for non-technical users.

4.0 Applications and Use Cases

The framework's modular architecture and declarative logic are designed to support a diverse range of applications. By providing a flexible, web-native foundation, Web Wide Worlds enables creators to build experiences tailored to specific goals while maintaining interoperability and scalability.

Interactive Web Content

Web Wide Worlds is ideally suited for creating spatial microsites that can enhance or replace traditional static web pages. The lightweight embedding of the Scene Viewer and the declarative nature of VEML facilitate the rapid creation of product demos, interactive concept boards, and explorable UX prototypes. These 3D scenes can be embedded directly into blogs, marketing materials, or educational platforms to drive user engagement through motion, sound, and reactive scripting.

Game Development and Simulation

The framework provides a robust foundation for both casual and structured game development. The combination of WorldSync for lightweight presence and optional Mirror Networking for authoritative simulation supports a wide range of multiplayer game

architectures. The open APIs allow developers to integrate external systems for inventory, scoring, or narrative branching, while flexible hosting options support various release strategies and audience targets.

Extension of Existing Worlds and Games

Existing Worlds from games such as Minecraft and Cities: Skylines can be ingested into a Web Wide Worlds server. This enables new capabilities and forms of interactions with these worlds that are not possible in the original game (such as flying an aircraft in Minecraft or driving through a Cities: Skylines city).

Educational Simulations

The declarative markup and prefab components of Web Wide Worlds empower educators and instructional designers to construct explorable learning environments. The VEML schema is designed so that non-programmers can "scaffold complex environments without imperative code," allowing them to build interactive simulations and guided walkthroughs to visualize complex scientific concepts, historical events, or spatial relationships, making abstract ideas tangible and engaging.

Collaborative Prototyping

WorldSync enables synchronous co-creation, allowing distributed teams to collaborate in real time. This workflow is perfectly suited for prototyping game mechanics, architectural layouts, and user interaction models. Critically, the framework's URL-based access allows teams to share and iterate on drafts with a simple link—a workflow impossible in traditional engine-based pipelines that require full project builds and installations for each collaborator.

5.0 Conclusion

Web Wide Worlds presents a modular, declarative, and open framework engineered to lower the barrier to entry for spatial computing. By abstracting complex logic into extensible markup and leveraging familiar web standards, the platform empowers a wider range of creators to build, share, and inhabit persistent, explorable environments. Its layered architecture provides a scalable foundation for applications spanning from interactive web content and educational simulations to game development and collaborative prototyping.

Web Wide Worlds is not merely a toolset, it is a proposition for how the web itself might evolve: toward a participatory, standards-driven ecosystem where creators of all skill levels can build, share, and inhabit digital worlds. We invite partners and collaborators to join us in building this open and evolvable foundation for the future of spatial computing.

Works Cited

[1] Five Squared Interactive, LLC, "Virtual Environment Markup Language (VEML) GitHub," [Online]. Available: https://github.com/Five-Squared-Interactive/VEML.

[2] Web3D Consortium, "What is X3D," [Online]. Available: https://www.web3d.org/x3d/what-x3d/.

[3] Open Metaverse Interoperability Group, "OMI GLTF Extensions GitHub Page," [Online]. Available: https://github.com/omigroup/gltf-extensions.

[4] Pixar Animation Studios, "USD Home," [Online]. Available: https://openusd.org/release/index.html.

[5] Five Squared Interactive, LLC, "WebVerse - Explore Worlds Like Websites," [Online]. Available: https://webverse.fivesqd.com/.

[6] Five Squared Interactive, LLC, "World APIs GitHub Documentation," [Online]. Available: https://github.com/Five-Squared-Interactive/WebWideWorlds/blob/main/docs/JS_World_APIs.md.

[7] Five Squared Interactive, LLC, "WorldSync," [Online]. Available: https://fivesqd.com/worldsync.

[8] Five Squared Interactive, LLC, "WorldHub," [Online]. Available: https://www.worldhub.me/.

[9] Mirror Networking, "Mirror Networking," [Online]. Available: https://mirror-networking.com/.

[10] Five Squared Interactive, LLC, "WorldOS GitHub Page," [Online]. Available: https://github.com/Five-Squared-Interactive/WorldOS/tree/main.