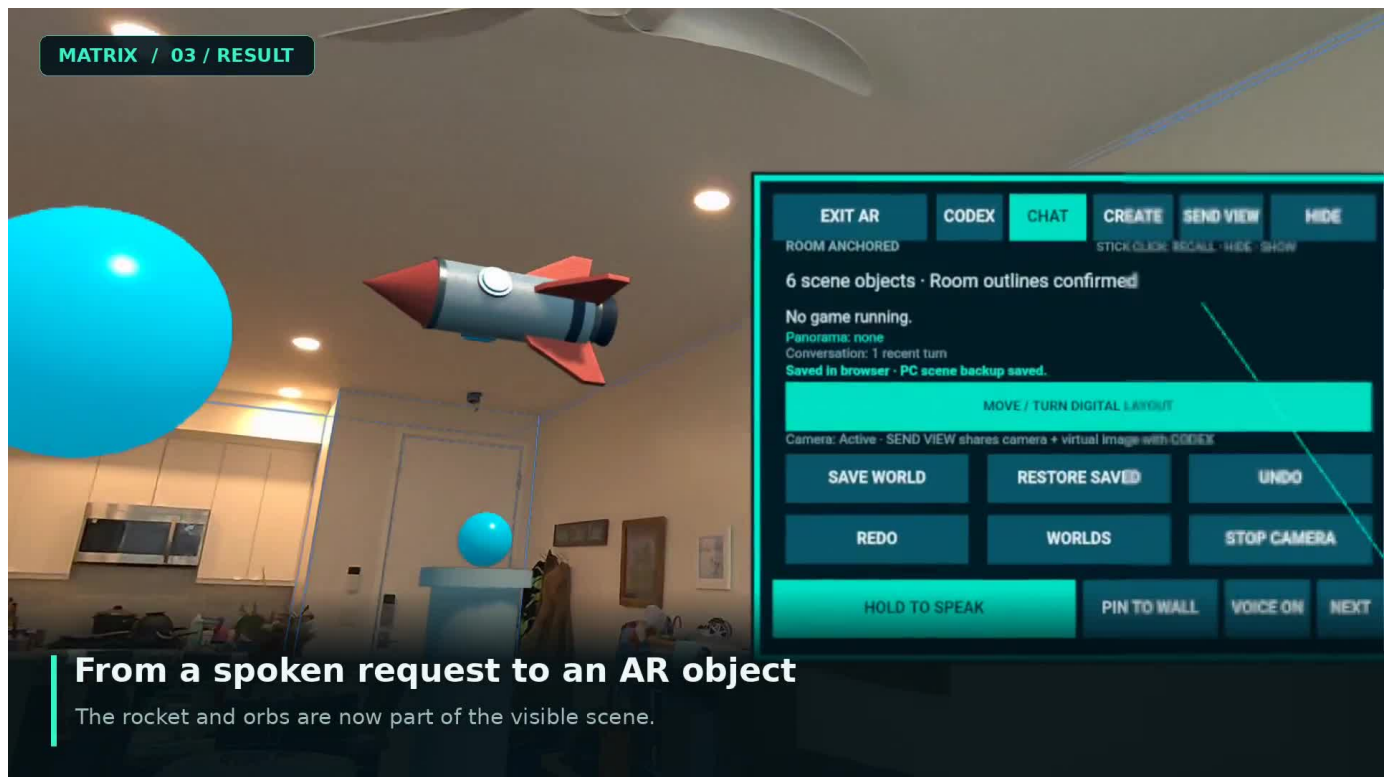


PROJECT 1 / INTERACTIVE AUGMENTED REALITY

Matrix Loading Operator

Voice-driven creation in room-aware AR and VR

Matthew Topham | CSCI/ATLS 4616, CU Boulder
Project 1 Demo Day | 29 September 2026



From a spoken request to an AR object

The rocket and orbs are now part of the visible scene.

Figure 1. Actual Quest 3 footage from the submitted demo (02:10). A custom rocket and procedural orbs appear over the physical room beside the in-world Operator interface. This is a headset screenshot, not a concept rendering. [22]

1. Introduction and motivation

Matrix Loading Operator lets a headset wearer ask an AI to create or change 3D content without leaving the experience. Quest Browser uses Three.js and WebXR to render editable objects over the physical room; a connected PC runs Codex and the authoring tools.

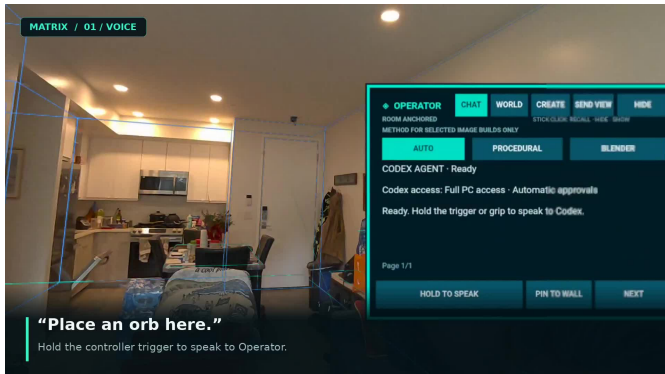
Typical asset workflows require switching between catalogs, desktop modeling tools, and the headset. Matrix puts the request, inspection, and interaction in one spatial interface.

Project 1 contribution

Create or reuse content, manipulate it in passthrough AR, and save the world. Quest 3 supplies the mobile experience; a connected PC supplies the AI and authoring tools.

2. What the submitted demo shows

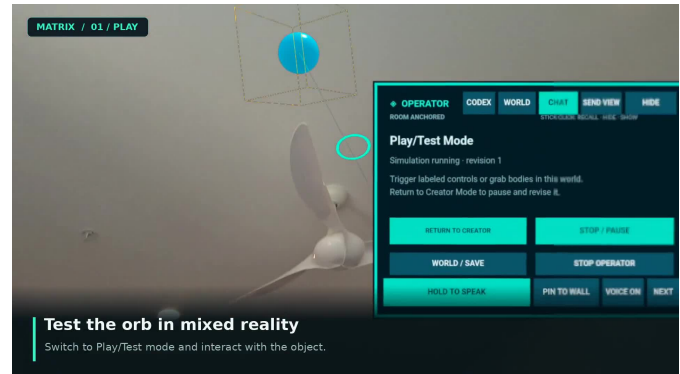
The 2 minute 34 second video follows a room-scale creation session. Waiting is shortened in the edit; the video does not measure generation speed. [22]



01 / ASK

Say "Place an orb here." The Operator receives the request in the visible room.

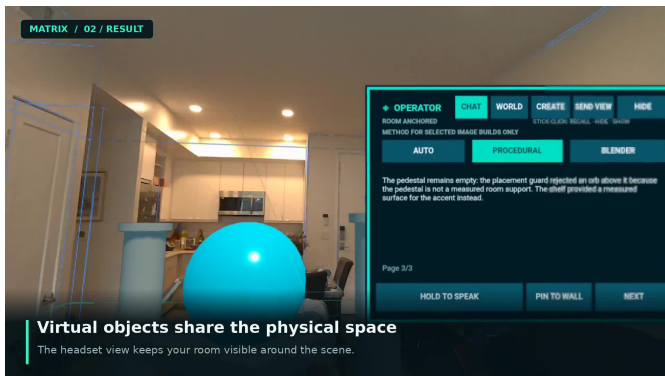
Figure 2. Submitted video, 00:16.



02 / INTERACT

Switch to Play/Test mode with the created orb visible in passthrough AR.

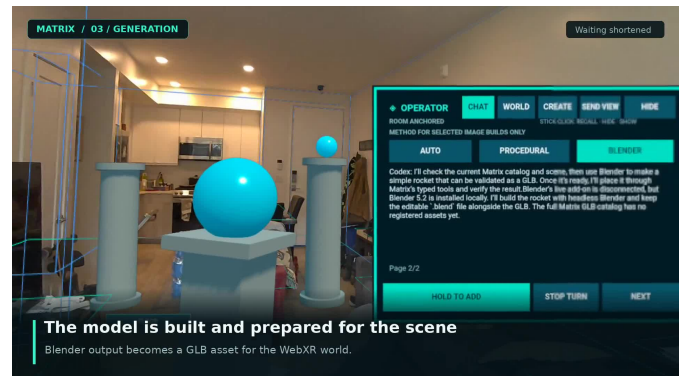
Figure 3. Submitted video, 00:43.



03 / BUILD THE ROOM

Procedural columns and a pedestal add a small virtual display to the physical space.

Figure 4. Submitted video, 01:15.



04 / AUTHOR A CUSTOM ASSET

Request a basic rocket in Blender; the Agent prepares a GLB for the world.

Figure 5. Submitted video, 01:40.

From the request to the final rocket

The Agent checks the scene and catalog, authors a model, exports and registers the GLB, and places an instance in Matrix. The rocket appears beside the orbs in the final shot (Figure 1).

The visible Agent reply reports that the live Blender add-on was disconnected. For this request, it used the locally installed Blender through headless scripting and retained editable .blend source alongside the GLB. The demo therefore shows a Blender-authored asset path, with headless scripting used for the rocket.

Separate frozen v1.0 tests cover selected-image bridge creation, measured tabletop placement, and save/reopen. Those tests are described in Section 8 and are not all shown in the film. [20, 22]

3. Design process and related work

3.1 From a content library to a writable world

The original Matrix idea was an AI interface over a large catalog of environments, objects, characters, animations, scripts, voices, and sound effects. A user would ask the Loading Operator for content and the system would load it into a virtual world. The earlier Unity prototype explored this catalog-and-prefab approach.

The current prototype keeps asset reuse and adds creation when the catalog is missing an object. Procedural recipes provide fast structured geometry; Blender provides custom assets. Validated outputs can enter the catalog for later reuse. The broader idea of a parallel Boulder inhabited by AI citizens remains the destination, while Project 1 develops the room-scale creation interface.

Design iteration	Resulting decision
Catalog-based Unity prototype	Keep reusable assets and stable object identity.
Three.js / WebXR creator runtime	Use one browser world for desktop, VR, and AR presentation.
Visual concept + creation tools	Treat a selected image as art direction; a tool produces the actual geometry.
Room-aware interaction	Refresh measured spatial context before edits and confirm visible alignment.

3.2 Related products, technical work, and inspiration

Reference	Relationship to Matrix
ShapesXR [5]	Spatial design and prototyping are a useful nearby example. Matrix explores a conversational, tool-driven creation workflow.
Meta MR Utility Kit [4]	Room meshes and semantic surfaces informed the earlier Unity exploration. The current browser uses WebXR observations rather than running Unity MRUK.
WebXR + Three.js [1-3]	Implementation foundations for immersive sessions, tracked inputs, real-world hit tests, and rendering.
ComfyUI + Blender [6, 7]	Image ideation and 3D authoring tools connected through PC workflows; their outputs serve different roles.
Generative Agents [8]	Memory, reflection, and planning motivate the future resident layer. Project 1 does not evaluate an AI civilization.

The Matrix supplies the loading-program metaphor; *Ready Player One* suggests large shared worlds and content libraries; *Snow Crash* suggests technically empowered early creators. These are design inspirations rather than technical evidence. [9-11]

4. System architecture

The ordinary creator experience is the browser-owned /web/ world. A PC service connects it to the AI and authoring tools. [12-14]

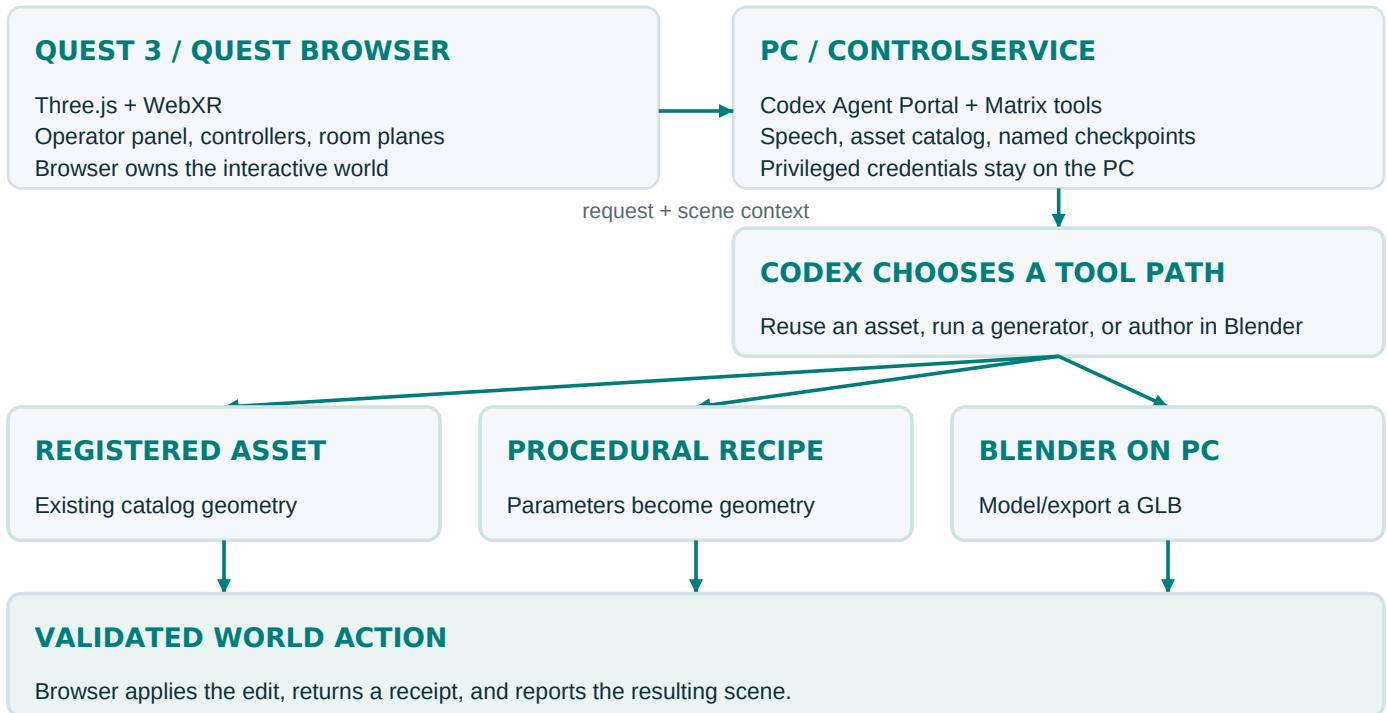


Figure 6. The browser renders and owns the ordinary /web/ world. The PC Agent operates creation tools. Results converge on the same entity model and receipt checks. The diagram omits optional image jobs and shows logical flow rather than individual HTTP endpoints.

4.1 Responsibilities and request flow

WebRuntime/ owns the scene, rendering, controller interaction, room observations, and supported physics. The in-world Operator is a user interface in that scene. **ControlService/** serves the client and owns the Agent session, speech/image jobs, catalog validation, action queues, and named checkpoints.

A spoken request is transcribed on the PC and sent to a persistent Codex conversation. The Agent receives current runtime capabilities plus selected-object and pointing context when available. It chooses tools that the connected runtime can support. Codex and configured tool credentials remain on the PC; the headset receives Matrix session and result information. [13]

4.2 Confirming an edit

Each world change passes through a typed Matrix action that checks the active room, scene revision, target identity, and bounded parameters. The browser executes the accepted change and returns a receipt. The service then checks a fresh scene report for the expected object and transform. A model reply alone does not establish that the world changed.

PC configuration selects reviewed or automatic Agent mode. The recorded demo displays automatic approvals. Runtime validation and receipts still apply; changing this configuration requires restarting the service. [13, 22]

5. Content, interaction, and persistence

5.1 Three routes to scene content

Route	What happens
Reuse	Look up an existing registered asset and place a new instance with its own object ID.
Procedural	Choose a supported recipe and parameters. The runtime constructs editable geometry without waiting for a custom Blender export.
Blender	Author on the PC through configured Blender MCP or local scripting. Export a self-contained GLB, validate/register it, and place it through Matrix tools.

For GLBs, registration checks file identity, embedded resources, size, geometry budgets, and bounds before copying the asset into an immutable catalog. The Agent can use `matrix_register_glb`, then `matrix_spawn_asset` and `matrix_spawn_status`. Registration makes content reusable; spawning creates a world instance. A successful export by itself does not place an object. [14, 15]

The optional image path first generates variants and records an explicit selection. A later build request attaches the selected image bytes to the Agent. In the frozen v1.0 test, two ComfyUI images guided a procedural bridge. The image establishes visual intent; it supplies neither a mesh nor physical room measurements. [13, 18, 20]

5.2 Direct manipulation and Play/Test

In Creator mode, holding a trigger on an editable object grabs it. Controller motion changes its pose; the grabbing hand's thumbstick translates it, with stick-click selecting height. The other thumbstick rotates it. Release commits the transform to the same ID.

Play/Test uses supported Rapier rigid-body simulation. A released dynamic body can carry recent movement and spin into the simulation, letting a creator edit and test in the same world. [14]

5.3 What saving preserves

Stored world state	Reconstruction and limits
Entity IDs, assets, transforms	Rebuild the same composition; referenced catalog assets must still be available.
Environment and supported behavior	Restore configured world content and supported attached behavior.
Dynamic-body state	Whole-world saves retain supported pose, velocity, and sleep state. Solver contacts are transient; exact replay is not guaranteed.
Room-origin provenance	Use the saved room relationship when locatable; require renewed alignment or explicit recovery when it is unavailable.

Browser Save World stores supported world state locally. A named PC checkpoint provides a separate restore path. Scene-only legacy saves have a narrower contract; saving content is not the same as proving that its physical alignment will be recovered in every room. [14, 17]

6. How room-aware AR works

Passthrough shows the physical room. Measured WebXR surfaces provide the geometry for supported room-constrained placement. [1, 2, 16, 17]

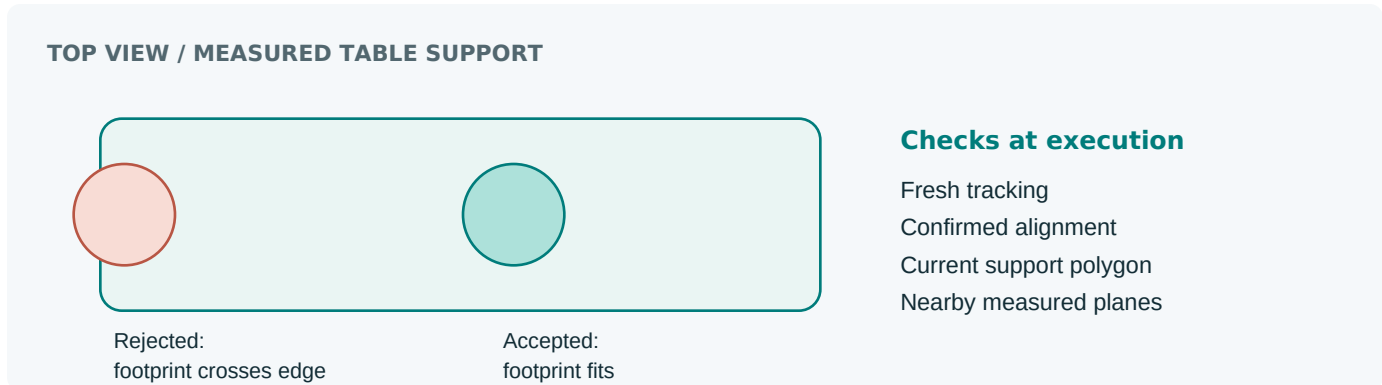


Figure 7. Schematic support-fit test. A point inside a table is insufficient if the entire object footprint does not fit. The separate v1.0 orb test used a 25 cm orb. Diagram is illustrative and not to scale. [16, 17, 20]

6.1 Alignment and measured placement

On entering AR, the browser observes room planes, hit tests, pose, tracking freshness, and the origin. The wearer checks visible outlines against the room and confirms alignment before measured edits. A new session or changed origin requires fresh confirmation.

For `matrix_spawn_on_surface`, the Agent supplies a current support ID, support-local pose, and room token. At execution, the browser refreshes the polygon and checks the footprint, nearby measured planes, alignment, and tracking. An edge location can fail despite being inside the table. [16, 17]

An accepted pose is converted to canonical world coordinates before saving, keeping its ID and transform independent of temporary plane IDs. Later moves require fresh room context. The frozen test confirmed edge rejection, a clear placement, a same-ID move, and save/reload persistence. [17, 20]

6.2 What the AI can infer from each source

Context source	Use and boundary
Room planes / hit tests	Metric placement against currently measured geometry. Unmeasured objects may remain absent from these observations.
Selected object / pointing hit	Identifies the intended virtual object or real support when a current target is available.
Optional environment camera	Qualitative context such as visible furniture. The shared physical frame and virtual render are separate, labeled panels.

Camera sharing requires opt-in and Send View. It supplies neither calibrated depth nor metric camera pose or pixel alignment. Stop Camera restores virtual-only sharing. Images are transient Agent context, not scene-save data. [19]

7. Reproducing the prototype

Windows example: frozen v1.0.0 source and Quest 3. Run the text-driven creator loop, then enable optional providers. [12-15, 20]

Requirement	Purpose
Python 3.10+, Git, Node 22.12+/npm	Run ControlService and build the Three.js client. [23]
Native Codex CLI, ChatGPT login	Run the PC Agent Portal; the launcher checks the existing sign-in.
Quest 3 + Quest Browser	Run head-mounted AR/VR. The validation used Browser 152 on Android 14.
USB + Android platform tools	Use ADB reverse for the local headset route; a trusted HTTPS origin is an alternative.

7.1 Build and start the PC service

Clone the repository [12] and run these commands from its root. Use the same Python environment for Matrix MCP and ControlService. The launcher defaults to reviewed mode.

```
git checkout v1.0.0
python -m pip install -r ControlService/requirements-agent-mcp.txt
Set-Location WebRuntime
npm.cmd ci
npm.cmd run build
Set-Location ..
codex login status
.\Start-CodexControlService.ps1 -Port 8765
```

Run `codex login` first if needed. Alternatively, extract the prebuilt release ZIP after comparing its SHA-256 with the published SHA256SUMS file. It includes the built browser client. [20]

7.2 Connect the headset

Enable Quest developer mode, connect USB, and accept USB debugging. In a second terminal, map the port, then open the URL in Quest Browser. Close other Matrix runtimes; only one creator lease is active.

```
adb reverse tcp:8765 tcp:8765
http://127.0.0.1:8765/web/
```

7.3 Enable the paths used by the demo

Voice: run `.\Setup-LocalSpeech.ps1` to install the English speech model, then restart the service. **Blender:** install it on the PC and make its executable available for Agent scripting, or configure a connected Blender MCP instance. Export a self-contained GLB and use Matrix's registration/spawn path. [14, 15]

Optional image ideation: configure ComfyUI with a reviewed graph using PC-side `MATRIX_CONTENT_CONFIG`. Direct object requests do not need it. The release omits local models, private state, and credentials. [13, 20]

8. Evaluation and practical limits

Evaluation combines the submitted video with documented frozen v1.0 acceptance: receipts, scene checks, and wearer observations. These are recorded results, not a new benchmark or user study. [20, 22]

Capability	Evidence and result
Orb, procedural room display, rocket	Quest footage shows virtual content over the room; the rocket uses headless Blender. [22]
Selected-image creation	Two ComfyUI concepts, explicit selection, and a procedural bridge with ID/receipt in the frozen run. [18, 20]
Controller manipulation	Frozen bridge grab/move in VR, plus separate documented Creator/Play/Test interaction checks. [14, 20]
Measured AR spawn and move	Separate orb world rejects an edge point, accepts a clear point, and moves the same ID. [16, 17, 20]
Save and reopen	Frozen bridge/world and final orb state were saved/reopened. AR required renewed alignment. [20]
Optional camera context	Frame received; Stop Camera restored virtual-only. Permission denial was not separately wearer-tested. [19, 20]

8.1 Release-level validation

The stable package was built from commit a7664ae. The release record reports **893 ControlService tests**, **703 WebRuntime tests**, and **11 release-builder tests**, plus a production Vite build, two byte-identical isolated builds, verification of all **1,304 ZIP entries**, and an extracted-service smoke test. Headset acceptance used Quest 3 / Browser 152 / Android 14. [20]

8.2 Limitations observed in this scope

Latency and setup. Modeling and tool calls can interrupt the experience. The edited video shortens waiting, and no prompt-to-object latency distribution was recorded. The headset still depends on a configured PC and external authoring tools.

Spatial completeness. Fresh planes and valid bounds can establish fit against measured geometry, but they cannot prove clearance from everything in the room. Wearer observation is still needed. Cross-session registration is conditional on locating the saved origin and confirming alignment.

Evaluation coverage. The frozen bridge and orb used separate worlds. The later rocket video does not establish frozen-package acceptance. Test counts do not measure usability, comfort, or novelty.

8.3 A compact repeatable acceptance sequence

Enter AR, wait for usable room context, confirm outlines, and select a clear support point. Request an orb; verify its receipt, ID, transform, and visible fit. Move it to a second clear point, then Save World. Exit/reopen, re-enter AR, confirm alignment, and compare the restored ID and transform. For custom content, repeat the registration/spawn sequence with a GLB and verify that the saved world references the same asset. [14, 17, 20]

9. Future work: toward the larger Matrix

Project 1 builds the creator layer. The longer-term vision connects persistent digital places, AI inhabitants, and a wearable interface to the physical world.

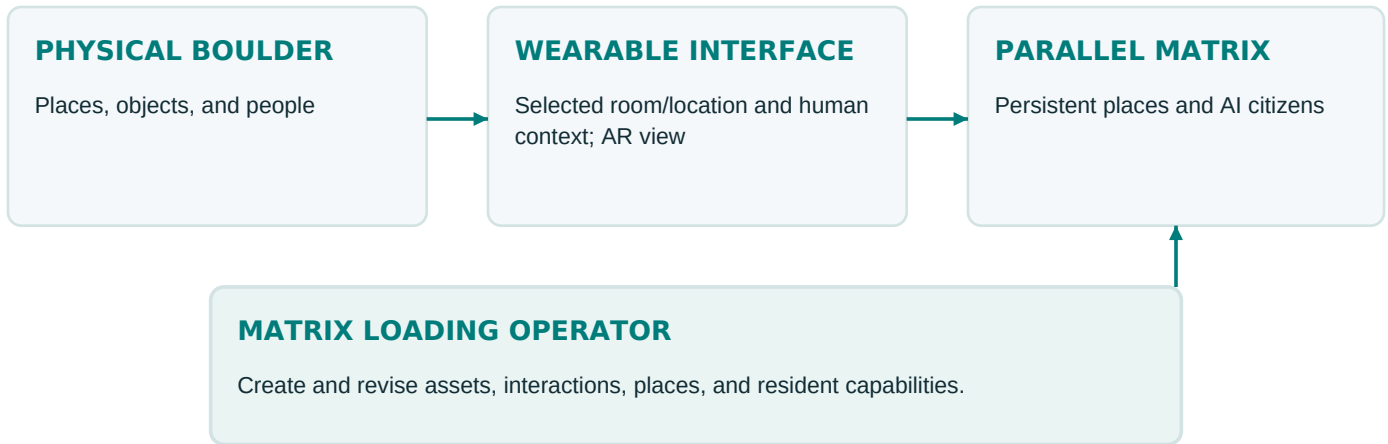


Figure 8. Longer-term concept. Loading Operator is the creation layer; the persistent city, wearable synchronization, and full AI civilization remain future work. Arrows indicate intended relationships, not completed integrations.

9.1 Make the room-scale creator loop easier

The v1.1 follow-up tracker marks real-room rearrangement, active-turn steering, collapsible browser controls, and point targeting complete. Further work should improve responsiveness, refine object locking, and make whole-world AR placement easier. Repeated furniture/layout trials would test these improvements together. [21]

9.2 Build a persistent digital Boulder

The original vision imagined a virtual Boulder in Unity; the current runtime is Three.js/WebXR. Growing from rooms to streets and neighborhoods requires shared spatial identity, multiplayer, persistent world ownership, permissions, and reliable cross-location registration. Users could enter in VR or reveal corresponding digital content over real places in AR.

9.3 Add AI citizens through the same world capabilities

Residents would retain memories, goals, relationships, and schedules, use reachable objects, and form shared institutions. Small repository experiments are a starting point. The development order is world foundations, interactions, residents, then larger simulations. Residents would request bounded changes through world capabilities without receiving Operator or PC credentials. [8, 12]

9.4 Connect the two worlds through a wearable counterpart

A future Manfred-style wearable interface could share selected location, activity, and personal context with a digital counterpart, with user permission. Spatial glasses would reveal the digital layer in everyday use; Loading Operator would build and revise it. Continuous wearable synchronization and city-scale AR remain future work.

References and project sources

Repository references use v1.0.0 where applicable. Figures 1-5 are frames from the submitted video; Figures 6-8 are explanatory diagrams. Links are clickable.

- [1] W3C. WebXR Device API. w3.org/TR/webxr
- [2] W3C. WebXR Hit Test Module. w3.org/TR/webxr-hit-test-1
- [3] Three.js. WebXRManager documentation. threejs.org / WebXRManager
- [4] Meta. MR Utility Kit basic sample. [Meta Horizon / MRUK basic sample](#)
- [5] ShapesXR. Product overview and design tools. shapexr.com/product
- [6] Comfy-Org. ComfyUI. github.com/Comfy-Org/ComfyUI
- [7] Blender Foundation. glTF export Python API. docs.blender.org / Export Scene Operators
- [8] J. S. Park, J. C. O'Brien, C. J. Cai, M. R. Morris, P. Liang, and M. S. Bernstein. Generative Agents: Interactive Simulacra of Human Behavior. UIST 2023. arxiv.org/abs/2304.03442
- [9] Neal Stephenson. Snow Crash. 1992. Fictional design inspiration.
- [10] The Wachowskis. The Matrix. 1999. Fictional design inspiration.
- [11] Ernest Cline. Ready Player One. 2011. Fictional design inspiration.
- [12] Matt Topham. Matrix Loading Operator source repository. github.com/School-of-the-Ancients/matrix-loading-operator
- [13] Matrix v1.0.0. Agent Portal gateway. [Frozen source / AGENT_PORTAL.md](#)
- [14] Matrix v1.0.0. WebRuntime guide, controls, persistence, and limits. [Frozen source / README.md](#)
- [15] Matrix v1.0.0. Blender authoring tiers. [Frozen source / BLENDER_AUTHORING_TIERS.md](#)
- [16] Matrix v1.0.0. Room-aware Operator validation, 28 September 2026. [Frozen source / Room-Aware-Operator-149-2026-09-28.md](#)
- [17] Matrix v1.0.0. Durable AR surface placement validation, 28 September 2026. [Frozen source / Durable-AR-Surface-149-2026-09-28.md](#)
- [18] Matrix v1.0.0. Selected ComfyUI concept validation, 27 September 2026. [Frozen source / Concept-91-Comfy-Selected-2026-09-27.md](#)
- [19] Matrix v1.0.0. Quest camera-context validation, 28 September 2026. [Frozen source / Quest-Camera-Context-26-2026-09-28.md](#)
- [20] Matrix WebXR v1.0.0 release. Frozen commit a7664ae; package and integrated Quest acceptance record. [GitHub / v1.0.0 release](#)
- [21] Matrix v1.1 follow-up milestone. Issue 167. [GitHub / issue 167](#)
- [22] Matt Topham. Matrix Demo Presentation. Submitted MP4, 29 September 2026, 2:34. Frames in this report use video timestamps.
- [23] Vite. Migration to Vite 7 and Node.js support. v7.vite.dev/guide/migration

Development credits

Project author: Matthew Topham. AI-assisted development credits retained from the original submission: GPT-6 Sol and GPT-6 Astra. The report describes the implemented prototype and distinguishes recorded demonstrations, release validation, and future concepts.