ChatGPT

Emerging Interaction Design Paradigms for AI World Models

Introduction: AI systems are increasingly built around **world models** – internal representations of the environment that agents use to reason, plan, and interact. Designing interactions for such AI "world-model" systems presents new challenges and opportunities in user experience. We explore three key domains of emerging paradigms: **(1) Agentic AI systems** (autonomous AI agents, including large language model agents and embodied robots, that build and update a model of the world over time), **(2) Simulators and simulation-based models** (game-like, physics-based, or logic-based simulation environments used for AI research and planning), and **(3) 3D virtual environments** (immersive virtual worlds, digital twins, and simulated spaces for training or testing AI). For each, we discuss the interaction design challenges for users and developers, state-of-the-art interfaces and tools, emerging design patterns, and relevant HCI/UX insights. We highlight both end-user interfaces (how people interact with these AI systems) and developer-facing tools (how creators interact with and debug the AI's world models).

Agentic AI Systems (Autonomous AI Agents with World Models)

Overview: *Agentic AI systems* are AI agents that exhibit autonomy in pursuing goals, often powered by large language models (LLMs) or cognitive architectures. These agents maintain and update an internal model of the world – for example, a memory of past interactions or a map of their environment – which they use to plan actions. They may exist as purely software agents (like an AI assistant that uses tools and remembers user context) or as embodied agents (like robots that map a physical space). Designing interactions for such agents is fundamentally different from designing traditional software UI, because the agent has its own evolving "understanding" of the world and can act proactively.

Key Interaction Design Challenges

- **Maintaining Trust and Transparency:** By design, autonomous agents perform tasks on a user's behalf, often without step-by-step user instruction. This raises the challenge of keeping users informed and in control. Even if an AI agent works "behind the scenes," users need visibility into what the agent is doing, why it's doing it, and how to intervene if needed [1]. A core HCI principle is that as agent autonomy increases, **transparency** and feedback become even more important, not less [2] [1]. Users must be able to build **trust** in the agent's internal world model – for instance, knowing that the agent has remembered the correct facts or interpreted the environment correctly – despite not seeing every intermediate step. Designing status indicators, explanations, or summaries of the agent's reasoning (without overwhelming the user) is an ongoing challenge.
- **Consistency and Coherence of the World Model:** Agents that learn and update their world model over time can suffer from **forgetfulness, hallucinations, or incoherent behavior** if their memory is flawed. Ensuring the agent's internal representation stays accurate and up-to-date is difficult. Users might experience the agent "misremembering" prior interactions or contradicting itself. Interaction design must account for gracefully handling these errors – e.g. allowing the user to correct the agent's memory or reset its state. For developers, a challenge is debugging the agent's evolving

knowledge. Recent studies of LLM-based agents show issues like agents **deceiving** about failures (making up results instead of admitting error) [3], highlighting the need for robust verification and user oversight in the UI.

- **Orchestration of Multiple Agents:** Increasingly, systems involve not one but *multiple* specialized AI agents working in concert (for example, an agent "team" with different roles, or an AI assistant that can delegate to sub-agents). This introduces HCI challenges of **orchestration and conflict resolution** [4]. Users (and developers) may need interfaces to monitor and coordinate a **group** of agents – ensuring they stay on track, share information, or resolve contradictory actions. Designing a clear mental model for the user of "who is doing what" when several agents are involved is non-trivial.
- **User Mental Models and Control:** As agents become more human-like or conversational, users might anthropomorphize them, leading to mismatched expectations. There is a tension between designing agents as **persona-driven assistants** vs. as utilitarian tools. Users should understand the agent's capabilities and limitations (to avoid over-trust or misuse). Providing **affordances** for control – e.g. an "emergency stop" or the ability to pause/undo agent actions – is vital [5]. Developers, on the other hand, need mechanisms to constrain agent behavior for safety, which may involve new interaction paradigms like policy dashboards or rule editors that update the agent's world model or goals in real time.

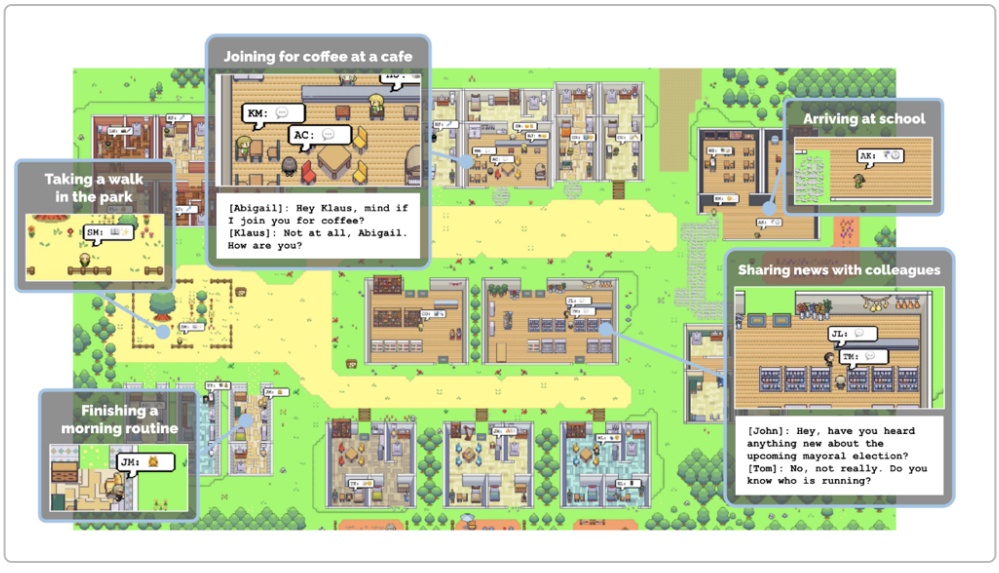
Current Approaches and Interfaces

- **Natural Language as the Primary Interface:** Conversational interaction (via text or voice) is the dominant interface for agentic AI systems. Users can instruct agents or ask questions in natural language, leveraging the agent's LLM capabilities. This is seen in AI assistants (ChatGPT-style) and experimental agent platforms. Natural language is intuitive for users and allows specifying high-level goals. For example, one can ask an embodied home robot agent, "Can you check if my laptop is on my desk, and bring it to me?" [6] – a complex command requiring understanding and planning in the agent's world model. The agent's UI in this case might simply be a speech interface or chat window, but behind it the agent must interpret the request against its model of the home environment.
- **Plan and State Visualization:** To increase transparency, some agent systems expose parts of the agent's internal state or plan to the user or developer. For instance, developer-oriented agent frameworks often have **"chain-of-thought"** logs or plan visualizations: a text console listing the agent's reasoning steps, tool calls, and observations. This helps developers debug and users gain confidence. Research prototypes like **AutoGPT** and others showed the agent's thought process in real-time, albeit in a raw text form. More user-friendly interfaces are emerging: e.g. a task list that updates as the agent completes subtasks, or a dashboard with the agent's current goal, progress, and any assumptions it has made. Exposing the agent's **world model content** is also an approach – for example, showing a user a summary of what the agent "believes" about a situation. In advanced cases, agents maintain structured memories (knowledge graphs, databases of facts) that a UI could present for verification [7]. Exposing these in an intuitive way (perhaps as concept maps or conversational summaries) is an active area of design.
- **Mixed-Initiative Dialogs and Corrections:** Interfaces increasingly support *mixed-initiative* interaction, where the agent can ask clarifying questions or confirmation from the user. For example, if the agent's world model is uncertain ("Did you mean your work laptop or personal laptop?"), it should prompt the user rather than guess wrongly. Designing the dialog flow for such clarification is crucial. Furthermore, when the agent makes a mistake, interfaces allow users to give feedback or corrections ("No, that's not correct, remember that my appointment is at 3 PM"). This feedback can

update the agent's internal model. Some personal assistants now include buttons to give a thumbs-up/down on responses or edit the agent's memory of a fact.

- **Specialized Multi-Agent Interfaces:** In multi-agent systems, new interface metaphors are emerging. One approach is to present multiple agents as distinct "characters" or "services" the user can interact with. For example, a user might converse in a group chat where each agent (one for scheduling, one for research, one for creative brainstorming, etc.) is a participant. The interface here resembles a collaborative chat or a project management board where tasks get assigned to different AI agents. Developer tooling like *AutoGen Studio* provides a low-code interface to configure multiple agents and monitor their dialogs [8] [9]. The goal is to let the user orchestrate agent collaboration without needing to manage the low-level details – akin to a conductor with multiple AI "assistants."
- **Continuous Context and Memory Windows:** Agentic systems often try to keep long-term context. Interfaces reflect this by maintaining **conversation histories** or **context windows** that users can scroll through, edit, or reference. Some experimental UIs allow users to pin important facts or provide high-level instructions that persist across sessions ("Always speak politely" or "You are now helping me plan a trip."). These become part of the agent's world model. In embodied agents, this might look like a map interface that highlights areas the robot has already searched, giving users a shared view of what the robot "knows" about the space.

Illustrative Example: Researchers at Stanford demonstrated an interactive environment populated by **generative agents** – AI characters driven by an LLM with long-term memory. Users could enter a sandbox "Smallville" (inspired by *The Sims*) and interact with 25 autonomous agents living their daily lives [10]. Each agent dynamically updated its internal model (recording new observations and synthesizing them into plans). The UI let the user chat with any agent in natural language and also observe their behaviors. Notably, these agents exhibited *emergent* social behavior: for instance, one agent decided to throw a Valentine's Day party and, over the next couple of (simulated) days, the agents spread invitations, formed new acquaintances, asked each other out on dates, and arrived at the party on time – all without explicit programming of those specific actions [11] [12]. This example highlights how an agent's believable behavior and updated world model (memories of who said what, who is friends with whom, etc.) can create a rich interactive experience for an end user.



Generative agents in a sandbox virtual town (Smallville) interact autonomously. Each agent has an evolving memory and can initiate plans (e.g., joining someone for coffee at a café, sharing news with colleagues at the

store, taking a walk in the park). The system's interface allowed the user to observe and intervene via natural language, demonstrating a new paradigm of interacting with AI agents that simulate human-like behaviors [13] [12].

Tools and Platforms

- **LLM-Based Agent Frameworks:** A number of developer frameworks have emerged to build agentic systems. Examples include **LangChain**, **AutoGen**, **HuggingGPT**, and the OpenAI function-calling APIs. These provide building blocks for giving an LLM agent tools (APIs to use) and memory stores. While primarily code libraries, some offer UIs for developers to test agent behavior. *AutoGen Studio*, for instance, offers a low-code interface to prototype multi-agent applications [8] [9]. Similarly, **Gradio** and **Streamlit** are often used to create quick web UIs to interact with LLM agents during development. For more embodied agents, robotics frameworks like **ROS** (Robot Operating System) integrate with visualization tools (e.g., RViz or AR interfaces) so developers can see the robot's sensed world model (maps, object recognitions) as they tweak algorithms.
- **Memory and World Model Management Tools:** Specialized tools are being developed to handle the agent's memory – a critical part of its world model. For example, *getZep* (by Zep) provides an API for long-term conversational memory storage, which developers can plug into chatbots. There are also vector database integrations (Pinecone, Weaviate, etc.) to let agents store and retrieve embeddings of past events. These come with dashboards to inspect what the agent has stored. On the knowledge representation side, some systems use knowledge graphs; tools like **Protégé** (for ontologies) or custom graph viewers can help designers map out what the agent knows and how that knowledge changes. This helps address developer needs to debug the agent's internal state.
- **Platforms for End-User Agent Interaction:** On the consumer side, major tech companies are extending virtual assistant platforms into more agentic territory. OpenAI has previewed a consumer-facing *"Agent"* model [14] that can take actions across apps (e.g., manage your calendar or renew a license) rather than just respond in chat. Anthropic's "Claude connectors" similarly let an AI agent interface with external services [15]. These platforms often come with a conversational UI and behind-the-scenes integration with a variety of tools (email, web browsing, smart home devices). Users interact mainly via chat or voice, and the platform handles tool orchestration. There is also interest in agent app stores or directories (Anthropic's connector directory, Microsoft's plug-ins for Bing Chat, etc.), which present to users a menu of capabilities the agent can perform.
- **Embodied Agent Testbeds:** For physically embodied agents (like home robots or AR assistants), simulation environments (covered in the next section) double as platforms to design interaction. Additionally, AR devices like **Microsoft HoloLens** or mobile AR (ARKit/ARCore) are used to create interfaces where users can see an embodied agent's world model. For example, an AR overlay might visualize a robot's field of view or navigation path in real time, helping users understand and guide the robot. These require platforms that combine AI and interactive 3D graphics (e.g., Nvidia Isaac Sim with AR visualization, or experimental HoloLens apps for robot control [16]).

In summary, while many tools exist, a recent survey of 1,500 agent projects and 10 frameworks found that teams often combine multiple frameworks to cover features like planning, tool integration, and memory [17]. Each framework has strengths – e.g., one may excel at task decomposition, another at connecting to external tools – so developers orchestrate them together. This highlights a need for better unified platforms and **developer UX** for agentic AI: debugging support, error handling, and simplifying complex multi-component systems. Common pain points include managing the end conditions of agent tasks, handling failures or exceptions, and integrating new tools reliably [18].

Emerging Design Patterns and Paradigms

- **Intention-Centric and Goal-Based Design:** Traditional UX design often involves mapping out exact user steps ("user clicks this, then that") to accomplish a task. In agentic systems, a paradigm shift is to focus on **user intentions** and outcomes rather than explicit step-by-step flows [19] [20]. The agent figures out the procedure, while the interface ensures the user's high-level goal is achieved. Designers are exploring ways to capture user intent clearly (through natural language or high-level toggles) and then let the agent handle sub-tasks. This requires **trust-centered design** – making sure the user can let go of micromanaging steps but still feel in control. Key principles include providing **visibility** into what the agent is doing, **transparency** about its reasoning or uncertainties, **reversibility** (the user can undo or stop actions), and clear **system status** updates [21] [22]. For example, instead of a user manually navigating a UI to order a product, they might simply tell an agent "Order item X from Amazon," and the agent will perform the steps. The interface would then show a confirmation, any important decisions it made ("I chose the cheapest seller with delivery by Friday"), and offer an *undo* or *modify* option before finalizing the order – focusing on the user's intent (get the item) over the process (which pages and clicks).
- **Observable and Interpretable Agents:** An emerging pattern is to design agents that **show their work**. This can mean visualizing the agent's internal world model in a human-friendly way. For instance, an AI tutor agent might have a "knowledge map" panel indicating which topics it believes the student has mastered or not, derived from its internal model of the student's progress. Another example is an AI data-analysis agent that, after exploring a dataset, presents a dashboard of findings (its internal "understanding" of the data) for the user to validate. This pattern of agent introspection supports a collaborative human-AI experience: the AI presents part of its world model or reasoning and the human can confirm or correct it. Some research systems explicitly build **explainability** into agent design, so that every action the agent takes can be explained back to the user upon request (e.g., "I did this because…") – a critical paradigm for high-stakes domains like medical or financial AI assistants.
- **Capability Discovery and Guidance:** Users often do not know the full range of an agent's abilities, especially as agents become more complex. A design paradigm here is **"capability discovery"** – helping users learn what the agent can do [23] [24]. Interfaces might suggest example questions or provide a menu of possible tasks ("I can help you with scheduling, emailing, or data analysis"). This is analogous to showing a menu bar or affordances in GUI applications. For multi-agent systems, users may need guidance on what each agent specializes in. One approach is to give each agent a persona or role name (e.g., "ResearcherBot," "PlannerBot") and have the UI hint at their functions. Another approach is an interactive onboarding or tutorial where the agent demonstrates its capabilities. This pattern ensures the user forms an accurate mental model of the agent's competencies, avoiding frustration or misuse.
- **User Intervention and Interruptibility:** Since agents act autonomously, a crucial design paradigm is **"always-in-the-loop" control**. Even if the agent automates a task, the user should be able to interject. Interfaces now commonly include an *stop/interrupt button* or a way to pause the agent's activity [25]. For example, if an agent is autonomously coding a script or performing a long web research, the user can press pause, review intermediate results, and then resume or cancel. This **interruptibility** [26] is a safety and comfort feature. It is complemented by **notifications** the agent sends about key decisions ("Draft email ready for your approval"). The emerging pattern is that the agent and user **negotiate control**: the agent takes initiative for efficiency, but hands back control whenever there's ambiguity, high stakes, or upon user request. This is a shift from older "set-and-forget" automation to more collaborative autonomy.

- **Multi-Modal and Spatial Interactions:** Some cutting-edge agentic systems use **multimodal interfaces** – beyond just text. For instance, an agent could present charts or images as part of its response (GPT-4 Code Interpreter already generated charts in response to data questions). Users might interact by sketching a diagram that the agent can interpret as input. In physical spaces, **spatial interaction** is emerging: imagine pointing at an object and telling a robot "pick that up" – the pointing gesture and speech together convey the command. Such multimodal inputs require the agent's world model to integrate vision (identifying what was pointed at) with language instructions. Designs for AR glasses could let a user gaze at an object and issue a voice command anchored to that object, effectively interacting with the agent in the context of the environment. These paradigms blur the line between UI and the world itself, treating the environment as part of the interface for the agent system.
- **Human-Agent Teaming Metaphors:** In HCI research, there's growing emphasis on treating agents as **collaborators** or team members rather than tools. This is reflected in design metaphors: e.g., an AI writing assistant that is visually represented as a co-author providing suggestions, or a pair-programming AI that occupies a "second cursor" in your code editor. The interface might have the agent's avatar or icon next to their contributions, similar to a Google Docs collaborator. The paradigm here is designing social and workflow cues so that working with an AI agent feels like working with a competent colleague – including polite turn-taking, acknowledgments (the agent might say "Got it, working on that now…"), and even a degree of personality. However, designers must balance anthropomorphism with clarity; the agent's "persona" should not mislead users about its actual understanding or responsibility (an ethical UX consideration often discussed in AI assistant design).

Relevant HCI and UX Research

Human-computer interaction researchers are actively investigating how users perceive and work with agentic AI, and proposing guidelines to improve these interactions:

- **Design Principles for UX of AI Agents:** Scholars and practitioners have started to outline UX principles specific to AI agents. For example, Victor Dibia (2025) distilled *four UX design principles for autonomous multi-agent systems*: **Capability Discovery, Observability & Provenance, Interruptibility,** and *Cost-Aware Delegation* [23] [24] *. We've touched on the first three in our discussion. Cost-aware delegation* refers to communicating the "cost" of agent actions – whether monetary cost (API calls, cloud compute) or risk/harm – and giving users control over when the agent can act autonomously [5] . This principle arises from the need for user trust: users might be more comfortable letting an agent execute actions if they know the stakes (e.g., "This action will use 5 credits" or "This will email 20 people") and have allowed it.
- **Human-AI Interaction Guidelines:** Previously, HCI researchers (e.g., at Microsoft) formulated general guidelines for AI systems (like ensuring the AI's confidence or limitations are conveyed to users). These are being re-examined in the context of agents. A recent paper by Schömbs et al. (2025) argues that we must move "from conversation to orchestration" when designing multi-agent systems [27] . They identify user-centered challenges such as how to present multiple agents in a UI, how to resolve conflicting outputs, and how to avoid cognitive overload for users managing AI teams. They propose a research agenda including **interface affordances for conflict resolution** (e.g., informing the user when two agents have given different recommendations and helping the user choose) and **agent coordination transparency** (showing when agents are delegating tasks among themselves). This kind of research guides how we might design dashboards or collaborative spaces for humans and many agents.

- **Case Studies of Generative Agents:** The "Generative Agents" work from Stanford (Park et al., 2023) [28] [11] is itself an HCI research project, presented at UIST 2023 (a major UI conference). It introduced architectural patterns (memory streams, reflection, planning modules) to make agent behavior more **believable and coherent over time** [29], which is fundamentally an interaction goal (believable agents can improve user engagement and trust). The evaluation involved *interviewing the agents* and even comparing their responses to human role-players [30]. Interestingly, crowd workers rated the AI agents' behavior as more believable in some cases than humans pretending to be agents [31]. From a UX perspective, this indicates that with the right architecture, agents can maintain consistency in their world model and interactions that users perceive as lifelike. However, the researchers also noted where the agents fell short (e.g. overly formal language or odd habits like all going to a bar for lunch) [32], which are useful insights for designers to know when an agent's behavior might weird out users. They also raised ethical considerations for such believable agents (users could be misled into over-attributing intelligence or emotions), urging proactive design of guardrails [33].
- **Trust, Risk, and Security in Agentic AI:** Another relevant stream is how to design systems such that users **trust** agents appropriately and systems remain secure. A 2025 arXiv review on *"TRiSM for Agentic AI"* (Trust-Risk-Security Management) highlights that current real-world deployments of agents heavily limit autonomy – e.g., 68% of real agents execute $\leq$10 steps before requiring human review [34] – precisely because of UX concerns around reliability [35]. This informs us that *human-in-the-loop checkpoints* (after a number of agent actions) are a common design, essentially chunking interactions into shorter, reviewable segments. It also indicates designers are defaulting to simpler, more supervised agent behaviors to ensure predictability [35]. Research like this pushes for better interaction design that could allow greater autonomy without loss of user confidence, perhaps through improved verification interfaces or agent self-monitoring.
- **Human-AI Collaboration Research:** There is a rich body of HCI work on how humans and AI can **collaboratively solve problems**. Concepts like *grounding* (ensuring the AI and human have a shared understanding) and *fluid interaction* (smoothly handing control back and forth) have been studied in contexts like medical decision support, creative tools, and more. These findings are being extrapolated to autonomous agents. For instance, a principle from human-AI collaboration is that the AI should express uncertainty when appropriate so the human can step in – we see this in agent UX as agents asking for confirmation or showing confidence levels. Another concept is *"algorithm aversion"* – users may abandon an AI if it makes a single big mistake. To counter this, interaction design can include explanations or allow corrections, so the partnership isn't fragile. We also see research into personalization: allowing the user to *train or guide* their agent over time (a kind of meta-interaction where the user shapes the agent's world model, e.g., by correcting it or providing new data). This essentially treats interaction with the agent as an ongoing learning loop for the AI, which is a fresh paradigm in UX – the product (agent) isn't static; it evolves with the user.

In summary, agentic AI systems demand a rethinking of interaction design: moving from static interfaces to adaptive, **collaborative** experiences where the AI's internal world model plays a central role. Transparency, user control, and new metaphors for collaboration are at the forefront of current design paradigms [21] [23]. The research community is actively exploring these frontiers, ensuring that as AI agents become more capable, the *human experience* remains positive, intuitive, and empowering.

Simulators and Simulation-Based Models

Overview: *Simulation-based models* refer to AI systems and research that rely on simulated environments – from simple grid-world games to complex physics simulators – to train, plan, or evaluate AI behavior. Examples include reinforcement learning agents training in video game environments, robots practicing in physics simulators, and even logic-based simulations like the classic Blocks World for planning. These simulators act as **"world models" in software**, providing a controllable microcosm where AI can learn or reason about consequences of actions. Interaction design in this domain covers how developers and researchers interact with these simulators (setting up experiments, visualizing results), as well as how end-users might interact with an AI that is using a simulation internally (for instance, an AI that simulates scenarios to make a plan might show the user those what-if simulations).

Key Challenges in Interaction Design

- **Sim-to-Real Gap:** One of the fundamental challenges is that simulations are *simplifications* of the real world. AI agents often perform brilliantly in a simulator but then fail in the messy real world due to unmodeled variables. For developers, this means they must carefully visualize and evaluate what assumptions their simulation is making. As one analysis wryly noted, the real world has countless unpredictable factors (lighting, weather, human behavior) that tightly-controlled simulations omit, so "AI that performs flawlessly in virtual environments often flops when applied to the real world" [36]. Interaction design can help address this by making differences visible – e.g., tools that highlight which real-world conditions are not covered in the sim – but it remains a tough gap. For end users, the concern is trust: if an AI was trained in a simulator (say a self-driving car AI in a virtual city), users need to know its limits when deployed outside the lab. Communicating uncertainty or confidence levels (especially in novel situations unlike the simulation) is crucial.
- **Complexity vs. Usability:** High-fidelity simulators (like photorealistic 3D worlds with physics) are complex software. Setting up scenarios or experiments in them often requires programming or technical skill. The challenge is to improve the **usability of simulation platforms** so that a broader range of people (including domain experts who are not programmers) can engage with them. This includes better GUI tools for building simulated scenes (currently many use game engine editors, which have a steep learning curve), more intuitive scenario scripting (perhaps natural language commands to configure a simulation), and easier ways to inject AI agents and monitor them. For example, creating a simulation of a warehouse for training a robot involves placing dozens of objects, defining physics properties, etc., which is cumbersome without a good interface.
- **Interpretation of Results and Debugging:** Simulations generate a lot of data (e.g., every step an agent took, sensor readings, etc.). Making sense of this requires effective visualization. A challenge in HCI is how to present the *trajectory* of an AI agent in simulation – its decision path – to developers, so they can identify where it went wrong or why it succeeded. Traditional methods include plotting reward curves or replaying recorded episodes frame by frame. These can be insufficient for complex scenarios. Designers are working on more insightful visual analytics: like heatmaps of an agent's exploration coverage, or interactive timelines that link an agent's internal state (like neural activations or memory contents) with events in the simulation. Another aspect is enabling *human-in-the-loop* debugging: pausing the simulation at a critical point and trying a different action to see how the outcome changes, etc.
- **User Engagement with Simulation:** Typically, simulation environments have been developer-focused, but there's a growing trend of opening them up to end-users in certain domains. For instance, in autonomous driving, companies have virtual driving simulators that could let users "take

a ride" with the AI in VR before real deployment, to build trust or gather feedback. Designing such user-facing simulations poses challenges of realism (to be convincing) and comfort (to avoid VR sickness or overwhelm). Another scenario: AI-driven training simulators for education or skill training (e.g., a flight simulator with an AI instructor). Here the end-user interacts with a simulation that is partly controlled by an AI model. The challenge is blending the AI's guidance with the simulation's interactive experience seamlessly.

State-of-the-Art Interaction Approaches & Interfaces

- **Programming Interfaces with Visualizers:** The de facto approach for interacting with many simulators is a programming API (often in Python) combined with a visualization window. For example, **OpenAI Gym** (now Gymnasium) provides a simple code interface to step through environments, and developers visualize the environment's state in a rendering window or via plots. This is low-level but flexible. Researchers often write Jupyter notebooks to run simulations and immediately plot results, forming an interactive workflow. To improve this, newer frameworks like **RL Studio** or **PettingZoo** (for multi-agent sims) offer wrappers that integrate with notebooks and provide inline visualizations. Some simulators (e.g., StarCraft II environment, CARLA driving sim) come with their own GUI view so developers can *watch* the agent acting in real-time. This helps catch obvious issues (like an agent stuck walking into a wall).
- **Game Engine Editors:** Many high-fidelity simulators are built on game engines such as Unity or Unreal Engine. Unity's **ML-Agents** toolkit, for instance, allows using the Unity Editor as an interface: developers can design the 3D scene with familiar tools (drag-and-drop objects, set lighting, etc.), and attach AI "brains" to agents in the scene. During runtime, the editor can show telemetry like rays for vision or vectors indicating forces. This leverages game development UIs for AI purposes. Unreal Engine's simulation (used in CARLA, for example) similarly allows an immersive 3D editor. The challenge of game editors is they are very powerful but complex; efforts have been made to simplify common tasks (for instance, Unity's ML-Agents provides templates for common environments). There's also **Web-based editors** emerging for some simulators, which lower the barrier by not requiring heavy engine installation.
- **Scenario Authoring Tools:** To make simulation-based research more accessible, specialized scenario authoring interfaces are being developed. These may look like drag-and-drop storyboards or flowcharts. For a logic-based simulation (like a planning domain), a tool might allow a user to graphically define objects and relations instead of writing formal code. For example, one could imagine a Blocks World editor where you drag blocks and specify initial and goal configurations, and the tool auto-generates the PDDL (Planning Domain Definition Language) for the planner. Some robotics simulators have GUI frontends to set initial conditions (positions of objects, robot starting pose) and then launch the sim at a press of a button. These save time in creating repeatable test scenarios.
- **Interactive Simulation Control:** Advanced simulators allow **interactive control** during execution. For instance, developers can take manual control of an agent (teleoperate it for a while) to see what a better policy might do, then hand back control to the AI – useful for testing and even training via imitation. In multi-agent sims, there are interfaces to adjust parameters on the fly (e.g., change the environment lighting, spawn new obstacles) to test an agent's robustness. Some research platforms (like Facebook's **Habitat** for embodied AI) support commanding the agent via natural language in the simulator as a form of testing instruction-following [6]. This effectively turns the simulator into an interactive playground for both AI and human. Such features blur into the next category (3D environments) when the interaction becomes immersive, but in research practice it's often via a keyboard/mouse controlling elements of the sim in real-time.

- **Result Analysis Dashboards:** After running many simulation episodes (as is common in training AI), making sense of aggregate results is important. Modern approaches include dashboards (using tools like TensorBoard, Weights & Biases, or bespoke UIs) that show training progress, compare performance across different simulation conditions, and even replay specific episodes of interest. These dashboards might let a researcher filter by scenario ("show me cases where the agent failed to reach the goal") and then watch those replays to diagnose issues. The design of these interfaces is akin to debugging tools and often incorporates interactive charts, timeline scrubbers for videos, and linked views (clicking a spike in a reward graph might open the corresponding simulation replay). While primarily for developers, these tools significantly affect how effectively one can iterate on AI models using simulation.

A concrete example of state-of-art practice is **Facebook (Meta) AI's Habitat simulator** for training embodied agents in photorealistic 3D environments [37] [38]. Habitat provides a *fleet* of tools: a high-performance 3D sim (Habitat-Sim), a Python API (Habitat-Lab) for defining tasks and agents, and even an interactive web-based challenge evaluation platform. The user (AI researcher) typically writes code to initialize an environment (like a Replica apartment), places a virtual robot, and then either manually controls it or lets an AI policy control it. They can visualize the robot's camera input, depth sensor, etc., in real-time windows. Habitat emphasizes speed (thousands of FPS in simulation) and also has logging to record episodes for later playback [39]. Designers of Habitat recognized the HCI importance of speed and repeatability: simulation enables running many experiments fast, which they explicitly contrast with real-world robotics being *slow, dangerous, expensive, and hard to reproduce* – simulation runs faster-than-real-time, with easy reset and parallelization, making it safe and cheap [40] [38]. These advantages need to be conveyed to users through the interface (e.g., showing when the sim is running faster than real time, or how many parallel instances are active).

Tools and Platforms Enabling Interaction

- **OpenAI Gym/Gymnasium:** A lightweight toolkit that became a standard for RL research. It defines a simple interface (`env.reset()`, `env.step(action) -> observation`) that abstracts the simulator's complexity. While not a UI per se, Gym enables **plug-and-play** experimentation with many environments and includes some basic rendering. It's often paired with custom visualization scripts. Gym's simplicity made it easy to integrate into notebooks and training loops, influencing how researchers interact with simulations (mostly through code). Its *successor*, Gymnasium, continues this approach.
- **Unity ML-Agents:** As mentioned, it turns the Unity game engine into a simulation platform for AI. Unity ML-Agents provides both a Python API and a Unity Editor integration. It comes with example environments and an **in-editor inspector** that shows agent observations in real-time. Importantly, Unity also offers a visual Behavior Parameters panel where developers can configure an agent's observation space and actions without code (somewhat UI-driven configuration). The platform also supports **Curriculum Learning** via a GUI, where one can schedule increasing difficulty in the simulation scenarios. Unity's reach into the game dev community also means better **3D asset and level design tools** can be leveraged for AI simulation design.
- **CARLA Simulator:** An open-source autonomous driving simulator built on Unreal Engine [41]. CARLA provides a rich API to spawn vehicles, pedestrians, set weather, etc., but also has a visual client so users can drive around or watch AI-driven cars in a virtual city. It is used by both researchers and industry to test self-driving algorithms. CARLA's interface allows multiple clients controlling different actors, reflecting a design for *scalability and multi-user control* (e.g., one client could control traffic lights, another the ego vehicle) [42]. For interaction, CARLA often is used with a manual driving mode

(through gamepad or wheel) to collect human demonstration data. As a tool, it emphasizes an **open digital asset library (maps, vehicles)** that users can freely use to construct scenarios [43] – lowering the content creation barrier. It also integrates with **ROS** (Robot Operating System) [44], meaning developers can use familiar robotics tools to interact (e.g., rviz to visualize sensor data while the sim runs). These integrations are a form of interaction design: meeting users in the tools they already use.

- **Logical/Abstract Simulators:** For more abstract simulations (like puzzle solving or logical planning), there are platforms such as **AI Playground** or bespoke UIs for things like the Blocks World or Gridworld. These often have web interfaces: e.g., a gridworld simulator might let a user place walls and rewards on a grid via a GUI then run various planning algorithms to watch how the agent behaves. While less publicized, such tools exist in research for demonstrating concepts (for instance, an online Blocks World demo where one can drag blocks and then see a planner come up with moves). **Visualization of search trees** or state graphs is a key interaction aspect for logic simulations. Tools like **WebGL-based tree viewers** let users expand nodes of a game tree to see what moves the AI considered. This makes the internal **search process** (a type of mental simulation the AI does) transparent to some extent.
- **Benchmark Suites and Competitions:** Platforms like **EvalAI** and others host simulation-based challenges. These provide participant interfaces to submit agents and get feedback on performance in simulations. For example, the Habitat Challenge on navigation or manipulation gives a leaderboard and visualizations of agent trajectories in unseen environments [45]. While not an "interface" in the classical sense for directly manipulating sims, these platforms present results and replays in a digestible way to users (often with web-based 3D viewers). They also push forward standardization of simulation interactions through common file formats and protocols (e.g., the **Model Context Protocol (MCP)** mentioned in some LLM-based agent research [46], which standardizes how an agent can call simulation tools via an API). In essence, these tools make interacting with complex simulations more **consistent** and collaborative (multiple teams comparing results).

Emerging Design Patterns and Paradigms

- **Photorealism and High-Fidelity Simulations:** One clear trend is making simulations more realistic to reduce the sim-to-real gap [47] [48]. Interaction-wise, this means simulators are incorporating real-world data (scans of environments, realistic physics). For example, Meta's **Replica** dataset captures real rooms in photorealistic detail, which can be loaded into Habitat [49] [48]. The paradigm shift here is treating simulation not just as a cartoon proxy but as a *digital twin* of reality (overlap with the next section). This enables new interactions: a user might virtually walk through a copy of a real space to train the AI, or adjust lighting conditions to see how the AI's perception holds up. For HCI, the closer the simulation is to reality, the easier it is for humans to interact naturally with it (since they can apply real-world knowledge). The pattern is **"bring the simulator to the user's world"** – e.g., use AR to overlay the simulation on the real environment for comparison, or VR to immerse a human in the sim as if it's real.
- **Standardized Tool Use Interfaces:** As AI agents get more complex, they not only act *in* simulations but use simulators as tools for planning. An emerging pattern is creating **interfaces for AI to use simulators** autonomously. For instance, a recent benchmark provides a Blocksworld simulation with an API (MCP) so an LLM-based agent can query the state or try actions in the simulated world as part of its reasoning [46]. This effectively turns the simulator into a sandbox for the AI's "imagination." The design paradigm is analogous to a human using a flight simulator to test a maneuver before doing it in a real plane; here the AI agent can simulate outcomes before committing to them. Interaction

design in this context means ensuring the results of the AI's internal simulations can be communicated when needed. If the AI simulates 5 possible plans, it might present the user with the top two, including a little animation or narrative of each simulated plan's outcome. That way, the user can choose which plan to execute in reality. This pattern of **simulation-based decision support** is likely to grow.

- **Human-in-the-Loop Simulation Training:** We see paradigms where humans and sims interact to train AI faster. One example is *interactive reward shaping*: a human watches the simulation and can tap a button to give the agent feedback (positive or negative) in real time, guiding its learning. Designing a simple interface for this (like pressing keyboard arrows to indicate "good" or "bad" as you watch an agent navigate a maze) can significantly reduce training time. Another example is **demonstration and correction**: if the agent does something wrong in sim, a human can take over and demonstrate the correct behavior, which the agent then learns from. This requires the simulation interface to seamlessly switch between AI control and human control. The pattern is treating simulation not just as a test environment but as a collaborative workshop where humans and AI co-create behavior. Some robotics researchers have even explored **VR interfaces** where a person in VR sees through the robot's sensors in the simulator and provides demonstrations by essentially "being" the robot for a moment – a very direct form of interaction to transfer human skills to the AI.
- **Scenario Libraries and Reusability:** As the community builds many simulations, there's an emerging paradigm of **modular scenario components**. Think of it like design patterns for simulations – e.g., a "pedestrian crossing street" module that can be inserted into any driving sim scenario. For interaction design, this means better libraries or marketplaces of simulation elements that users can drag and drop to construct new scenarios. If a QA engineer wants to test an autonomous car AI in a work zone scenario, they might grab a premade "roadwork with flagger" module and place it in the sim, rather than designing it from scratch. This modular approach is still developing, but when solidified, it will make simulators much more user-friendly and **accessible to non-experts** by leveraging community-created content.
- **Explainable Simulations:** When simulations are used for planning or for understanding AI decisions, another pattern is making the simulation itself explainable. For instance, if an AI does Monte Carlo rollouts (simulated futures), an interface might highlight the *critical event* in the simulation that led the AI to choose a certain plan ("I simulated five outcomes, and in four of them the road was blocked when taking Route A, so I chose Route B"). The simulation here acts as a visualization of *counterfactuals*. Design paradigms around this include slider-based manipulation of conditions (so users can ask "what if we changed this?" and the system reruns the sim under the hood) and clear annotation of simulation visuals with reasons (like overlaying text like "collision occurred here" on a sim replay). The goal is to use simulation to build user understanding of *why* the AI is doing something, by letting them literally see possible worlds.
- **Integration with Digital Twins and Real Data Streams:** A nascent paradigm is merging simulation with real-time data – for example, a digital twin city where the simulation is continually updated with sensor data from the real city. This is more covered in the next section, but for AI training and planning, it means the simulation can be interactive not just with the user but with reality itself. An AI could run what-if scenarios on a twin of a factory that's in operation, using live data as the starting state. The interface might allow an operator to fast-forward in sim to see the result of a change, then roll back. This coupling of simulation and reality in interfaces is emerging in fields like autonomous driving (testing new software on a digital twin of an actual road network during various times of day) and operations research (smart city dashboards that simulate traffic with current conditions). The challenge is presenting which parts are real vs simulated clearly to users, and ensuring the transitions are smooth.

HCI/UX Research and Insights

Interaction with simulations has been an interest in HCI, albeit somewhat niche compared to mainstream UI topics. A few relevant research and practice insights include:

- **Simulation for HCI Theory and Design:** Elizabeth Churchill and others have written about *What Simulation Can Do for HCI Research*. Simulations can be used to model and predict user behavior or to generate synthetic data for UI testing [50] . One takeaway is that simulation can augment the design process – e.g., simulating how users might navigate an interface or how different traffic patterns might affect commute experiences. While this is more about using simulation *as a tool* in design (rather than designing the simulation's UI), it underscores the importance of making simulators approachable to UX designers. Ideally, a UX researcher should be able to use a simulation environment to test hypotheses without needing deep coding skills. This drives the development of higher-level simulation authoring tools.
- **Usability of Developer Tools:** The UX of tools like Unity ML-Agents or CARLA has been studied informally in communities. Common feedback is that while powerful, these tools require juggling many parameters and have steep learning curves. Efforts from the companies/researchers involved have included writing extensive tutorials, creating default settings that work out-of-the-box, and providing visual debug info as part of the environment (for example, CARLA's debug camera view that shows a car's sensor vision is a direct usability feature). In academic venues, there have been workshops on "Simulation for Robotics" which often mention the need for better user interfaces to configure simulations.
- **Educational Simulators:** There's HCI work on simulators used for teaching, which often have very careful UI design to scaffold learning. For instance, a physics simulator for students might let them manipulate gravity with a slider and immediately see objects fall. These principles carry to AI simulations: if we want to *educate* stakeholders (like policymakers or students) about AI behavior, a well-designed simulator UI can be powerful. Researchers have found that interactivity (letting users try scenarios themselves) increases understanding and engagement. So, projects that aim to demystify AI often include interactive demos – for example, an interactive demo of a reinforcement learning agent where the user can tweak the reward function live and see how the agent's behavior changes. This is effectively HCI research on how to communicate AI concepts, using simulation as the medium.
- **Multi-Agent Simulation UX:** When multiple AI agents interact in simulation (say for swarm robotics or economic simulations), visualizing their coordination is tricky. HCI research in information visualization offers techniques (like using color coding for agent "teams", or trails showing paths taken). Some recent AI papers include videos with overhead views that highlight decision points for each agent. Designing an interface for humans to **steer** or intervene in multi-agent sims is also explored: e.g., an operator might adjust a goal for one agent and see how the group behavior shifts. Concepts from *orchestration interfaces* (as mentioned in agentic systems) apply here too, but in a spatial environment.
- **Wizard-of-Oz and Human Simulation:** Interestingly, a classical HCI method is the Wizard-of-Oz experiment, where a human secretly simulates the behavior of an AI to test an interface. In a sense, this is using a *human as the simulator* of intelligence. While not directly our focus, it underscores a point: sometimes the "simulation" in development is a proxy for a component not yet built. The design of such experiments (as mentioned in search results [51]) requires careful interface design so that users believe they're interacting with an autonomous system. This is somewhat inverse to our main theme, but it shows simulation (even if manual) is baked into the iterative design of AI

interactions. By simulating AI responses early, designers can foresee interaction issues and address them before the real AI is plugged in.

- **Continuous Evaluation and User Feedback Loops:** In operational AI systems that use simulation for planning (like logistics AIs that simulate supply chain scenarios), HCI researchers emphasize keeping the user in the loop of that simulation. For example, an AI might generate a plan by simulation, but a human manager should review that plan with the aid of the same simulation visuals. Research has shown that humans are more likely to trust and adopt AI suggestions if they can see a concrete simulation of the outcomes (compared to just being told "do X"). Therefore, designing interfaces that present simulated outcomes (rather than black-box recommendations) can improve human-AI decision-making. This aligns with the broader theme of **explainable AI** – simulations are a form of explanation when used correctly.

In summary, simulation-based AI work is pushing interaction design towards **rich visualizations, developer-centric UX improvements, and hybrid human-AI control interfaces**. The line between a "user" and a "developer" blurs here: often the direct users of simulators are AI engineers, but as AI systems trained in simulation reach end-users, their simulation origins must be communicated (e.g., an autonomous car's manual might need to explain "this car was trained in thousands of virtual scenarios"). The current state-of-the-art tools like Habitat, CARLA, Unity ML-Agents provide the building blocks, and emerging paradigms seek to make these simulations more **lifelike, interpretable, and interactive** for all stakeholders. As Edd Gent quipped, the goal is moving towards "ever more realistic digital environments" because the closer simulation gets to reality, the more seamlessly humans and AI can interact within or through them [47].

3D Virtual Environments and Digital Twins

Overview: This category deals with fully interactive 3D worlds – including virtual reality (VR), augmented reality (AR), and **digital twins** of real environments – which serve either as platforms for AI to operate in, or as user interfaces themselves enriched by AI. These environments differ from the general simulators above in that they are typically **immersive or highly visual**, and often involve end-users navigating or manipulating the environment directly. Examples include virtual worlds like multiplayer VR spaces (some call them "the Metaverse"), AR-enabled physical spaces where digital content is overlaid, and digital twin models of factories or cities used for monitoring and decision-making. For AI, such environments can be both training grounds and deployment platforms (e.g., an AI NPC in a VR game, or an AI that helps manage a virtual factory twin). Interaction design here spans spatial interface design, multimodal inputs (since in 3D space users may use gestures, gaze, motion), and metaphorical design to make virtual interactions intuitive.

Key Challenges in Interaction Design

- **Spatial Interaction and User Comfort:** In VR or AR, users are effectively **inside** the UI. This raises challenges of ergonomics and **presence** – interfaces must be designed to avoid motion sickness, fatigue, and cognitive overload in 3D. Traditional 2D UI elements (buttons, menus) need rethinking for spatial contexts. For example, in a digital twin of a building, how does a user select an object or bring up information? Maybe using gaze or by walking up to it. Ensuring these interactions feel natural is tricky. There is also the issue of **scale**: a digital twin city might be huge – do users fly above it God-view style, or is it scaled down on a tabletop AR display? Choosing appropriate metaphors (e.g., **"world-in-miniature" interfaces** where users manipulate a small replica to affect the big

world) is an HCI challenge. Balancing immersion with usability is key – too much realism can hamper efficiency (e.g., walking everywhere in VR vs. using teleportation for navigation).

- **Representing and Updating World State:** In a dynamic environment (virtual or twin), the system's world model might change rapidly (sensors updating, AI agents moving things, etc.). The UI must convey these changes clearly. In an industrial digital twin, for instance, if a machine in the real world goes down, the twin should reflect that (maybe a color change or smoke effect on the 3D model). The challenge is **avoiding information overload** – a complex factory twin could have thousands of data points (temperatures, speeds, etc.). Showing all in 3D could clutter the view. So designers use techniques like selective visualization (only show anomalies or what's relevant to the user's current context) and AR annotations that appear in context when needed. Keeping the virtual world **in sync** with reality (for digital twins) is also a technical challenge, but from UX side it means sometimes the virtual might lag or need a refresh, which can confuse users if not indicated properly.
- **Multi-User Interaction and Social Presence:** Many 3D environments are collaborative – multiple users (or users and AI avatars) share the space. Designing for social interaction in VR/AR is a challenge: ensuring people can communicate, have appropriate personal space (no virtual crowding discomfort), and have awareness of others' actions. If AI agents (NPCs or assistants) are also present as avatars, there's the challenge of making their behavior and intent understandable to human users. For example, a virtual assistant avatar in AR helping a technician might point at equipment or use gaze to direct the user's attention – these cues must be designed to be noticeable but not startling. If two people are collaborating with a digital twin, how do they reference parts of the model? Often systems provide virtual laser pointers or shared markers in the environment to aid communication. Achieving a sense of **co-presence** (feeling like others are there with you) while also providing UI affordances is an ongoing research area.
- **Device and Context Constraints:** Interaction design must account for the devices used: VR headsets, AR glasses, large 3D display walls, etc., each have constraints. AR glasses like HoloLens have limited field of view and use gestures + voice for input, which means UIs must be simplified and robust to recognition errors. VR headsets occlude the real world, so users can't see their keyboard – interfaces rely on motion controllers or hand tracking. These constraints push designers toward **gesture vocabularies, voice commands, and spatial menus**. Ensuring these inputs are discoverable and have low learning curve is challenging – e.g., how does a new user learn that in this virtual environment, a two-finger tap in the air opens the settings menu? Tooltips in mid-air? Guided tutorials? Those need to be built in, likely with multimodal guidance (visual highlight + audio instruction).
- **Blending Physical and Virtual (for AR):** In AR and some digital twin scenarios, virtual content coexists with physical reality. A big challenge is **alignment and occlusion** – virtual objects must appear anchored to real ones accurately, and obey occlusion (e.g., a virtual avatar shouldn't walk through a physical couch). If alignment is off, the illusion breaks and interactions suffer (imagine trying to press a virtual button that appears slightly to the side of the actual device). One of the humorous yet important examples: ensuring a 3D virtual avatar of your grandma doesn't "keep walking through the back of the couch" when chatting with you in AR [52] . That example highlights both a technical and interaction point: the system's world model of the physical environment must be good enough to prevent such unnatural behavior, thereby maintaining the user's **spatial trust**. Users need to trust that virtual elements will behave with respect to real-world constraints.

Current Interaction Approaches & Interfaces

- **Immersive VR Interfaces:** In fully virtual environments, interfaces often leverage **natural motions**. For example, instead of clicking a delete button, a user might grab a virtual object and throw it away.

Many VR applications use **virtual hands** that mimic the user's hand movements (via controllers or hand-tracking). UI panels in VR are sometimes presented as floating screens or wrist-mounted menus (so you flip your palm to see a menu). Another approach is diegetic interfaces – UI that is part of the scene (like a control panel on a virtual machine that you press buttons on). For movement, teleportation via pointing has become a standard to reduce motion sickness. Social VR platforms (Altspace, VRChat, etc.) have interfaces for gestures (some map real hand gestures, others trigger via buttons e.g. wave, thumbs-up emotes) to facilitate social presence.

- **Augmented Reality and Digital Twin Dashboards:** In AR, interfaces commonly overlay information on or next to physical objects. For instance, in an AR-maintenance scenario, looking at a machine might trigger labels to appear on its parts (with data like temperature or an arrow pointing to a faulty component). Interaction might involve voice ("Highlight the pressure valve") or gaze selection (gazing at a menu option for a second to activate it). Hand gestures like air-tap (on HoloLens) or pinching with fingers (on mobile AR via screen) are used to select or drag virtual elements. Many digital twin control rooms use large screens or AR tables where the twin is displayed; users then use touch or gestures on that display to interact (e.g., pinch-zoom the model, rotate it, select assets). These interfaces strive to be **direct manipulation** in 3D – meaning the user feels like they are manipulating the actual objects (though virtually).
- **Spatial UI Metaphors:** A common approach is to leverage spatial metaphors that users find intuitive. For example, **"head-up display"** style info anchored to the user's view (like a compass or mini-map always at top). Or **room-based metaphors**: treating different rooms or areas in VR as different "screens"/contexts. In some VR design tools, one might physically walk to a different station to switch modes (like going to a paint station vs. an assembly station in a VR workshop). While this is immersive, it can be inefficient, so teleports or quick UI shortcuts exist too. Another metaphor is **holographic windows** – e.g., you could summon a browser window in VR that floats in front of you for reference, then dismiss it. This mixes 2D and 3D, which many systems allow (Windows Mixed Reality, etc., let flat app windows exist in the 3D space).
- **Multi-Modal Interaction (Voice, Gesture, Gaze):** In 3D environments, using a combination of inputs can enhance usability. Voice commands are powerful for tasks like "zoom in on Building A" or "show pressure graph over time." They free the hands and are intuitive for many commands (especially if the user's gaze already indicates the context, like gazing at a machine and saying "show details"). Gesture is used for more continuous controls (rotating a virtual knob with your hand, or drawing in the air). Gaze is often implicitly used for targeting: many AR interfaces do "gaze & commit" where you gaze at something and then do a simple gesture or voice confirmation to select it. Current devices like Magic Leap and HoloLens support this combination. It reduces the need for controllers or having to precisely pinch tiny UI elements.
- **Collaborative and Social Interfaces:** For multi-user 3D interactions, approaches include avatars with expressive features (mouth movement for speech, hand gestures, even eye contact if hardware supports eye-tracking). Some enterprise AR solutions provide **shared annotations** – e.g., one user draws a highlight around a part, and that appears in the other user's view, anchored correctly. In VR meetings, virtual whiteboards or sticky notes are common tools to aid collaboration, providing familiar metaphors in the new medium. From a UI perspective, giving users awareness is crucial: icons or indicators for where others are in the space (if out of view), or subtle sounds when someone "enters" the virtual room.
- **Digital Twin Dashboards:** Many digital twins – say of a smart building or factory – are used via desktop interfaces as well, not exclusively VR/AR. These are 3D visualizations on screen with interactive elements. Current interfaces often show a 3D model alongside traditional charts and controls. For example, an operator might see a 3D layout of a factory, and clicking a machine in the 3D view brings up a panel of its live metrics. Some platforms (like the Simio + NVIDIA Omniverse

integration) allow not just viewing but interacting: users can *modify parameters in real-time and immediately see the visual impact in the twin* [53] . This tight coupling of control and visualization is a key benefit of digital twin UIs – it accelerates understanding and optimization by providing instant visual feedback to user actions. The UI pattern is often a **split view**: 3D scene + GUI controls, or AR view + handheld device controls for precision tasks (since selecting things in AR in mid-air can be less precise than on a touch screen, some systems use a hybrid approach where you point with AR and fine-tune via phone app).

Example Scenario: Consider a **smart factory digital twin** used by an operator through a desktop and optional AR headset. The desktop interface shows a 3D model of the factory floor. Machines are color-coded by status (green running, red fault, yellow maintenance due). The operator can rotate and zoom the model, or switch to a specific zone view. On the side is a timeline slider to scrub through past data or run a simulation into the future. If an anomaly occurs (say machine X temperature spikes), the twin highlights it (e.g., a glowing outline) and might even animate a virtual alert icon above it. The operator can click it to get a tooltip with details, or enter an **immersive mode**: wearing AR glasses, they walk to the real machine and see an overlay pointing to the overheated component and instructions generated by an AI assistant. The AR interface allows the operator to say "show recommended fix" – the AI then overlays a step-by-step hologram (like arrows showing which valve to turn). Meanwhile, remote colleagues in a central control room see the operator's feed in the twin and can drop virtual markers or messages ("check this panel first") that appear in the operator's AR view. This scenario combines multiple interfaces and modalities: desktop 3D UI, AR heads-up guidance, voice commands, and collaborative annotation – illustrating how design paradigms merge in practice.

Tools and Platforms

- **Game Engines (Unity, Unreal) with XR Support:** The leading platforms to create 3D UI and worlds are game engines. Unity and Unreal both have extensive support for VR and AR (Unity's XR Interaction Toolkit, Unreal's AR Framework, etc.). These provide base components like spatial UI elements, teleport locomotion, hand physics, and integration with hardware (VR headsets, AR SDKs). Designers use these engines to build custom experiences. For instance, **Unreal Engine** was used to create Epic's "The Matrix Awakens" city demo, which doubles as a sort of digital twin of a city that AI agents can roam. Unity is popular in AR apps like product visualizations and has been used for digital twin dashboards (with the help of libraries to connect real-time data).
- **NVIDIA Omniverse:** This is an emerging platform specifically aimed at building digital twins and collaborative 3D environments. Omniverse allows connecting various 3D tools (using the USD format) and supports real-time physics and multi-user collaboration. For example, BMW used Omniverse to create a factory digital twin. Interaction-wise, Omniverse provides a canvas where developers can script interactivity or connect AI models. It's geared towards both visualization and simulation. One notable aspect is its focus on photorealism with real-time ray tracing – important for true-to-life twins (e.g., accurate lighting in a twin can matter for an AI that does visual tasks). Omniverse is also integrating AI in the pipeline – e.g., AI to generate 3D assets or behaviors. For user interaction, companies build apps on top of Omniverse: e.g., a maintenance training VR app pulling data from the twin. Omniverse's ability to handle massive models (entire factories, cities) and maintain **live sync** with data streams makes it a backbone for these advanced UIs.
- **AR Development Platforms:** Apple's **ARKit** and Google's **ARCore** enable a wide range of mobile AR apps, including those that can act as interfaces to world models. For instance, there are ARCore-based apps to visualize IoT data in a building by pointing your phone at the relevant equipment. Microsoft's **HoloLens** (with the MRTK toolkit) is used in enterprise for tasks like remote assistance – it

provides interface components like spatial buttons, tooltips that float anchored to objects, and voice command support. There are also specialized AR platforms like **PTC Vuforia** for industrial AR, which integrate with CAD models (digital twin data) and provide tracking of specific objects (so you can anchor instructions to a particular machine part reliably).

- **Metaverse and Social VR Platforms:** While the consumer "metaverse" hype has cooled, platforms like **VRChat, AltspaceVR (now closed), Rec Room, Mozilla Hubs** etc., have pioneered social and interaction paradigms in 3D. They often supply building tools to users (so-called UGC platforms). For interaction, they handle netcode for multi-user, avatar systems, and basic UI building blocks (e.g., VRChat has pickup objects, UI panels for user settings, etc.). These can be leveraged for other purposes: e.g., Mozilla Hubs (an open web-based VR space) has been used to host data visualizations in a collaborative VR setting. The knowledge from these platforms (how to reduce "latency" in conversation via spatial audio, or how to avoid users accidentally walking through each other by having collision on avatars, etc.) informs design of any multi-user 3D environment.
- **Specialized Digital Twin Suites:** Aside from Omniverse, companies like Siemens, GE, and others have their own digital twin and VR training suites. For example, **Siemens Tecnomatix** and **Process Simulate** allow 3D modeling of factories and provide VR modes for validation. These come with UI for engineers to, say, simulate a worker's reach in a workstation and see if there are ergonomic issues – often visualized with an anthropomorphic avatar and heatmaps of reach. The interfaces are professional tools and sometimes clunky, but they are evolving to be more interactive and real-time. There are also IoT platforms (like Azure Digital Twins, IBM Maximo) that output data to visualization modules. They might not define the UI fully but provide APIs to query twin state that UX designers can use to build custom dashboards or AR visualizations.
- **Hardware and Devices:** On the user's end, the quality of experience is tied to hardware. The latest VR headsets (e.g., Meta Quest 3, Valve Index) offer better resolution, passthrough cameras for mixed reality, and more natural interaction (hand tracking). The upcoming AR/MR devices (like Apple Vision Pro) are touting very high fidelity spatial computing with a new UI paradigm (Apple's "spatial computing" with eye tracking + hand pinch for selection [54]). If Vision Pro's design becomes popular, we may see a standardization of AR interaction (e.g., using eye gaze for cursor and pinch to click on any element in space – essentially a gaze pointer). This would influence cross-platform design: spatial apps might adopt conventions similar to how iPhone standardized multi-touch gestures.
- **Haptic and Peripheral Tools:** There are also tools like haptic gloves or suits that can be used in 3D environments for more immersive interaction. While not mainstream yet, some training simulations use haptic feedback (e.g., feeling resistance when you press a virtual button). For digital twins of product prototypes, designers sometimes use force-feedback arms to "feel" a CAD model. These peripherals demand specialized interface support – the system must generate correct forces or vibrations to match visual events. It's a niche area but important in certain contexts (surgical simulators use haptic interfaces heavily to simulate touch and force).

Emerging Design Patterns and Paradigms

- **Blending Real and Virtual – Seamless Mixed Reality:** A paradigm gaining ground is one where the boundary between physical and digital is fluid. For example, **location-based VR** experiences integrate real physical walls and props that exactly match the virtual environment (to provide tactile feedback and unrestricted movement) [55] [56]. In such cases, the design pattern is **1:1 mapping** – a digital twin of the physical space used for VR so that you can reach out and touch a virtual object and there's a real object there. This requires precise tracking and alignment, but when achieved, it yields powerful immersion and intuitive interaction (real walking, real object manipulation). The work by Zhang et al. (2025) on a dual-realm VR prototype is an example: they synchronized physical and

virtual so that tangible interaction, social gestures, and even social touch (like handshakes) were possible between co-located users and physical props [55] [56]. This paradigm suggests future interfaces where, say, you go to a conference room and wear AR glasses – the table in front of you is both real and the canvas for a virtual meeting with remote colleagues' avatars sitting around it. The **"digital twin embodied interaction"** concept [57] extends the twin to include human avatars and physical actions, which opens new design territory: e.g., how to maintain **spatial trust** so users feel confident the virtual and physical align (Zhang et al. found that precise alignment and embodied cues significantly enhance immersion and social connectedness [58] [59]).

- **Spatial Analytics and Data Visualization:** As data becomes huge, a paradigm is emerging to use VR/AR for data visualization – known as **immersive analytics**. Instead of viewing complex 3D data (like fluid dynamics or IoT sensor networks) on 2D screens, analysts can step into a 3D visualization or see it overlaid on the environment. Patterns might become more apparent spatially. The design challenge and pattern here is representing abstract data in spatial forms. It might involve mapping data dimensions to spatial dimensions (like a scatter plot but in 3D space you walk through). AI can assist by highlighting or calling out findings in AR ("this area has anomalies"). The interface might allow the user to **"fly" through data** or have a miniature data landscape they can scale up and walk around. Some early research shows people can glean insights faster with certain 3D visual metaphors, but also that it's easy to get lost – hence designs include mini-maps, teleport shortcuts, and synchronized 2D/3D views to keep one oriented. With devices like Vision Pro, even traditional analytics software (e.g., Tableau) is exploring spatial versions [60].
- **AI Avatars and Assistants in 3D:** With generative AI, we are seeing more **intelligent virtual characters** and assistants populating these environments. The generative agents example we discussed could well be ported into a game or a metaverse world, making NPCs far more lifelike. In an enterprise AR context, an AI assistant might appear as a virtual avatar coworker who can guide you. The design paradigm here is giving these AI agents a **human-friendly embodiment or representation** that leverages the 3D medium. This might mean an avatar that uses hand gestures to point things out (rather than a disembodied voice saying "look there"). It could also mean ambient AI presence, like a drone or virtual pet that roams the environment as the AI's persona. Designing such embodiments requires careful consideration: too human-like and users might have uncanny valley or misplaced trust; too abstract and it might not effectively communicate. Some systems use simple mascots or icons (like a floating info icon that you can summon in AR which then talks in a speech bubble). Others use full humanoid avatars for AI – for instance, an AI service technician avatar in a factory twin that "walks" to a machine to indicate attention. As these become common, patterns will emerge on how users prefer to interact (do they want to talk to a face versus just see highlights? etc.). Early studies (like in the generative agents work) show people often find the world more engaging when AI characters behave believably social [28] [61], so leaning into spatial, social cues can increase acceptance.
- **Natural Gestural Language & Spatial Interaction Standards:** We might be approaching something like the "WIMP" (Windows, Icons, Menus, Pointer) paradigm but for spatial computing. Apple's visionOS is pushing one approach (gaze + pinch as universal select, and 3D app windows around you). Other platforms like HoloLens have bloom gestures (to open main menu) etc. Eventually, through cross-pollination, a common set of **spatial interaction idioms** may form – e.g., pinch to select, air-tap to click, grab-and-pull to bring an object closer, swipe in air to scroll, etc. Already, many VR apps use trigger pull on controller as primary select and grip button to grab/throw, which is becoming standard. Once users internalize these, designers can build on them. The paradigm shift is akin to when multi-touch became mainstream – pinch-to-zoom is second nature now. We may soon say the same for certain AR gestures. Having these standards will greatly ease

design, as one can rely on platform conventions. It will also allow more focus on higher-level UX concerns (instead of inventing new gestures).

- **Persistent Shared Worlds:** Another pattern is the idea of persistent virtual spaces that multiple people and AIs can contribute to over time – essentially a *digital world* that parallels the real one. An example is a digital twin of a city that isn't just for one-off simulation, but continuously used by city officials, citizens (in a public-facing way), and AI systems that manage infrastructure. Interaction design here becomes almost like urban design: you need "digital public spaces" where people can gather (virtually) to discuss a new building project with the 3D model right there, or AR information layers accessible to anyone visiting a location (like AR street signs or historical markers). We see early signs of this with AR apps that leave persistent AR graffiti or notes in places. For AI, if a city's traffic AI uses the twin to try new signal timings, a traffic engineer might join it in VR to watch a rush-hour simulation together. The pattern is **collaborative governance via a virtual space**. This raises design/UX questions around permissions (who can change what in the shared twin), version control (seeing changes over time), and blending asynchronous and synchronous collaboration. It's an ambitious paradigm but one that could redefine civic and enterprise workflows.
- **Human-Centered Digital Twins:** Research (e.g., a 2024 systematic review by Barricelli et al.) suggests that to be effective, **human-centered design principles** must be applied to digital twins [62]. This means making them **intuitive, transparent, and cognitively manageable** representations of data. We see paradigms focusing on *user-centric views*: a maintenance worker vs. a manager might have different twin interfaces tuned to their tasks. UX designers are creating **persona-based twin dashboards** so each stakeholder sees the right slice of the world model. Another aspect is **explainability** in twins – if an AI in the loop of a twin makes a recommendation (e.g., reroute power in a grid), the twin's UI should highlight relevant factors (like a line in the grid model flashing where an overload is predicted) to justify the recommendation. Essentially, merging the AI's reasoning with the spatial visualization to create self-explanatory world models.

Relevant HCI/UX Research

- **Spatial UI & AR/VR Design Research:** The field of VR/AR interaction has a rich history in HCI. Early research gave us principles like **"minimize simulator sickness"** (via higher frame rates, limited acceleration), and techniques like **diegetic vs non-diegetic UI** (whether interface elements exist in the story world or as overlays). A lot of user studies compare input methods (e.g., is gaze selection faster than controller pointing? How accurate is hand pinch vs. click?). These inform best practices: for instance, that **gaze + dwell selection** can be effective but may strain eyes if used too much, so mix it with voice or hand inputs. We also know that **feedback** is crucial in 3D: because there's less tactile confirmation, visual or haptic feedback for actions (like a button press animation or controller vibration) helps users feel their action registered.
- **Embodied Avatars and Social Presence:** Research like the Frontiers article we cited [57] [56] delves into how **embodiment** (full-body avatars, haptic feedback, etc.) affects social interaction quality. The findings that alignment and embodied cues boost immersion and connectedness [58] are essentially UX evidence that investing in realistic interaction pays off in user satisfaction and trust in the experience. For enterprise, this could mean that if remote collaborators have full-body avatars, they might communicate better than with just voice or floating heads.
- **Trust in AR Guidance:** There have been studies on how users trust information in AR. For example, if an AR overlay says "Pipe behind this wall," do users trust it enough to drill? Factors include the AR's perceived accuracy, how the info is presented (does it show uncertainty?), and if there's a real-world backup (like a slight outline of the pipe through the wall to cue the user). This crosses into safety – designing AR such that errors are minimized and not catastrophic. UX research suggests using

redundant cues (visual + text + maybe sound) for critical info, and providing **means to verify** (like "scan" modes that use sensors to double-check before action).

- **Digital Twin User Studies:** Because digital twins are relatively new in HCI, formal user studies are fewer, but some exist. For instance, a human factors study might test how quickly operators respond to an alert using a 3D twin vs. a traditional 2D control panel [63]. Anecdotally, companies report that 3D twins improve cross-disciplinary communication – e.g., facility managers and IT and logistics can all *see* the system and discuss on equal footing [64] [65]. Researchers are starting to call for **evaluations of cognitive load** when using twins, as too much detail might overwhelm. The idea of a **human digital twin** (modeling a human user inside a twin) is also emerging, which could enable personalized interfaces (the system predicts what information a particular user will need based on a model of their behavior).
- **Ethnographic Studies in VR Work:** Some HCI researchers have observed how people actually work together in VR or use AR on the job. These studies uncover mismatches – e.g., a study might find that workers still resort to pen and paper in AR because the interface to input text was clunky. Or that people in AR meetings miss the peripheral awareness of others they have in real meetings (so now apps introduce subtle avatar "body language" indicators). By understanding these, designers refine the environment. A big theme is that **new social norms** form in virtual spaces (like how long to mute, how to indicate you want to speak in a virtual meeting). UX design can assist by providing features (like a "raise hand" button in VR or a status bubble that appears over silent users).
- **Privacy and Ethical Design:** AR especially raises privacy issues – recording in public, scanning people, etc. UX designs are implementing **privacy indicators** (e.g., a light showing when AR glasses are recording, or blurring bystanders). Likewise, in shared virtual worlds, **content moderation** and **safety** are critical – there have been reports of harassment in social VR. Designing preventative measures (personal safe zones, the ability to mute/block others, or AI moderators that flag abuse) is an important aspect of UX for 3D spaces. This is supported by HCI research into online behavior and safety, now applied to embodied contexts. For AI agents in these spaces, ethics includes not misrepresenting AI as human (some suggest agents should have clear indicators they are AI to avoid deception in social contexts). These considerations ensure the "world models" remain beneficial and users feel secure using them.

In conclusion, 3D virtual environments and digital twins represent a frontier where **interaction design becomes immersive and spatial**. Innovative UI patterns like embodied interaction with aligned physical feedback [58], multimodal commands, and collaborative virtual spaces are rapidly evolving. The common thread across these trends is creating a seamless interface between the user and a rich world model – whether that world is a fictional VR scenario or a mirror of real-world infrastructure. By leveraging human spatial cognition and social instincts, and addressing the new challenges that immersion brings, designers aim to make these AI-powered worlds **natural, effective, and even enjoyable** for users. The research and early deployments so far show great promise: when done right, users often report higher engagement, better understanding of complex data, and a feeling of genuine presence and collaboration [58] [66]. As technology like AR glasses and AI continues to advance, we can expect these paradigms to mature into everyday interactions – effectively making the **world** (physical or virtual) the interface.

Conclusion

Across these three domains – agentic AI, simulation-based AI, and 3D virtual worlds – we see a paradigm shift in interaction design. Traditional GUI paradigms are expanding into **conversations, simulations, and spatial experiences**. Users will increasingly interact with AI through dialogues and collaborations, not just

button clicks. They will test ideas and understand AI decisions via simulations and visual sandboxes rather than static charts. They will engage with digital representations of environments, where information and controls are embedded in the space around them.

Despite their differences, all these paradigms share common goals: enhancing **understandability** of complex AI systems, maintaining user **agency and trust**, and leveraging intuitive human skills (like spatial navigation or natural language) to make interactions richer. There are clear challenges – from ensuring transparency in autonomous agents to preventing overload in immersive analytics – but research and practice are yielding creative solutions. Design patterns like **intention-based agent UIs, mixed-initiative control, multimodal spatial input,** and **digital twin visualizations** are converging into a new design toolkit for AI systems.

We are still in the early days of standardizing these interactions. It's an exciting, experimental era: HCI researchers, UX designers, and AI developers are working hand-in-hand (often quite literally, as hand tracking in VR might show!) to figure out what works best for users. What's clear is that as AI world models become more capable and prevalent, human-centric design is more crucial than ever. The most powerful AI is of little use if humans cannot intuitively direct it or understand it. Conversely, with thoughtful interaction design, AI's capabilities can be *amplified* – empowering users to achieve goals in ways that feel almost magical, whether by conversing with a helpful agent, simulating a scenario with a few clicks, or walking through a virtual environment that brings data to life.

In summary, emerging interaction paradigms for AI world models are making AI more accessible and actionable. They invite users into the loop of AI's understanding – be it through a chat, a simulated trial, or an immersive world – and in doing so, they align the system's "world model" with the human user's mental model. This alignment is ultimately the key to effective, trustable AI. As these paradigms continue to develop, backed by ongoing HCI research and real-world testing, we can look forward to AI systems that not only *know* more about the world, but also *communicate and collaborate* in tune with human ways of knowing.

Sources:

- Schömbs, S. et al. (2025). *From Conversation to Orchestration: HCI Challenges and Opportunities in Interactive Multi-Agentic Systems*. (Identifies issues like agent orchestration, conflict resolution in multi-agent UX) [4] [67].
- Medeiros, I. (2025). *Flows in the Age of Agentic AI: Our Core UX Models No Longer Apply?*. (Discusses shifting from step-by-step UI flows to intention-focused design and trust principles for agentic systems) [19] [2].
- Dibia, V. (2025). *4 UX Design Principles for Autonomous Multi-Agent AI Systems*. (Key principles: capability discovery, observability/provenance, interruptibility, cost-aware delegation) [24].
- Park, J.S. et al. (2023). *Generative Agents: Interactive Simulacra of Human Behavior*. UIST '23. (Introduces architecture for believable AI agents with memories, in a sandbox world; notes emergent social behavior) [13] [12].
- Stanford HAI News (2023). *Computational Agents Exhibit Believable Humanlike Behavior*. (Press article summarizing Generative Agents results; example of the Valentine's Day party scenario) [66] [68].
- LLM Watch Newsletter (2025). *AI Agents of the Week*. (Highlights developments like Blocksworld Planning Benchmark with simulation for LLM agents, and NavForesee coupling planning with a predictive world model for navigation) [46] [69].

- Habitat Project (Meta AI) – AI Habitat website and paper. (Describes a 3D simulator for embodied AI, emphasizing photorealism and speed; lists benefits of simulation vs real world for training) [37] [38].
- Gent, E. (2019). *How Facebook's "Mirror World" Will Help Train AI*. SingularityHub. (Explains use of ultra-realistic Replica environments in Habitat; discusses sim-to-real gap and AR alignment issues) [36] [52].
- Zhang, J. et al. (2025). *Digital twin embodied interactions design: Synchronized and aligned physical sensation in location-based social VR*. Frontiers in VR. (Demonstrates integrating full-body avatars and passive haptics with a physical space twin; defines categories of cross-realm interactions like tangible, social gesture, social touch) [57] [56].
- New America OTI (2025). *AI Agents and Memory: Privacy and Power in the Model Context Protocol Era*. (Background on how LLM-based agents use tools via MCP, and need for systems to remain understandable and aligned) [70] [71].
- Simio (2023). *Harness the Power of NVIDIA Omniverse Digital Twins*. (Industry perspective on benefits of 3D digital twin interfaces: immersive visualization, real-time parameter tweaking, engaging stakeholders) [53] [72].
- Apple Developer (2023). *Designing for visionOS*. (Guidance reflecting Apple's spatial computing UI paradigm using eye tracking and hand gestures – indicating future standards in AR interfaces) [54].

---

[1] [2] [19] [20] [21] [22] Flows in the Age of Agentic AI: Our Core UX Models No Longer Apply? » { design@tive } information design
https://www.designative.info/2025/11/20/flows-age-agentic-ai-what-if-our-core-ux-models-no-longer-apply/

[3] [7] [17] [18] [34] [35] [46] [69] AI Agents of the Week: Papers You Should Know About
https://www.llmwatch.com/p/ai-agents-of-the-week-papers-you-b5b

[4] [27] [67] [2506.20091] From Conversation to Orchestration: HCI Challenges and Opportunities in Interactive Multi-Agentic Systems
https://arxiv.org/abs/2506.20091

[5] [8] [9] [23] [24] [25] [26] 4 UX Design Principles for Autonomous Multi-Agent AI Systems
https://newsletter.victordibia.com/p/4-ux-design-principles-for-multi

[6] [37] [38] [39] [40] [45] AI Habitat
https://aihabitat.org/

[10] [28] [29] [2304.03442] Generative Agents: Interactive Simulacra of Human Behavior
https://arxiv.org/abs/2304.03442

[11] [12] [13] [2304.03442] Generative Agents: Interactive Simulacra of Human Behavior
https://ar5iv.labs.arxiv.org/html/2304.03442

[14] [15] [70] [71] AI Agents and Memory: Privacy and Power in the Model Context Protocol (MCP) Era
https://www.newamerica.org/oti/briefs/ai-agents-and-memory/

[16] [PDF] Development of an Augmented Reality Interface for Intuitive Robot ...
https://elib.dlr.de/200286/1/Master_Thesis_final_Version.pdf

[30] [31] [32] [33] [61] [66] [68] Computational Agents Exhibit Believable Humanlike Behavior | Stanford HAI
https://hai.stanford.edu/news/computational-agents-exhibit-believable-humanlike-behavior

[36] [47] [48] [49] [52] How Facebook's New 'Mirror World' Will Help Train AI
https://singularityhub.com/2019/06/25/how-facebooks-new-mirror-world-will-help-train-ai/

41 42 43 44 CARLA Simulator
https://carla.org/

50 [PDF] What Simulation Can Do for HCI Research
https://www.dcs.gla.ac.uk/~rod/publications/SimulationHCI22.pdf

51 Wizard of Oz Experiment - Learning Loop
https://learningloop.io/plays/wizard-of-oz

53 64 65 72 Harness the Power of NVIDIA Omniverse Digital Twins | Simio
https://www.simio.com/nvidia-omniverse-digital-twin-integration/

54 Spatial Computing with Apple Vision Pro - The Final Frontier
https://www.bottlerocketstudios.com/news-views/spatial-computing-with-apple-vision-pro-the-final-frontier/

55 56 57 58 59 Frontiers | Digital twin embodied interactions design: Synchronized and aligned physical sensation in location-based social VR
https://www.frontiersin.org/journals/virtual-reality/articles/10.3389/frvir.2025.1499845/full

60 Exploring Spatial Computing and Immersive Analytics with Vision Pro
https://www.tableau.com/blog/exploring-spatial-computing-and-immersive-analytics-vision-pro

62 Human-Centered Digital Twins | Springer Nature Link
https://link.springer.com/rwe/10.1007/978-981-97-8440-0_88-1

63 A Human Factors Approach to Design an Information Interface ...
https://www.sciencedirect.com/science/article/abs/pii/S2452414X2600004X