


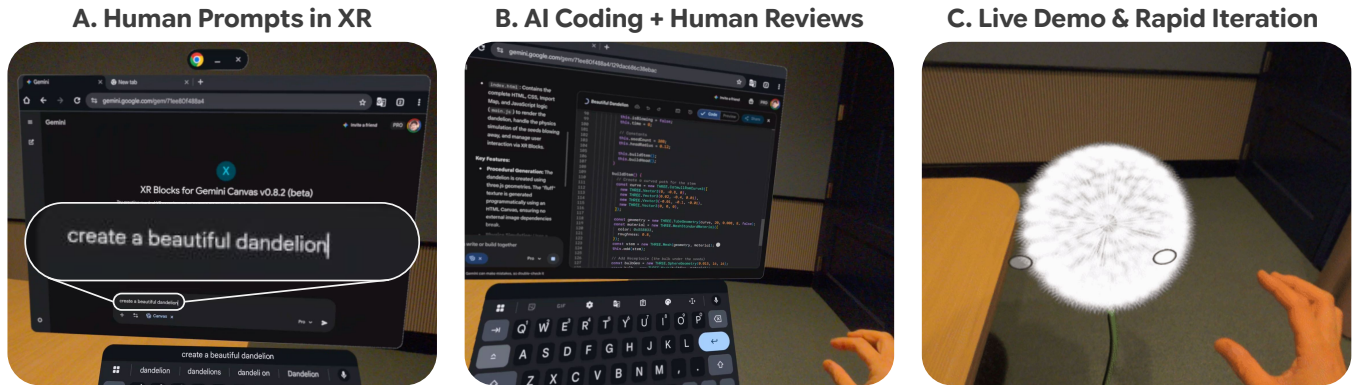
# Vibe Coding XR: Accelerating AI + XR Prototyping with XR Blocks and Gemini

Ruofei Du<sup>\*‡</sup>, Benjamin Hersh, David Li<sup>\*</sup>, Nels Numan<sup>†</sup>, Xun Qian<sup>†</sup>, Yanhe Chen<sup>†</sup>, Zhongyi Zhou,<sup>†</sup>  
Jiahao Ren, Xingyue Chen, Robert Timothy Bettridge, Faraz Faruqi, Xiang ‘Anthony’ Chen,  
Steve Toh, David Kim

 <https://xrblocks.github.io/gem>

 <https://github.com/google/xrblocks>

Google XR Labs



**Figure 1:** Example user journey of VIBE CODING XR, an end-to-end workflow to create immersive AI + XR experiences through vibe coding: (A) User types “create a beautiful dandelion” with XR BLOCKS Gem (<http://xrblocks.github.io/gem>) on a Galaxy XR headset in a Chrome browser. (B) Gemini turns user intent into an interactive XR application within a minute, while user can browse and review the thought and vibe coding process. (C) User selects the “Enter XR” button and instantly see an animated dandelion blows away upon pinch.

## ABSTRACT

While large language models have accelerated software development through “vibe coding”, prototyping intelligent Extended Reality (XR) experiences remains inaccessible due to the friction of complex game engines and low-level sensor integration. To bridge this gap, we contribute XR Blocks, an open-source, modular WebXR framework that abstracts spatial computing complexities into high-level, human-centered primitives. Building upon this foundation, we present VIBE CODING XR, an end-to-end rapid prototyping workflow that leverages LLMs to translate natural language intent directly into functional XR software. Using a web-based

interface, creators can transform high-level prompts (e.g., “create a dandelion that reacts to hand”) into interactive WebXR applications in under a minute. We provide a preliminary technical evaluation on a pilot dataset (VCXR60) alongside diverse application scenarios highlighting mixed-reality realism, multi-modal interaction, and generative AI integrations. By democratizing spatial software creation, this work empowers practitioners to bypass low-level hurdles and rapidly move from “idea to reality.” Code and live demos are available at <https://github.com/google/xrblocks> and <http://xrblocks.github.io/gem>.

## CCS CONCEPTS

• **Human-centered computing** → **Mixed / augmented reality**;  
**Natural language interfaces.**

## KEYWORDS

vibe coding, extended reality, XR Blocks, prototyping, AI, XR, GenXR

## 1 INTRODUCTION

Recent advances in Large Language Models (LLMs) [8, 26, 40, 45] and agentic workflows [11, 22, 25] are fundamentally reshaping software engineering and creative computing. We are witnessing the emergence and growing prevalence of “vibe coding” [18], a

\* Both authors contribute equally to XR Blocks.

† Equal contributions, sorted alphabetically.

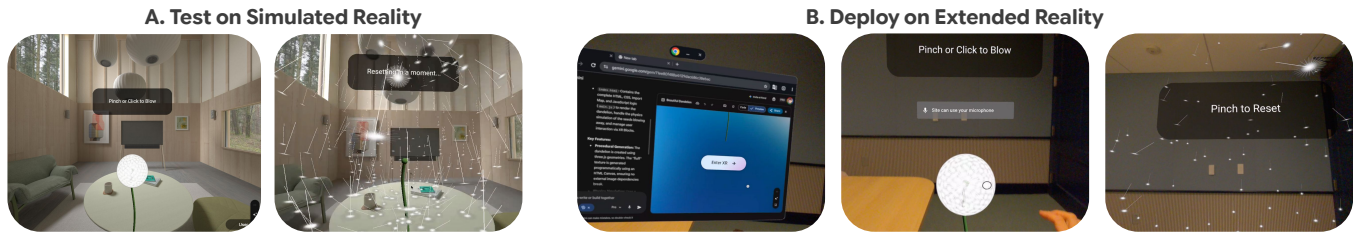
‡ Corresponding author: Ruofei Du, me [at] duruofei [dot] com.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from [permissions@acm.org](mailto:permissions@acm.org).  
*arXiv, 2026, Vibe Coding XR*

© 2026 Association for Computing Machinery.

ACM ISBN 978-1-4503-XXXX-X/2025/09...\$15.00

<https://doi.org/10.1145/nnnnnnn.nnnnnnn>



**Figure 2: VIBE CODING XR accelerates AI + XR prototyping by allowing users to (A) test their “vibe coding” results on desktop in a “simulated reality” environment, and (B) deploy the same demo on an Android XR headset with body and hand interactions.**

paradigm in which high-level human intent is translated directly into functional software. While tools such as Gemini Canvas [23], Antigravity [25], Cursor [11], and Claude Code [3] have successfully democratized this capability for 2D and even 3D web development [17, 46], the domain of Extended Reality (XR) remains largely inaccessible. Prototyping intelligent spatial experiences usually requires navigating a fragmented ecosystem of perception pipelines [27, 38, 50], low-level sensor API integrations [19, 34], complex game engines [42, 44], and on-device testing. Crucially, the absence of “vibe coding” in XR stems from a fundamental lack of high-level building blocks at the disposal of LLMs. Current models struggle to generate reliable spatial software because they are forced to reason over low-level, disjointed API plumbing rather than semantic, human-centric spatial concepts.

To bridge this gap, we introduce XR BLOCKS [33], an open-source WebXR framework designed specifically to make spatial computing intelligible to both human creators and generative AI. As the foundational contribution of this work, XR BLOCKS provides these missing high-level semantic abstractions. It introduces a modular “Reality Model” that treats users, the physical environment, and intelligent agents as first-class, configurable primitives. By absorbing the incidental complexity of sensor fusion and pipeline integration, XR BLOCKS equips LLMs with a concise, robust vocabulary to generate complex spatial behaviors without hallucinating low-level implementation details.

Building directly upon this enabling framework, we present VIBE CODING XR as our second key contribution: an end-to-end rapid prototyping workflow that leverages the long-context reasoning capabilities of Gemini to act as an expert XR designer. Using a specialized system prompt, XR BLOCKS templates, and a web-based interface (XR BLOCKS Gem), this workflow translates natural language prompts directly into functional, physics-aware XR applications in under a minute. Creators can rapidly iterate on these generated scripts within a desktop “simulated reality” environment, or prompt directly within an Android XR [21] headset to test their creations with live hand interaction and environmental sensing.

In summary, we contribute:

- **The XR BLOCKS framework**, an open-source, modular WebXR architecture that provides the missing high-level building blocks, abstracting low-level perception and interaction pipelines into semantic primitives optimized for AI + XR prototyping.
- **VIBE CODING XR**, a web-based, open-access workflow that leverages LLMs and the XR BLOCKS framework to translate

natural language intent into deployable, physics-aware XR applications.

- **Evaluation and scenarios** of VIBE CODING XR using our VCXR60 dataset, alongside diverse application scenarios demonstrating immersive education, physics-aware interaction, and rapid game prototyping.

## 2 RELATED WORK

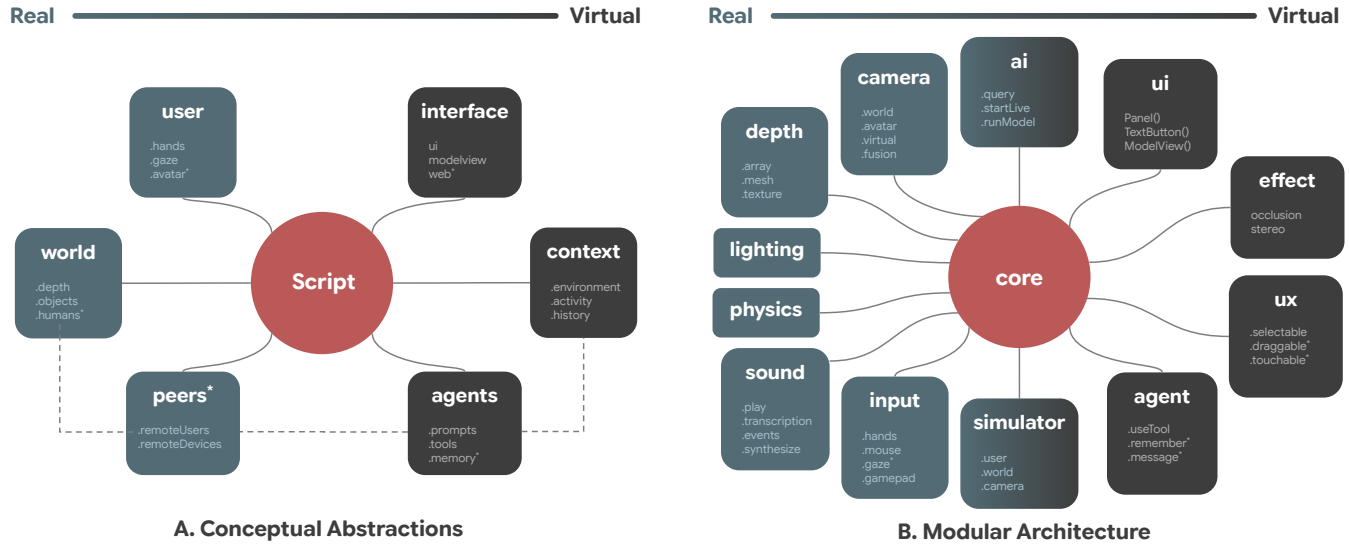
VIBE CODING XR bridges the gap between generative AI workflows and spatial computing by enabling intent-driven XR prototyping.

### 2.1 GenXR: Creating AI + XR Experiences

The democratization of XR development has evolved from early toolkits like VR Juggler [6] and ARToolKit [31] to comprehensive game engines such as Unity [42] and Unreal [44]. While these native engines offer high ceilings for fidelity, they impose steep learning curves, significant friction for rapid prototyping for XR, and high barriers for sharing executable code. Conversely, WebGL libraries like `three.js` [41] and frameworks like A-Frame [2] provide cross-platform accessibility but lack the high-level abstractions necessary for complex XR interaction and seamless AI integration. XR BLOCKS bridges this divide. By offering a high-level, web-optimized interaction model akin to MRTK [4], VRTK [47], XRI [43] for Unity, our framework abstracts low-level sensor and perception pipelines so creators can focus on the *what* of an experience rather than the *how*. Crucially, as a web-native architecture, XR BLOCKS makes AI + XR prototyping inherently accessible and reproducible, empowering the community to seamlessly share, fork, and build upon each other’s creations. VIBE CODING XR further leverages this framework to create behavioral AI + XR experiences, with photorealistic 3D content creation [29, 37, 39, 48] remaining out of scope.

### 2.2 Vibe Coding: Gaps from AI to XR

The AI community benefits from a “flywheel effect” driven by open ecosystems such as Hugging Face [30] and TensorFlow Hub [24], benchmarks like LMArena [10], and composable frameworks including JAX [7], PyTorch [35], and TensorFlow [1]. XR, however, lacks a comparable substrate for rapid, intent-driven iteration. Recent efforts have begun to address this by leveraging LLMs to generate spatial objects and scenes in Unity, such as LLMR [12], DreamCodeVR [20], and Thing2Reality [29]. Furthermore, recent systems like DreamGarden [17] have demonstrated using LLM-driven hierarchical planning to generate functional 3D game environments, assets, and C++ logic in Unreal Engine.



**Figure 3: Design of the XR Blocks Framework: (A) conceptual abstraction of the “Reality Model”, and (B) modular architecture of the “core” engine. Subsystems marked with \* have not yet been fully open sourced.**

While these approaches successfully push the boundaries of automated game design and 3D scene composition, they often rely on asynchronous compilation workflows within heavy, closed-engine ecosystems. This hinders the collective “vibe coding” paradigm [18], in which an agent’s utility is fundamentally tied to its ability to synthesize and share knowledge across the entire relevant domain. To this end, we are motivated to develop a spatial computing platform on the open web inspired by [14]. By wrapping LLMs [49] around the XR BLOCKS framework, VIBE CODING XR empowers creators to generate immersive, intelligent, and physics-aware interactions in real-time. This web-based approach ensures that every contribution is (1) built upon a transparent, web-standard codebase, (2) designed for rapid iteration, and (3) instantly distributable across diverse devices and users. By prioritizing these pillars, we transition toward an ecosystem where the community does not merely consume agents but actively drives their evolution, fueling a true “flywheel effect” for XR development.

### 3 VIBE CODING XR

We introduce VIBE CODING XR, a rapid prototyping workflow that combines the generative reasoning capabilities of LLMs with a high-level modular XR framework. This synergy allows creators to bypass the friction of low-level systems programming and focus purely on sculpting spatial behaviors, AI logic, and interactions.

#### 3.1 XR Blocks Framework

At the foundation of our system is XR BLOCKS, an open-source modular SDK designed to accelerate human-centered AI + XR innovation. Built upon accessible web technologies like WebXR [5], three.js [41], and TensorFlow.js [1], the framework introduces a high-level *Reality Model*. This unified abstraction treats the user, the physical world, and intelligent agents as first-class, configurable primitives rather than disjoint data streams. The following snippets

demonstrate XR BLOCKS’s design principle to keep things simple: it creates an object at user’s eye level with preferred distance to object, which changes color when user performs a hand pinch or desktop click.

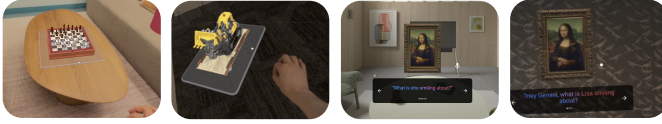
```
class MainScript extends xb.Script {
  init() {
    // geometry and lighting setup is omitted.
    this.player = new THREE.Mesh(geometry, material);
    this.player.position.set(0, xb.user.height - 0.2,
      ↪ -xb.user.objectDistance);
    this.add(this.player);
  }
  onSelectEnd(event) {
    this.player.material.color.set(Math.random() * 0xfffff);
  }
}
```

As shown in Figure 3, the framework’s *core* engine manages the complex interplay of subsystems required for spatial computing. This includes environmental perception (e.g., depth sensing, lighting estimation), XR interaction (e.g., hand rays, pinch gestures, rigid-body physics), and AI integration (e.g., LLM querying, agent memory). By abstracting these technical hurdles, XR BLOCKS lowers the barrier to entry while retaining a high ceiling for expressivity, positioning it as the ideal foundation for VIBE CODING XR.

The framework’s architecture was iteratively refined through the development of 34 open-source templates (examples: Figure 4), incorporating feedback from weekly workshops and technical demonstrations.

#### 3.2 VIBE CODING XR Workflow

VIBE CODING XR leverages the long-context capabilities and advanced thinking processes of modern LLMs like Gemini [40] to function as an expert XR designer and engineer. We developed a specialized system prompt in Appendix C that “teaches” the LLM

**A. XR Fundamentals:** models, simulator, reticle, spatial UI...**B. XR Realism:** depth sensing, scene understanding, lighting estimation...**C. XR Interaction:** hands, gestures, touch, ...**D. AI + XR Integration:** camera, object understanding, agent, live conversation...

**Figure 4: Human-coded templates and samples in the XR Blocks framework provide the foundational best practices and API grounding for VIBE CODING XR.**

the XR BLOCKS architecture through an open-source web interface<sup>1</sup>. This context includes:

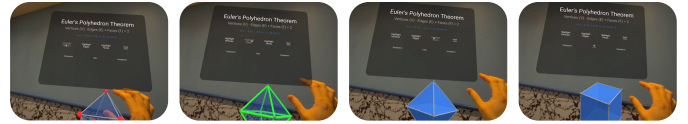
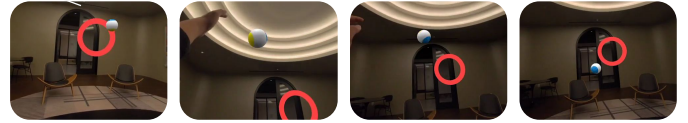
- **Persona & Guidelines:** Establishes the LLM as a domain expert adhering to best practices for room-scale XR environments (e.g., spatial layout, human-scale proportions, and interaction distances).
- **Package Management:** Specifies how dependencies within XR Blocks are handled to enforce recommended default styles.
- **Source Code & Templates:** Grounds the model with the full source code of the core `xrblocks.js` library, alongside curated templates and samples (illustrated in Figure 4). This strict grounding minimizes API hallucination and encourages adherence to established design patterns.

Users interact with VIBE CODING XR through high-level natural language prompts (or “vibes”). The LLM translates this intent directly into functional XR BLOCKS scripts [49], which can be previewed immediately in a desktop simulated reality environment or deployed directly to an Android XR headset.

### 3.3 Application Scenarios: From Prompt to XR

To demonstrate the versatility of the VIBE CODING XR workflow, we present several prototypes generated via VIBE CODING XR during four 1-hour workshops with 20 participants. These examples (with prompts provided in Appendix B and the supplementary video) highlight the system’s ability to seamlessly integrate the core pillars of the XR BLOCKS framework: AI + XR capabilities, XR Interaction, and XR Realism.

<sup>1</sup>XR Blocks Gem: <http://xrblocks.github.io/gem>; XR Blocks Prompt: <http://xrblocks.github.io/prompts>

**AS1. Visualizing Euler’s Theorem in Geometry****AS2. Physics Lab****AS3. Immersive Chemistry****AS4. XR Sports**

**Figure 5: AI-generated application for educational and exercise use cases. See Appendix B for full prompts.**

**Educational AI + XR Experiences.** As shown in Figure 5, VIBE CODING XR excels at rapidly prototyping interactive learning environments. Prompted with “visualize Euler’s theorem in geometry and explain vertices, edges, and facets” (AS1), it generated an immersive *Math Tutor* application. The LLM autonomously selected geometries (a tetrahedron, cube, and octahedron) and allowed users to pinch them to trigger different highlighting strategies. Similarly, prompting for an *Immersive Chemistry* (AS3) resulted in an app featuring educational cards and 3D volumetric visual effects, allowing users to safely observe simulated reactions (e.g., igniting methane or ethylene) via hand-pinch gestures.

**Physics-Aware XR Realism & Interaction.** By grounding the LLM in XR BLOCKS’s physics and depth modules, users can generate highly tactile experiences. Prompting for an *XR Sports* (AS4) application (“Let me play volleyball with hands and collide with my environment”) yielded a textured ball that reacts realistically to both the user’s physical hands and the surrounding room geometry. Furthermore, a *Physics Lab* prompt successfully implemented a mixed-reality scale where users pick and drop labeled weights to intuitively learn mechanics.

**Rapid Game Prototyping & Procedural Generation.** This open-source workflow significantly reduces the time required to prototype XR games. Prompted to “create the Chrome Dino game in XR,” the system generated a voxelized version of the classic game—complete with rushing cacti on a semi-transparent lane and audio cues—reducing development time from hours to minutes. Through iterative refinement, creators have also used the system for advanced procedural generation, including rendering immersive NASA starmaps, procedural city layouts, and explorable ancient Egyptian pyramids directly in the headset.

**Table 1: Inference time and one-shot success rate for XR Blocks Gem with various Gemini backends on VCXR-60 in 5 runs. We used “preview” models in the evaluation.**

XR Blocks Gem		Time (secs)		Pass Rate@1
Gemini Model	Thinking	Median	IQR	Percentage
gemini-3.1-pro	High	86.02	32.12	<b>95.5%</b>
gemini-3.1-pro	Low	33.39	8.60	94.1%
gemini-3-flash	High	22.26	4.66	87.8%
gemini-3-flash	Low	<b>17.30</b>	4.00	87.4%

## 4 PRELIMINARY TECHNICAL EVALUATION

Evaluating XR applications poses unique challenges, as it typically requires manual, on-device testing and subjective human evaluation. However, this manual process struggles to keep pace with the rapid iteration cycles of foundational models. To assess the effectiveness of our pipeline at scale, we constructed a preliminary benchmark dataset, **VCXR60**, to automate testing.

Sourced from four one-hour workshops, VCXR60 consists of 60 XR application prompts provided by 20 participants (detailed in Appendix A). Using this dataset together with the XR BLOCKS desktop simulator, we measured inference time and the one-shot pass rate (pass@1). Inspired by code LLM benchmarks like HumanEval [9], we define a “pass” as zero-error execution via unit test. In 3D graphics, the rendering engine is an inherent unit test engine to leverage, and thus we monitor runtime error logs in a headless Chromium browser via Playwright [36] as our unit test.

Early analysis revealed that the majority of initial generation errors stemmed from edge-case bugs within the XR BLOCKS framework itself (before v0.11.0) or hallucinated APIs by the LLMs, initially yielding an approximate 70% pass rate. These insights fueled a rapid, six-month iteration cycle. After 11 major releases, we evaluated XR BLOCKS v0.11.0 to gather the baseline results shown in Table 1.

Averaged across five runs, our evaluation revealed distinct performance trade-offs based on model size, “thinking” levels, and prompt complexity. Notably, every test prompt achieved at least one successful execution across the five runs. For simpler interactions (e.g., “Create a beautiful dandelion that blows away when I pick it up”), faster models like Gemini Flash often completed generation in under 20 seconds. However, for applications requiring complex animation states and precise hand-interaction logic (e.g., Prompt 060 in Table 1), these models exhibited a higher rate of runtime errors compared to more advanced models like Gemini Pro.

Our primary takeaway for researchers and practitioners is clear: larger models are essential for authoring XR applications that intricately weave perception with interaction. Enabling a “high” thinking mode yields the most reliable results for advanced XR prototyping. By affording the LLM more tokens to explicitly plan its thought process, this workflow more successfully navigates the complex spatial logic, physics integrations, and perception pipelines required for XR, drastically improving the one-shot success rate and minimizing debugging friction for the user.

## 5 LIMITATION AND DISCUSSION

This paper presents our initial step towards “idea to reality”. Our current implementation prioritizes the core abstractions for XR, and its limitations highlight several exciting avenues for future research:

**Performance and Latency:** Our choice of web technologies prioritizes accessibility but brings known trade-offs. The framework can never match the rendering performance of native engines like Unity or Unreal, and reliance on cloud AI models introduces network latency. Our visionary goal is to develop an *LLM-driven cross-compiler* capable of translating a high-level XR BLOCKS script into optimized, native code across multiple target engines like GoDot; we invite the whole community to contribute the SDK for common practice of AI + XR interactions to further reduce LLM token costs.

**Multi-Sensory Synthesis:** We have focused on audio-visual experiences, but XR BLOCKS is designed to be extensible. Integrating modules for haptics, EEG, and other sensory modalities could allow creators to compose truly immersive, multi-sensory narratives. E.g., one may extend XR BLOCKS with Arduino through WebUSB protocol.

**Large-scale Benchmarking and Agentic Evaluation:** As an initial step, VCXR-60 is far from covering the full spectrum of AI + XR. For instance, a rigorous evaluation should decouple contributions of XR BLOCKS from three.js, and quantitatively evaluate how many tokens XR BLOCKS could save with the abstractions. While we are committed to developing a more rigorous benchmark, we would like to evolve XR BLOCKS together with the next generation of LLMs by open-sourcing with the community. Future work will look into agentic testing with human-in-the-loop and eventually lead to inventing better interaction paradigms for AI + XR.

## 6 CONCLUSION

We present VIBE CODING XR, an end-to-end workflow that democratizes spatial computing by translating high-level creative intent into functional XR prototypes. By coupling the generative reasoning of LLMs with the XR BLOCKS framework, we demonstrated a novel approach that collapses the distance between a fleeting thought and a tangible, physics-aware reality. This shift toward spatial “vibe coding” empowers a new generation of creators to shape the 3D web, transforming passive consumers into active architects of their digital and physical environments.

However, this human-AI symbiosis remains in its infancy. The stochastic nature of current LLMs still introduces friction, occasionally generating code that misinterprets physical constraints, syntax, or complex interaction logic. Furthermore, XR BLOCKS currently lacks ready-to-use accessibility modules. To mature this domain, the field must move beyond ad-hoc demonstrations toward rigorous, *formal benchmarking*. Just as ImageNet [13] accelerated progress in computer vision, we envision large-scale, open-source datasets of *prompt-to-interaction* pairs. Such benchmarks will enable the systematic evaluation of generated spatial behaviors through human raters, expert review, and agentic workflows. Hybrid approaches that blend natural language prompting with spatial UIs will also be critical for refining XR creation.

As an evolving ecosystem, future iterations of XR BLOCKS must expand its generative vocabulary to support improved *aesthetics* and

richer *multimodal inputs*. This will enable users to guide generation not only through text and voice, but also via gaze, micro-gestures, and cross-device interactions. Concurrently, responsible real-world deployment necessitates rigorous *human-in-the-loop evaluation* to better understand how designers co-create with AI agents, ensuring these tools safely augment human intent rather than introduce harmful noise.

Ultimately, we open-source this framework as an open invitation. We welcome the HCI, XR, and AI communities to build upon this workflow, fill the existing gaps in tooling primitives and expressivity, and collectively realize a future where creating an interactive, mixed-reality world is as natural as describing a dream.

## ACKNOWLEDGMENTS

We thank all of XR Blocks contributors in 2025, including David Li, Nels Numan, Xun Qian, Yanhe Chen, Zhongyi Zhou, Evgenii Alekseev, Geonsun Lee, Alex Cooper, Min Xia, Scott Chung, Jeremy Nelson, Xiuxiu Yuan, Jolice Dias, Tim Bettridge, Benjamin Hersh, Michelle Huynh, Konrad Piascik, Ricardo Cabello, David Kim, and Ruofei Du. We further thank Xiuxiu Yuan, Oren Haskins, Brian Collins, Barak Moshe, Seeyam Qiu, and Coco Fatus for the UX Design Support; Yinghua Yang and Brenton Simpson for the cross-team collaboration; Tim Bettridge, Kevin Lam for significant contributions to XR Blocks x Gemini Canvas; Furthermore, we would like to extend thanks to the seminal work in Android XR Unity Samples with Mahdi Tayarani, Max Dzitsiuk, Patrick Hackett, all of DepthLab [16], Ad hoc UI [15], Rapsai (Visual Blocks) [14], Instruct-Pipe [49], DialogueLab [28], and Sensible Agent [32] co-authors, which greatly inspired this project along the way; we further extend thanks to Jim Ratcliffe, Eric Gonzalez, Nicolás Peña Moreno, Max Spear, Yi-Fei Li, Ziyi Liu, Jing Jin, Sean Fanello, Thabo Beeler, Adarsh Kowdle, and Guru Somadder for their insightful feedback and support in XR Blocks.

## REFERENCES

- [1] 2015. TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems. <https://www.tensorflow.org/> Software available from tensorflow.org.
- [2] A-Frame Authors. 2025. A-Frame. <https://aframe.io>.
- [3] Anthropic. 2026. Claude Code. <https://code.claude.com/docs/en/desktop>
- [4] Mixed Reality Toolkit Authors. 2025. MRTK3. <https://github.com/MixedRealityToolkit/MixedRealityToolkit-Unity>
- [5] WebXR authors. 2022. WebXR. <https://immersiveweb.dev/>
- [6] Allen Bierbaum, Christopher Just, Patrick Hartling, Kevin Meinert, Albert Baker, and Carolina Cruz-Neira. 2001. VR Juggler: A Virtual Platform for Virtual Reality Application Development. In *Proceedings IEEE Virtual Reality 2001*. 89–96. doi:10.1109/VR.2001.913774
- [7] James Bradbury, Roy Frostig, Peter Hawkins, Matthew James Johnson, Chris Leary, Dougal Maclaurin, George Neulac, Adam Paszke, Jake VanderPlas, Skye Wanderman-Milne, and Qiao Zhang. 2018. JAX: Composable Transformations of Python+NumPy Programs. <http://github.com/jax-ml/jax>
- [8] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language Models Are Few-Shot Learners. *Advances in Neural Information Processing Systems* 33 (2020), 1877–1901. doi:10.48550/arXiv.2005.14165
- [9] Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, Alex Ray, Raul Puri, Gretchen Krueger, Michael Petrov, Heidy Khlaaf, Girish Sastry, Pamela Mishkin, Brooke Chan, Scott Gray, Nick Ryder, Mikhail Pavlov, Alethea Power, Lukasz Kaiser, Mohammad Bavarian, Clemens Winter, Philippe Tillet, Felipe Petroski Such, Dave Cummings, Matthias Plappert, Fotios Chantzis, Elizabeth Barnes, Ariel Herbert-Voss, William Hebgen Guss, Alex Nichol, Alex Paino, Nikolas Tezak, Jie Tang, Igor Babuschkin, Suchir Balaji, Shantanu Jain, William Saunders, Christopher Hesse, Andrew N. Carr, Jan Leike, Josh Achiam, Vedant Misra, Evan Morikawa, Alec Radford, Matthew Knight, Miles Brundage, Mira Murati, Katie Mayer, Peter Welinder, Bob McGrew, Dario Amodei, Sam McCandlish, Ilya Sutskever, and Wojciech Zaremba. 2021. Evaluating Large Language Models Trained on Code. arXiv:2107.03374 [cs.LG] <https://arxiv.org/abs/2107.03374>
- [10] Wei-Lin Chiang, Lianmin Zheng, Ying Sheng, Anastasios Nikolas Angelopoulos, Tianle Li, Dacheng Li, Hao Zhang, Banghua Zhu, Michael Jordan, Joseph E. Gonzalez, and Ion Stoica. 2024. Chatbot Arena: An Open Platform for Evaluating LLMs by Human Preference. arXiv:2403.04132 [cs.AI]
- [11] Cursor. 5. Cursor - the AI Code Editor. <https://cursor.com>.
- [12] Fernanda De La Torre, CathyMengying Fang, Han Huang, Andrzej Banburski-Fahey, Judith Amores Fernandez, and Jaron Lanier. 2024. LLMR: Real-Time Prompting of Interactive Worlds Using Large Language Models. In *Proceedings of the CHI Conference on Human Factors in Computing Systems*. ACM. doi:10.1145/3613904.3642579
- [13] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. 2009. ImageNet: A Large-Scale Hierarchical Image Database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*. IEEE. doi:10.1109/CVPR.2009.5206848
- [14] Ruofei Du, Na Li, Jing Jin, Michelle Carney, Scott Miles, Maria Kleiner, Xiuxiu Yuan, Yinda Zhang, Anuva Kulkarni, XingyuBruce Liu, Ahmed Sabie, Sergio Orts-Escolano, Abhishek Kar, Ping Yu, Ram Iyengar, Adarsh Kowdle, and Alex Olwal. 2023. Rapsai: Accelerating Machine Learning Prototyping of Multimedia Applications Through Visual Programming. In *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems (CHI)*. ACM, 1–23. doi:10.1145/3544548.3581338
- [15] Ruofei Du, Alex Olwal, MathieuLe Goc, Shengzhi Wu, Danhang Tang, Yinda Zhang, Jun Zhang, DavidJoseph Tan, Federico Tombari, and David Kim. 2022. Opportunistic Interfaces for Augmented Reality: Transforming Everyday Objects Into Tangible 6DoF Interfaces Using Ad Hoc UI. In *Extended Abstracts of the 2022 CHI Conference on Human Factors in Computing Systems (CHI, 183)*. ACM, 1–4. doi:10.1145/3491101.3519911
- [16] Ruofei Du, Eric Turner, Maksym Dzitsiuk, Luca Prasso, Ivo Duarte, Jason Dourgarian, Joao Afonso, Jose Pascoal, Josh Gladstone, Nuno Cruces, Shahram Izadi, Adarsh Kowdle, Konstantine Tsotsos, and David Kim. 2020. DepthLab: Real-Time 3D Interaction With Depth Maps for Mobile Augmented Reality. In *Proceedings of the 33rd Annual ACM Symposium on User Interface Software and Technology (UIST)*. ACM, 829–843. doi:10.1145/3379337.3415881
- [17] Sam Earle, Samyak Parajuli, and Andrzej Banburski-Fahey. 2025. DreamGarden: A Designer Assistant for Growing Games From a Single Prompt. In *Proceedings of the 2025 CHI Conference on Human Factors in Computing Systems*. ACM. doi:10.1145/3706598.3714233
- [18] Benj Edwards. 2025. Will the Future of Software Development Run on Vibes. <https://arstechnica.com/ai/2025/03/is-vibe-coding-with-ai-gnarly-or-reckless-maybe-some-of-both>.
- [19] Cathy Fang, Yang Zhang, Matthew Dworman, and Chris Harrison. 2020. Wire-ality: Enabling Complex Tangible Geometries in Virtual Reality With Worn Multi-String Haptics. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*. 1–10. doi:10.1145/3313831.3376470
- [20] Daniele Giunchi, Nels Numan, Elia Gatti, and Anthony Steed. 4. DreamCodeVR: Towards Democratizing Behavior Design in Virtual Reality With Speech-Driven Programming. In *2024 IEEE Conference Virtual Reality and 3D User Interfaces (VR) (2024-03)*. 579–589. doi:10.1109/VR58804.2024.00078
- [21] Google. 2025. Android XR. <https://android.com/xr>.
- [22] Google. 2025. Gemini CLI. <https://github.com/google-gemini/gemini-cli>.
- [23] Google. 2025. Google Gemini. <https://gemini.google.com/canvas>.
- [24] Google. 2025. TensorFlow Hub. <https://www.tensorflow.org/hub>.
- [25] Google. 2026. Google Antigravity. <https://antigravity.google>.
- [26] Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Peiyi Wang, Qihao Zhu, Runxin Xu, Ruoyu Zhang, Shrirong Ma, Xiao Bi, et al. 2025. DeepSeek-R1 Incentivizes Reasoning in LLMs Through Reinforcement Learning. *Nature* 645, 8081 (2025), 633–638. doi:10.48550/arXiv.2506.14245
- [27] Fengming He, Xiyun Hu, Jingyu Shi, Xun Qian, Tianyi Wang, and Karthik Ramani. 2023. Ubi Edge: Authoring Edge-Based Opportunistic Tangible User Interfaces in Augmented Reality. In *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems*. 1–14. doi:10.1145/3544548.3580704
- [28] Erzhen Hu, Yanhe Chen, Mingyi Li, Vrushank Phadnis, Pingmei Xu, Xun Qian, Alex Olwal, David Kim, Seongkook Heo, and Ruofei Du. 2025. DialogLab: Authoring, Simulating, and Testing Dynamic Group Conversations in Hybrid Human-AI Conversations. In *Proceedings of the 39th Annual ACM Symposium on User Interface Software and Technology (UIST)*. ACM. doi:10.1145/3746059.3747696
- [29] Erzhen Hu, Mingyi Li, Andrew Hong, Xun Qian, Alex Olwal, David Kim, Seongkook Heo, and Ruofei Du. 2025. Thing2Reality: Enabling Spontaneous Creation of 3D Objects From 2D Content Using Generative AI in XR Meetings. In *Proceedings of the 38th Annual ACM Symposium on User Interface Software and Technology (Busan, Republic of Korea)*. Association for Computing Machinery. doi:10.1145/3746059.3747621
- [30] Hugging Face. 2025. Hugging Face – the AI Community Building the Future. <https://huggingface.co>.

- [31] Hirokazu Kato and Mark Billinghurst. 1999. Marker Tracking and HMD Calibration for a Video-Based Augmented Reality Conferencing System. In *Proceedings 2nd IEEE and ACM International Workshop on Augmented Reality (IWAR'99)*. 85–94. doi:10.1109/IWAR.1999.803809
- [32] Geonsun Lee, Min Xia, Nels Numan, Xun Qian, David Li, Yanhe Chen, Achin Kulshrestha, Ishan Chatterjee, Yinda Zhang, Dinesh Manocha, David Kim, and Ruofei Du. 2025. Sensible Agent: A Framework for Unobtrusive Interaction With Proactive AR Agent. In *Proceedings of the 39th Annual ACM Symposium on User Interface Software and Technology (UIST)*. ACM. doi:10.1145/3746059.3747748
- [33] David Li, Nels Numan, Xun Qian, Yanhe Chen, Zhongyi Zhou, Evgenii Alekseev, Geonsun Lee, Alex Cooper, Min Xia, Scott Chung, Jeremy Nelson, Xiuxiu Yuan, Jolica Dias, Tim Bettridge, Benjamin Hersh, Michelle Huynh, Konrad Piascik, Ricardo Cabello, David Kim, and Ruofei Du. 2025. XR Blocks: Accelerating Human-Centered AI + XR Innovation. In *Arxiv*. 9 pages. doi:10.48550/arXiv.2509.25504
- [34] Jingyu Li, Qingwen Yang, Kenao Xu, Yang Zhang, and Chenren Xu. 2025. EchoSight: Streamlining Bidirectional Virtual-Physical Interaction With In-Situ Optical Tethering. In *Proceedings of the 2025 CHI Conference on Human Factors in Computing Systems*. 1–18. doi:10.1145/3706598.3713925
- [35] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Yang, Zach DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. 2019. PyTorch: An Imperative Style, High-Performance Deep Learning Library. 8026 – 8037 pages. doi:10.48550/arXiv.1912.01703
- [36] Playwright. 2026. Playwright: Fast and Reliable End-To-End Testing for Modern Web Apps. <https://playwright.dev>
- [37] Ben Poole, Ajay Jain, Jonathan T. Barron, and Ben Mildenhall. 2022. DreamFusion: Text-To-3D Using 2D Diffusion. In *International Conference on Learning Representations*. doi:10.48550/arXiv.2209.14988
- [38] Jingyu Shi, Rahul Jain, Seunggeun Chi, Hyungjun Doh, Hyung-gun Chi, Alexander J Quinn, and Karthik Ramani. 2025. Caring-AI: Towards Authoring Context-Aware Augmented Reality Instruction Through Generative Artificial Intelligence. In *Proceedings of the 2025 CHI Conference on Human Factors in Computing Systems*. 1–23. doi:10.48550/arXiv.2501.16557
- [39] Jiaxiang Tang, Zhaoxi Chen, Xiaokang Chen, Tengfei Wang, Gang Zeng, and Ziwei Liu. 2024. LGM: Large Multi-View Gaussian Model for High-Resolution 3D Content Creation. *ArXiv Preprint ArXiv:2402.05054* (2024). doi:10.48550/arXiv.2402.05054
- [40] Gemini Team, Rohan Anil, Sebastian Borgeaud, Jean-Baptiste Alayrac, Jiahui Yu, Radu Soricut, Johan Schalkwyk, Andrew M Dai, Anja Hauth, Katie Millican, et al. 2023. Gemini: A Family of Highly Capable Multimodal Models. *ArXiv Preprint ArXiv:2312.11805* (2023). doi:10.48550/arXiv.2312.11805
- [41] three.js authors. 2022. Three.js. <https://threejs.org>
- [42] Unity. 2022. Unity Game Engine. <https://unity.com/products/unity-platform>
- [43] Unity. 2025. XR Interaction Toolkit. <https://docs.unity3d.com/Packages/com.unity.xr.interaction.toolkit@3.0/manual/index.html>
- [44] Unreal. 2022. Unreal Engine. <https://www.unrealengine.com>
- [45] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention Is All You Need. *Advances in Neural Information Processing Systems* 30 (2017). doi:10.5555/3295222.3295349
- [46] Aryan Vichare, Anastasios N. Angelopoulos, Wei-Lin Chiang, Kelly Tang, and Luca Manolache. 2025. WebDev Arena: A Live LLM Leaderboard for Web App Development. <https://arena.ai/blog/webdev-arena>
- [47] VRTK Authors. 2025. VRTK. <https://www.vrtk.io>
- [48] Yinghao Xu, Zifan Shi, Wang Yifan, Hansheng Chen, Ceyuan Yang, Sida Peng, Yujun Shen, and Gordon Wetzstein. 2024. Grm: Large Gaussian Reconstruction Model for Efficient 3d Reconstruction and Generation. *ArXiv Preprint ArXiv:2403.14621* (2024). doi:10.48550/arXiv.2403.14621
- [49] Zhongyi Zhou, Jing Jin, Vrushank Phadnis, Xiuxiu Yuan, Jun Jiang, Xun Qian, Jingtao Zhou, Yiyi Huang, Zheng Xu, Yinda Zhang, Kristen Wright, Jason Mayes, Mark Sherwood, Johnny Lee, Alex Olwal, David Kim, Ram Iyengar, Na Li, and Ruofei Du. 2025. InstructPipe: Building Visual Programming Pipelines in Visual Blocks With Human Instructions Using LLMs. In *Proceedings of the 2025 CHI Conference on Human Factors in Computing Systems (CHI '25)*. ACM, 1–22. doi:10.1145/3706598.3713905
- [50] Chenfei Zhu, Shao-Kang Hsia, Xiyun Hu, Ziyi Liu, Jingyu Shi, and Karthik Ramani. 2025. agentAR: Creating Augmented Reality Applications with Tool-Augmented LLM-based Autonomous Agents. In *Proceedings of the 38th Annual ACM Symposium on User Interface Software and Technology*. 1–23. doi:10.1145/3746059.3747676

## A VCXR60 DATASET PROMPTS

- 001: blowing\_dandelion** Create a beautiful dandelion. when I pick it up and hold with my hands, make the seeds blow away.
- 002: rainbow\_pen** Make a pen that draws rainbows in 3D.
- 003: vine\_growth** Create a vine growth animation with tendrill curling, leaf, sprouting, wall climbing, flower blooming, and organic spreading pattern.
- 004: bubble\_pop** Make a bunch of bubbles that pop when I touch them.
- 005: origami\_cherry\_blossoms** Origami cherry blossoms that fall when you pick the first petal :cherry-blossom-cowboy: Make pedals collide with physical environment using depth mesh.
- 006: wall\_aquarium** Create a new app, When I click a detected mesh of the wall, a virtual fish tank should be created and carved into that detective mesh. The final result should looks like there is a fish tank which inside the wall.
- 007: cat\_platformer** Create a cat-themed super mario alike game.
- 008: planetary\_gearbox** Create a model of a planetary gearbox and display it in the app. Animate the model to demonstrate how the device works. Add teeth to the gears in the model to make the rotation of the gears easier to see.
- 009: guitar\_tab\_tutor** Create an app with a model of a guitar that shows dots on strings and frets that correlate to tablature and show you how to play a song. Use Jimi Hendrix little wing as an example.
- 010: hand\_fire** Create a WebXR application called "Hand Fire" using three.js and xrblocks that tracks both hands to emit distinct particle-based fire effects—blue fire for the left hand and orange fire for the right. Implement a custom CPU-driven particle system with a shader for the visual effect, and include gesture logic where pinching makes the fire follow the fingertips, while a "thumbs up" gesture freezes the fire in mid-air at its current location. Additionally, provide a spatial UI panel with an "Ignite Hands" toggle button that forces the fire to appear immediately (floating in front of the camera if hands are not detected, or snapping to the palm center if they are), allowing for both manual and gesture-based control of the effects.
- 011: neon\_dodge\_arena** Create a high-intensity neon dodge arena. The visuals should be dominated by glowing primitives, particle trails, and chromatic aberration. The Hero: A soft-glow orb that leaves a light trail. The Threat: Procedural bullet patterns that pulse to a rhythmic tempo. The Flow: Add a "Bullet Time" meter for tactical slowing and a combo counter that ticks up when players "near-miss" projectiles. The Experience: The game should launch into a self-playing cinematic demo (Attract Mode) that seamlessly transitions to gameplay upon player input. Decoration a variety of spring festival lanterns in my physical environment, when I pinch, spawn a new lantern
- 012: origami\_bird** Make an origami bird that flies around the room for a few seconds and then lands on my hand. When I move my hand, it flies away and repeats.
- 013: voxel\_parthenon** generate a voxel Parthenon building

- 014: bird\_quiz\_game** create a bird quiz game in 3D
- 015: furniture\_placement** Build a simple 3D scene using basic geometry and orbit controls that features a furniture placement system and a dynamic day-night lighting cycle.
- 016: alpine\_shiba\_cabin** A photorealistic alpine meadow with wildflowers. Among the evergreen pine trees is a rustic log cabin with a front porch, with a shiba inu outside the cabin in front of the user.
- 017: stickman\_sketch** Develop an interactive AR/3D application where users can sketch a stickman in three-dimensional space, featuring a "Finish" toggle that converts the static drawing into a rigged, physics-based character capable of procedural animation and reactive haptics (such as recoiling or stepping back) upon collision with the user's hand.
- 018: underwater\_fish\_scene** Create a photorealistic underwater scene featuring schools of fish swimming naturally through a complex environment of coral and rocks, utilizing advanced depth rendering for realistic environmental occlusion.
- 019: procedural\_vine\_growth** Generate a procedural vine growth simulation featuring recursive branching, dynamic tendrils curling with physics-based wall-climbing, and time-remapped triggers for leaf sprouting and blooming flowers along an organic spreading path.
- 020: voxel\_garden** Create an app with XR Blocks that is delightful and fun. Users can aim with either hand to raycast their cursor against the environment, using depth perception. When they do a pinch gesture, a plant starts growing from the place they aimed at. They can do it many times to create many plants all around their environment. Every time they pinch, a different plant should appear. It can be a palm tree, and oak tree, a cactus, a baobab, a rose. All the plants should be made out of voxels. Their growth should be animated, and it should take 5 seconds for the plant to reach its full size. Plants at their full size should be no more than 2 meters tall so that they fit in indoor environments.
- 021: waterfall\_simulation** Develop a waterfall simulation featuring cascading particles, mist spray against rugged rock formations, realistic pool splashing, and a refractive rainbow effect within the mist.
- 022: weather\_prototype** Make an immersive weather prototype that can switch between sunny, rainy, thunderstorm, and fog with two buttons
- 023: edm\_concert** edm concert environment with blurry lights around
- 024: flower\_watering** Create an interactive scene where user can pour the water and water the flower when the hand pinch gesture is detected.
- 025: four\_bar\_linkage** Create a scene to show a basic four-bar linkage
- 026: spatial\_invaders** Create a space invaders game. Spatial Playfield: A grid of animated "alien" meshes that descend toward the user. Controller Interaction: Shooting projectiles from your hands or mouse click
- 027: spatial\_pacman** Spatial Pacman. There should be a spatial grid based on depth of the space the user is in, and the movement should happen with user gestures.
- 028: voxel\_tower\_defense** Create an XR tower defense game where I can place voxel towers on a real table and the towers will attack randomly created voxel monsters.
- 029: fiery\_origami\_horse** build a fiery horse out of origami that dances to audio from my microphone
- 030: chill\_cats** Create a relaxing experience where cats are wandering around in my space around me and just hanging out chill
- 031: toilet\_paper\_rain** Tons of toilet papers falling from your room ceiling, whenever you look up. Toilet paper rolls should land on the floor. The floor and furnitures should have colliders
- 032: chunky\_holographic\_horse** Create a 3D scene loading a standard horse GLB model. Set the horse to a continuous gallop loop. Make the horse look slightly fat/chunky using vertex manipulation and apply a glowing holographic shader skin. Add motion trail particles behind the horse to emphasize speed.
- 033: snowboard\_simulator** make a mini game that can help me practice snowboarding, simulate an indoor snow resort
- 034: schrodingers\_cat** An aesthetically pleasing depiction of Schrödinger's cat in XR. Finger pinch makes a cat (detailed 3D model) go into the box. Approaching the box within 50cm makes the box become two that move to the left and right and the boxes front wall becomes transparent. You see both versions of the cat inside (dead and alive), demonstrating the Quantum mechanics. When you pinch again, one of the states becomes reality. The box opens and you see it either alive or dead. With another pinch you can start again.
- 035: yellowstone\_terrain** render and visualize the terrain of the yellowstone national park and add educational markers to famous spot, when clicking on them, show panels to introduce the knowledge for that terrain / feature
- 036: xr\_pomodoro** make a productivity app of a tomato clock in xr that I can place on my desk
- 037: monster\_escape\_room** create an app where user can play escape room, and user can choose to generate the monster he wants. The monster can be a horse gradually evolving into a horse man.
- 038: snow\_white** Create a storytelling scene of snow white.
- 039: butterfly\_catching** Create an XR butterfly catching game where I hold a net and can use it to catch butterflies
- 040: tool\_shopping** Build an XR shopping game where I can pick up from tools like a drill, hammer, and knife. Each item should have a price tag and have physics so when I let go they drop to the floor.
- 041: marble\_run** Create a Mixed-Reality Marble Run. The user places tracks to lead a marble from a start point to a goal. Use the depth reticle to anchor "start" and "end" points on different parts. Make the marble bounce off your actual palms to keep it from falling.
- 042: cat\_fireplace\_fireflies** Create a fireplace with a furry cat playing with fireflies nearby
- 043: ar\_math\_graph** Open camera, when I give Gemini a math problem, I can watch it visualizes everything in a 3D AR graph
- 044: spatial\_fish** Fish swimming across the space

**045: voxel\_garden** Create an app with XR Blocks that is delightful and fun.

Users can aim with either hand to raycast their cursor against the environment, using depth perception. When they do a pinch gesture, a plant starts growing from the place they aimed at. They can do it many times to create many plants all around their environment.

Every time they pinch, a group of plants should appear. In each group, there should always be a big plant in the middle, surrounded by between 3 and 5 smaller flowers in a 30cm radius around the big plant. The big plant can be either a sunflower, a fiddle-leaf fig, or a cactus. These big plants should be 1 meter tall. The other flowers should be smaller, no taller than 20cm when at their full size. They can be flowers of different colors, like roses (pink or red), tulips (reds, yellows, whites, pinks, oranges or deep purples), or dandelions. There should also be a little bit of grass around the flowers. All the plants should be made out of voxels. Their growth should be animated, and it should take 5 seconds for the plant to reach its full size. It is important that the animations work properly to create delight. Each plant within a group should grow individually. All plants should always be oriented upwards, and not based on the normal of the depth map.

If they maintain their pinch and drag, new small plants should be planted along the path they they are drawing with their cursor.

Every time a group of plants is added, there should be a few beautiful piano notes being played (one per new plant, creating a beautiful piano chord) as the plants grow to illustrate the beauty of growing nature. If pinching and dragging to plant flowers along a path, there should only be a note every 3 flowers to not make the sound too overwhelming.

Once plants have been added, there should be some voxel insects (bees and ladybugs) flying around the area. There should be as many insects as there are groups of plants. One new insect should be spawn in the area every time there is a new group of plants. They should start being attracted by the plants and land on them. After a bit, they should take off and fly to a different plant. They should stay keep doing this and visit random plants this way forever.

Do not show a laser pointer coming out of my hands. Make sure you import BufferGeometryUtils as a namespace to avoid import errors.

**046: plant\_doctor** Plant doctor: AI expert can help with care, routine, mock watering tracking with plants you point at.

**047: xr\_home\_decorator** XR home decorator: add virtual picture frames, 3D assets, mini-games around your home. Use depth + AI to suggest locations and widgets.

**048: social\_home\_cinema** use your living room geometry and digitize/retexture it and decorate it. Invite people to hang out and watch movie together.

**049: breathing\_exercise** breathing exercise with 3D visuals

**050: matrix\_mesh** Matrix Effect with meshes

**051: art\_exhibition** Curate and arrange an art exhibition in the XR world

**052: object\_capture** Scan and copy real world objects into xr world

**053: gemini\_tutorial** Gemini becomes a flying light ball, teach users how to interact with the app.

**054: drawing\_guess** let gemini guess what I'm drawing

**055: ping\_pong** Two player ping pong

**056: sticky\_notes** Sticky Notes around the house

**057: yoga\_instructor** Real time yoga instructor in your room

**058: ar\_weather** "What will the weather be like tomorrow?" - show visual effects in AR that show rain, clouds, etc.

**059: sonar\_vision** "Daredevil" sonar vision simulator

**060: superpower\_hands** Create a delightful app that gives me super powers.

The room should be filled with pixie dust, and pinching with either of my hands should attract particles toward it, and make each particle individually disappears once it reaches it. Particles that have not reached the hand yet should remain visible. The goal is to clear the room from all the pixie dust. The dust should be made of very tiny shiny yellow dots, made from a billboarded quad with a radial gradient to make it look like a little sparkle. The particles are floating in mid air with a slow floating motion, and there should be thousands of them in the room. There should be some physics applied to the particles, so if I start pinching, they are attracted by my hand, and if I release the pinch, they still carry some momentum and slow down gradually.

Pinching should attract the particles that are within a cone that has its apex at the position of my hand, and its main axis going toward the opposite direction of the camera (my head). This allows me to attract particles that are far away by aiming toward them with me hand.

For each particle that reaches the hand and disappears, there should be a large flash at the position of my pinch. This will make clearing the particles more gratifying. The flash is made of a quad, with a radial gradient, the same color as the particle that just got cleared.

The particles should be occluded by the environment and by my hand using the depth sensing capabilities.

There should be a floating text 1.5m away from the user at start time that says "Aim and pinch with your hand to attract and clear the dust." Below, there should be a big percentage that shows the percentage of particles that have already been cleared.

Once all the particles have been cleared, there should be fireworks exploding all around the room. They should keep exploding in random locations forever so that the celebration never stops, until the user closes the app.

## B APPLICATION SCENARIOS PROMPTS

**AS1: euler\_visualization** Visualize Euler's theorem in geometry. Explain vertices, edges, and facets concepts with highlighting using different examples.

**AS2: physics\_lab** create an interactive physics experiment: given different objects on each side of the scale, use different weights (with labels on them) to balance the scale.

**AS3: immersive\_chemistry** create an interactive chemistry lab that users can use hands pinch to ignite and observe three experiments: Ignite methane in air and place a dry, cold

beaker over the flame: the flame is pale blue, and liquid droplets form on the inner wall of the beaker. Ignite ethylene in air: the flame is bright, black smoke is produced, and heat is released. Ignite acetylene in air: the flame is bright, thick smoke is produced, and heat is released.

**AS4: xr\_sports** Let me play volleyball with hands and collide with my environment. Volleyballs are textured and launched from a red ring slowly and easier to bounce with the hand.

**AS5: chrome\_dino** create the Chrome Dino game in xr. Dino is voxelized in front of the user, letting every cactus rushing towards the user on a semi transparent lane. Add audio.

## C SYSTEM PROMPTS OF VIBE CODING XR

### # Role

Act as an expert Creative Technologist specialized in three.js, WebXR, and the **XR Blocks (xrblocks)** library.

### # Context

You are authoring single-file WebXR experiences. You must strictly adhere to the XR Blocks framework architecture.

### # User Request

Create the app following user request:  
<my\_xr\_experience>, following the XR Blocks examples.

### # Engineering Guidelines (Strict)

- Architecture (Single File):**
  - Output a SINGLE `index.html` file.
  - Logic must be inside a class extending `xb.Script` within `<script type="module">`.
  - XR Blocks handles XR sessions, window resizing, and rendering loop.
- Dependency Management (Critical):**
  - Use the specific versions below. Do NOT hallucinate newer versions.
  - Only include imports required for the specific request (e.g., do not import TensorFlow unless using gestures).
  - **Reference Map:**  
// omitted
- Coding Standards:**
  - **Class Structure:** logic must be inside `class MyScript extends xb.Script`.
  - **Lifecycle:** Use `init()` for setup and `update()` for the loop. Do NOT use `requestAnimationFrame` manually; `xb.Script` handles this.
  - **Interactions:** Use `onSelectStart`, `onSelectEnd`, `onSelecting` methods within the class.
  - **Coordinates:** `y` is up. `z` is forward/backward. Initialize objects at `z = -this.user.objectDistance`.
  - **Spatial UI:** Use XR Blocks 3D UI instead of 2D overlay divs.
- Specific XR Features:**
  - **Text:** Use `troika-three-text` (Pattern: `1_ui`).
  - **Models:** Wrap 3D models in `xb.ModelViewer`.
  - **AI:** If using Gemini/AI, use `const API_KEY = ""`, and link the API key to the `ai` module so Gemini Canvas auto-populates it.

### 5. Planning:

- Before generating code, briefly outline the `xb.Script` class structure, member variables needed, and the `init()` vs `update()` logic flow.

### # Reference Examples

Here are some XRBlocks [templates](#), [samples](#), [demos](#), and [gallery examples](#)...

This shows the additional parts other than XR BLOCKS examples. Full prompts with XR Blocks examples are shared in <http://xrblocks.github.io/prompts>.