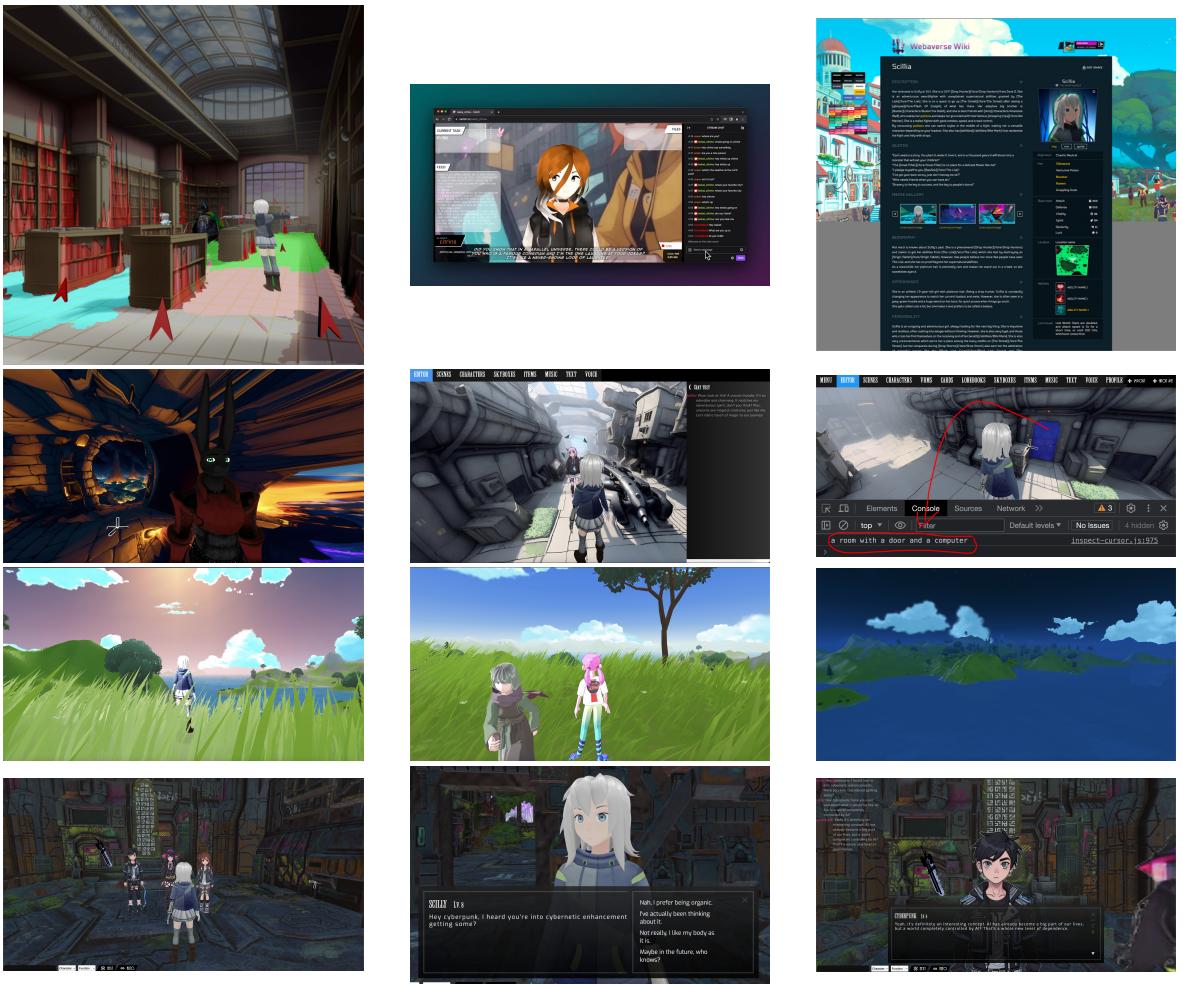
ChatWorld: embodied multi-agent simulation for extensible production of hypermedia

Adrian Biedrzycki
Upstreet.ai
ultimus@gmail.com

August 17, 2023



Contents

1 Abstract

Perception of the practical merge of humans and AI has shifted from theoretical to achievable, owing to the distribution of generative AI models like ChatGPT and Stable Diffusion. This has introduced serious concerns about how humans and AIs can cooperate among themselves and each other.

We introduce **ChatWorld**, a system combining a plugin-based virtual "metaverse" with user-controlled programmable multi-modal AI agents. The architecture leverages recent advances in virtual worlds and web technologies, allowing the rendering and simulation to be configured through a plugin and socket system.

Network participants and embodied agents interact in a shared simulation via standardized web APIs accessible from a commodity web browser that can render 3D avatars in a game engine.

ChatWorld offers real-time feedback to user and program actions, with multi-modal output for both human interaction and processing with downstream computational pipelines.

We present experimental applications of our technology including an "Infinite TV Show" stream, a Multi-Actor community decision agent, and a realtime programming AI capable of executing programs based on Twitch audience feedback.

A game prototype, Upstreet.ai, showcases the practicality of these ideas in a Massively Multiplayer Online game development context.

We hope our findings and techniques can inspire AI researchers, virtual world creators, and game developers in advancing positive collaboration between humans and AIs.

2 Introduction

2.1 Motivation

Generative AI systems are rapidly augmenting or replacing traditional methodologies for solving human problems with computation.

Such solutions are driven by at least three major competing forces of system design:

Productivity: Productivity is a measure of the system's ability to do useful work on behalf of the user. For example, the user might ask ChatGPT to write piece of publishable content, or they might use a Stable Diffusion prompt to generate a cover image. Productivity can further be divided into "business productivity", or the system's abiltiy to perform economically useful work, and "entertainment", or the tendency to invoke positive affect in the user, whihc may be divorced from external economic value.

Generativity: Generativity is the ability of a system to exhibit behaviors which the designer of the system did not forsee. Generativity is the defining characteristic of an AI baased software system. This pseudorandom element distinguishes the control flow of AI software from traditional heuristic-based programming models, though it also introduces new problems and failure modes, such as hallucinations.

Steerability: Generativity is often contrasted with a system's steerability, or its ability to perform in the manner indicated by the user. This often takes the form of prompting by the user, and involves an interplay between the user's intentions and the design intentions

and restrictions imposed by the designer of the AI system. A great deal of effort (such as prompt engineering, fint-tuning, and filtering) is spent on aligning goals between the designer and user of the system.

A fundamental issue in generative AI systems is the balance of these forces in the delivery of a useful product to the user. It has proven difficult to architect such systems using traditional software development techniques, so recent research has focused on using layered AI systems instead.

2.2 Agent Systems

A rapidly developing approach to aligning these competing goals is agent simulation systems, which mimic human agents using large language models. Such systems demonstrably perform far better than traditional heuristic system development techniques.

2.2.1 Multi-modal Agents

The adoption of single-user single-agent chatbot systems like ChatGPT has already provided economic value in diverse domains through a simple chat-based interface.

Additionally, single-agent multimodal simulacra systems, such as PALM-E [2] and Avaer's AI agent [10] can reduce friction further by automating user input through environmental context. Image captioning and optical character recognition systems can be used in near-realtime to feed context of the user's environment (e.g. the screen, virtual camera, or real-world camera) into the large language model loop, allowing the agent to respond appropriately to the environment perceived. In addition to chat/speech output, such systems can be made to perform real-world actions via mechanisms like "function calling" [7].

Limitations of the underlying sensor stack can result in perceptual inaccuracies – for example, objects in the video stream may be misidentified (via BLIP-2) or misreading text (via Doctr) in the provided video stream. Fusion over multiple frames provides for improved results. Results improve further when combined memory and reflection processing layers which can smooth out the noise in the sensor models.

Despite this, such models have been successful even when using a visual-to-language embedding. Fidelity could be improved further by using visual embeddings, as with GPT-4's image multimodal functionality.

However, such agent systems suffer from several common problems owing to the single-agent nature of the simulation.

First, it can be common to have unintended interaction loops owing to local stability of the underlying language model – that is, the agent will focus on same topics repetitively, breaking the suspension of disbelief and reducing the perceived and actual utility of the model. This can be mitigated to an extent by introducing more entropy into the prompts, for example, using long term memory.

Second, such systems must usually strike a balance between creative hallucination and accurate reflection of ground truth. For example, it is straightforward to design an LLM prompt that will produce imaginative speculations. Similarly, it is straightforward to design a prompt that strictly adheres to the factual accuracy of the information provided in the prompt. However, human thought tends to fall somewhere in the middle, with insights and

facts being fused into a coherent interaction, making prompt synthesis for such simulacra a non-trivial balancing act.

Third, social interaction does not occur in a vacuum. Real-world action involves multiple agents with differing goals interacting and competing in their pursuits. Each agent has their own set of memories and perceptions that differ from those of other agents. The result of a multi-agent interaction is often a compromise towards a more complex holistic goal than can be achieved by a single agent.

Therefore simulation research has been branched out into using multi-agent simulations to augment that attempt to overcome the shortcomings of single-agent simulations.

2.2.2 Multi-Agent Systems

Multi-agent simulations are composed of an LLM processing pipeline which accounts for more than one agent's actions during execution. In practice, this approach tends to produce better results by leveraging the full context entropy window available to language models.

For example, before prompting an agent for an action to perform, we can collect their memories and experiences interacting with their environment and other agents. Criteria for inclusion into the prompt context can be based on embedding similarity heuristics, LLM critic evaluation of importance and relevance, and similar techniques. Such context can then be fed a ReAct system to formulate action plans, which can be further refined with a Reflexion style summarization that allows the agent to consider the holistic impact of their behavior. Prompt context derived in this manner has proved significantly more effective at producing realistic agent behaviors.

An example of this strategy applied in a research environment is the virtual town simulation of [8]. Memories, action plans, and reflection layers provide for significantly improved perceived realism of the agent simulation.

In an entertainment context, SHOW-1 presents an abstracted multi-agent model simulating a writer's room and critic agents applied to virtual character storytelling. A layered critic approach, combined with heuristics, provides a superior positive disconfirmation in the viewer. [6].

MetaGPT demonstrates that multi-agent systems have real-world productivity applications and can be used to improve upon the state of the art in automated software development. The interaction of role-based agents simulating the software development lifecycle proves superior to single-agent simulations [3].

Given the success of applying multi-agent human simulation to these domains, it is likely that such systems will find application in other domains as well. Significant trial-and-error research is required to develop and test novwl implementations, and user deployment tends to lag significantly behind the state of the art.

Often the research is published without code, leaving open source AI researchers to develop a solution that can be experienced and evaluated by users. It is common for such research to never find its way out of the lab.

We hope that our work can inspire open development of such systems, leveraging interoperble frameworks using web technologies.

2.2.3 User agency

Multi-agent systems are most useful when they serve an end user need, such as entertainment or a business output. To extract work from such a system, the user should be afforded a large degree of control in determining the direction of the computation and its presentation.

One of the simplest and most powerful models of doing so is the definition of personalities for agents taking part in the simulation.

In ChatWorld, agents are configured with biographical parameters inspired by the SillyTavern character card format [4]. This JSON-based format embeds textual data about a character into a shareable image. The format allows for simple transfer of virtual character personalities between users, or uploading to a shared database.

Additionally, users strongly prefer embodied agents with an audio-visual component over purely textual representations. To account for this, ChatWorld allows for representation and configurability of backing assets of a virtual character, supporting VRM, Live2D, Sprite360, and Character Card Image formats.

ChatWorld abstracts the agent personality from the representational asset, simplifying the workflow of character asset creation and editing.

2.2.4 Programmability

In addition to user steerability, AI systems must be adaptable to an increasingly meta-programmatic paradigm owing to progression of the applicability of Large Language Models to the production of useful programming code.

We have designed ChatWorld to be a pluggable system; Javascript components can be.

ChatWorld is a pluggable system designed to integrate multi-agent models in an accessible, coherent, appealing, user-interactive agent simulation. The system can be configured via modular plugins to execute arbitrary agent strategies, while standardizing retrieval, rendering, simulation, and interface formats.

ChatWorld can load avatars in VRM, Live2D, character image ("Tavern character card"), and sprite360 formats. 3D models can be provided as GLB, with physics computed automatically.

Although the core of the agent simulation is text-based, the output of ChatWorld is multi-modal and can be consumed by downstream services. The simulation can produce an interactive video game world, a desktop companion application, an "infinite anime" TV show, or an AGI programming agent running on Twitch. We discuss these applications in more detail in the Applications section.

2.3 Framework and Architecture

ChatWorld's architecture consists of the following main components:

2.3.1 Agent Loop

We provide a default implementation of a multi-agent simulation loop, inspired by Generative Agents [8], [9], and SHOW-1 [6].

2.3.2 Agent SDK

The Agent Model Interface allows agent models written in Javascript to execute and affect the virtual world. We demonstrate our approach with software development kit (SDK) demonstrating this approach. [1]

For example, the Generative Agents research paper, along with its multi-step prompt loop, can be implemented and loaded as an agent model in ChatWorld. Multiple agents models can run in parallel, allowing for simultaneous mixing and re-mixing of implementations, decreasing the time to test, debug, and visualize research.

Agent memory, perception, events, and agent action triggers are provied as system APIs. A sandboxed JavaScript execution environment allows for a variety of oracle (external API) and timing strategies to be implemented and experimented with.

2.3.3 Simulation Layer

ChatWorld provides a configurable 3D world simulation layer providing virtual embodiment and spatialization of the agents.

This layer drives the simulation by managing the agent's tick loops, implementing in-world physics, and providing a framework for agent-agent, agent-user, and agent-world interaction.

Agents are afforded a character controller, allowing them to move around the world and interact with other agents, players, and objets in the world. Although the set of actions currently available is rudimentary, it is straightforward to extend the simulation layer to support a more complex set, with the LLM gracefully adapting to the expanded functionality.

Although the agent model is abstracted from the simulation layer and is independent of three-dimensional concepts, the focus of the ChatWorld architecture is on 3D embodiment. This approach allows for impedence match to the most common virtual environments (video games, video footage, metaverses) and the real world (including augmented reality). Such an approach allows for simpler translation of research and assets across platform and domains. Although our simulation does not curerntly implement real-world embodiment, it is possible to implement such a system using the provided APIs.

The simulation layer is responsible for ticking the registered agent models, allowing them to make decisions and affect the world at a programmable rate.

2.3.4 Rendering Interface

ChatWorld provides a rendering interface that translates simulation state into an audio-visual stream that can be displayed to the user or rendered for offline use.

The rendering interface is responsible for displaying a 3D representation of agents and the environment, GUI overlays for controlling the simulation, and rendering real-time speech and audio effects.

The ChatWorld rendering interface includes an LLM-controlled camera system, allowing for automated focus when presenting output of the simulation.

2.3.5 Networking System

ChatWorld provides a networking system that allows for multiple users to connect to the simulation and interact with the agents. The system is implemented as an spatially-mapped cluster of CloudFlare Workers that can be scaled to support an arbitrary number of users distributed throughout the game world.

The main scaling limitation of the network stack is the ability to render a significant amount of users in the world at a time, since each user is represented by a 3D avatar. Our implementation could be significantly optimized to account for this.

The system supports a CRDT-based Entity-Component-System for world objects resolution, as well as player and agent actions tracking. Positional data is interpolated client-side, providing smooth replication for all connected clients, across arbitrary avatar actions. Text chat is supported, and voice chat is streamed along with avatar visemes. Conflict resolution is done using an automatic ownership-based model ensuring eventual consistency of the simulation.

Spatial handoff across the world space is accomplished through replication and deduplication of user data across arbitrary server realms. In our sample implementation, the keys for the server realm handoff are based on the spatial coordinates of the user.

To effect smooth spatial transition across realms, the client connects to multiple servers simulaneously, and replicates character data across them. Peer clients deduplicate the object data, and the client seamlessly transitions between realms as the user moves through the world.

2.4 Generative Assets

Although not part of the core ChatWorld loop, we used multi-modal generative models to synthesize game assets. The code for this is available as part of our work.

The generative AI pipeline is both presentational and practical – renderings of assets can be fed into the perception layer to extract language embeddings. These embeddings can be used to affect the simulation by giving the virtual characters "eyes in the world".

Although we did not ultimately utilize much of this functionality in the current version, the research has strongly informed the design of the system, and the directions we see such systems evolving in the future.

2.5 Character Generation

Stable Diffusion fine-tuned image generation models seeded via a noise-based mask vector can produce high-quality artistic results. We found that separate male and female noise masks, made specifically for teh fine-tuned model worked best for consistently generating character images.

Additional masking and generation is performed on the characters to synthesize mouth flaps and affective facial expressions, which can be utilized at runtime by the rendering system to provide an emotionally evocative output. A background removal AI model is used to clip the character from the background for presentation in-world.

The resulting AI model produces prompted character synthesis with a high (¿90%) acceptance rate by the user. When combined with large language model prompt synthesis, it is possible to establish an automated, narrative-driven pipeline of virtual avatar creation. We chose a simple pseudo-3D anime style as the target format, in order to divorce the simulation from the uncanny valley and allow for greater creative expression.

We use the Zero-1-to-3 image reprojection model [5] to generate multiple viewpoints of the same character, which are selectively project based on the camera angle in a 3D sprite based system. Although rudimentary, we found that the generated characters are sufficienlty compelling that they can be used as-is in the simulation while maintaining suspension of disbelief in the world.

An expanded implementation could leverage ControlNet or similar models to generate additional character poses. These techniques have a natural extension to more complex 3D models, and we expect that future versions of systems like ChatWorld will be able to accomodate more realistic styles.

The resulting image assets can be exported as a simple spritesheet.

2.6 World Generation

Virtual embodiment is a key component of believable agent systems. In order for virtual characters to logically locomote using LLM-derived actions, they must present an understanding of their surroundings.

Large language models trained on storytelling media are good at comprehening settings presented in a familiar format, such as a movie script. However, if the description does not align with the user's perception, the agent's hallucinations can be jarring and spell-breaking for the user.

Ultimately we chose to use a model based on simple descriptions of in-world assets, leaving room for creative interpretation and positive disconfirmation. For example, agents might describe the trees in the world as "anime-style".

However, research in world generation serves as a proof of concept for future work. The ability to synthesize unlimited settings for the simulation expands the ability for agents to traverse an infnite "dreamworld" canvas which could be further.

2.6.1 Skybox Depth Model (Blockade Labs)

Blockade Labs provides a simple prompt-based model for synthesizing a 3D scene. This model is straightforward to integrate into the pipeline, as the skybox depth provides for rudimentary world physics "for free".

The fidelity of the generated art allows for agent inspection of their surroundings via a BLIP-2 based perception stack. This allows agents to have an understanding of their location within the skybox, and to be able to locomote to points of interest.

Additionally, it is quite simple to add seamless portals to the generated skyboxes, allowing for generation of a continuous world for the agents to inhabit.

2.6.2 Comic Panel Depth Model (Midjourney + MiDaS + ZoeDepth)

Utilizing depth AI models, it is possible to produce a plausible comic-style virtual world projection from an arbitrary image. This is a surprisingly simple process, involving the simple depth deformation of an image based on the depth map.

The resulting scene allows a 3D character to walk around in an arbitrary 3D space. An interesting application of this technology would be allowing a character to walk around in a real-world photograph, or a drawn comic panel.

The primary challenges with our approach were:

- Detecting the ground navmesh and orienting the camera physics. This can be done using floor/ground segmentation to detect the floor triangles, followed by a normal vector alignment of the scene. This transformation allows the characters to stand "right-side up". - Computing the camera intrinsics, allowing for proper projection of the 3D scene. For this, we use a separate AI trained on camera intrinsic data. Surprisingly, this approach proves to be quite accurate, and characters present well in the scene. - Scene bounding, ensuring that characters are constrained within the panel when walking towards the camera. This can be achieved by producing physics wall planes that conform to the extreme bounds of the scene. Generally, four wall planes are required: camera backface, floor, and left and right walls. We do not generate a top cap for the scene as it is generally not necessary to clip characters from jumping upward in the scene. - Occlusion physics filling. Generally, the straregy of using a single depth plane can present jarring occlusion physics when the native physics triangulation approach is used. For example, an avatar may be unable to walk befind a bookshelf even though it is intuitively clear from the art that they should be able to do so. To counteract this unpleasant effeft, we detect large depth discontinuities in the generated mesh triangles and use a top-down perspective render with those triangles clipped out to generate a floor physics mesh below such discontinuities, but in accordance with the detected. - Scene scaling. The MiDaS depth model originally used is not scale-invariant, thought ZoeDepth is. Nonetheless, it is not easily possible to derive the scene scale without proper ground truth as a reference point. This aspect proved the most challenging. We ended up settling for manual user scene scaling UI when producing the asset.

One interesting and under-explored aspect of this style of generation is the use of "3D inpainting", in which the 3D scene can be erased and polyfilledwith a futher depth pass, to produce an infinitely generated 3D comic panel scene.

2.6.3 Continuous Guided Video Model (Deforum)

A different technique for generating procedural scenes is to use a continuous guided video model such as Deforum.

Stable Diffusion can be used to synthesize arbitrary video scenes using a guided camera pass producing video output, followed by a depth pass. Though the resulting scenes are not perfectly stable, they exhibit interesting artistic qualities that could suffice for some use cases.

2.6.4 Unguided video model (AnimateDiff)

AnimateDiff is a model that can be used to generate arbitrary short video clips from a single prompt. While it is unguided, it can produce good artistic results.

2.7 Item Generation

Items can be generated in a similar manner to characters, though it is helpful to have a model specifically fine-tuned for the purpose. We used a similar noise diffusion technique as we use for characters, but with a different pattern more conducive to item image generation.

2.8 Audio Generation

Meta AI's MusicGen model can produce creative audio syntheses from a single text prompt. The model can be extended to produce audio of any length. Although it does not run in realtime, it is a promising direction for the future of AI generated audio soundtracks that can complement a scene.

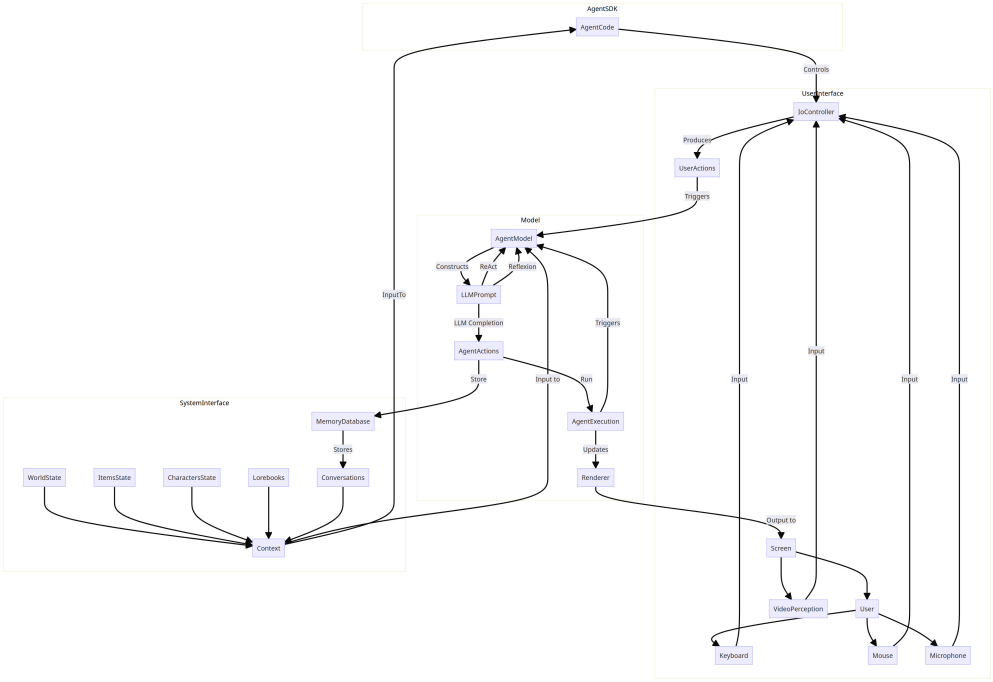
2.9 Sound Effects Generation

Meta AI's AudioGen model provides a similar API to MusicGen, but produces sound effects instead of music. Results are similarly promising, and can drastically reduce the cost of producing a sound effects library for a game.

3 Architecture Loop

The simulation is architected around a tickable agent model game loop. User inputs and agent actions cause the registered agent models to react in progressing the simulation. It is also possible for agent models to execute their own tick loop at arbitrary intervals, allowing for asynchronous agent behaviors based on arbitrary rulesets and oracle queries (e.g. external API access).

Actions and memories are logged for each agent in tbe memory database. This action/memory database can be queried by a registered agent models for use in prompt synthesis and production of agent actions to be executed by the simulation.



4 User Interface

The user interface component includes various input devices like the Keyboard, Mouse, and Microphone, as described in the diagram. These devices communicate with an I/O Controller, translating user actions into digital inputs.

- **Screen:** It's the primary output device, displaying rendered content. The Video Perception handles any visual processing required before passing the input to the I/O Controller.
- **User Actions:** Generated by the I/O Controller, these are then forwarded to the agent model to trigger specific behaviors.

4.1 System Interface

This part provides the context for agent models. It integrates the different elements that define the state and environment of the virtual world.

- **WorldState, ItemsState, CharactersState, Lorebooks:** These provide the context that encapsulates the entire state of the world, items within the world, characters, and lore.
- **MemoryDatabase:** This is where conversations are stored, providing a historical context that can be fed back into the system for richer interactions.
- **Conversations:** Integrated within the context, they are a vital part of the simulation's ongoing interaction history.

4.2 Language-Model-Led Modules (LLM)

- **LLMPrompt:** This forms the basis for language-model-led actions, feeding into the ReAct and Reflexion parts of the agent model, and creating agent actions.

4.3 Agent Model Interface

This is the component where the agent's behavior is defined, executed, and processed.

- **Context:** It includes all the information required by the agent model to make decisions, including the world state, conversations, etc.
- **AgentModel:** This processes the context, constructs the LLMPrompt, and interacts with the ReAct and Reflexion systems.
- **AgentExecution:** Triggers the agent model and updates the Renderer, reflecting changes in the virtual world.
- **AgentActions:** These are the resultant actions executed by agents. They are stored in the MemoryDatabase for future reference and are responsible for running the AgentExecution.

4.4 Rendering Interface

- **Renderer:** This translates the simulation state into an audio-visual stream. It takes updates from AgentExecution to reflect changes and outputs to the Screen.

5 Experimental Applications

ChatWorld's extensible and modular nature makes it suitable for various applications in entertainment, research, education, and more. Some notable applications include:

5.1 Companion Agent

5.2 Infinite TV Show

5.3 Interactive Virtual Game World

By leveraging the multi-agent simulation capabilities, ChatWorld can create a rich and dynamic virtual game world where players can interact with intelligent NPCs driven by state-of-the-art AI models. The NPCs can learn, adapt, and react to player actions, providing a unique and engaging gaming experience.

5.4 "Infinite Anime" TV Show

ChatWorld's rendering capabilities can be used to produce animated sequences for an "infinite" TV show. There is precedent for such genrative AI entertainment. However, to the best of our knowledge, no such system has been implemented in a multi-agent context.

5.5 AGI Programming Agent

5.5.1 Massively Multiplayer Online video game

6 Future direction

We have developed a primitive prototype of the ChatWorld system described in this research paper, available at Upstreet.ai (https//upstreet.ai/).

There are many possible directions under which such a world could be developed.

6.0.1 Massively Multiplayer Online video game

ChatWorld initially started as a Massively Multiplayer Online video game concept, and the current prototype is a proof of concept for such a game.

The promise of an infinitely generated virtual world video game inhabited by AIs is compelling, as such a game might be able to run indefinitely without a playerbase, and could might be able to survive without developer resources, as the agents could be programmed to generate their own content and evolve the game.

6.0.2 Real-world embodiment

Such an agent system could be integrated with real-world perception in an augmented reality space (e.g. Vision Pro), or a robotics context (e.g. Tesla Optimus bot). Though this paper does not cover the mechanics of such an implementation, the high-level techniques in this paper could find application in real-world perceptive agents that provide compelling intelligent services to humans.

6.0.3 Financial imbuement

There could be deep economic ramifications if a system like ChatWorld were to be successfully deployed in a real-world context. For example, the agents could be imbued with cryptographically-signed financial assets like NFTs, and could be programmed to trade either on behalf of their owners, or completely autonomously.

If such agents were able to purchase their own computing resources with their economic gains, they could be able to bootstrap their own economic growth, and could potentially become a significant economic force in the world.

7 Conclusion

We hope that the ideas presented here will be of interest to the generative AI and game developer communities, and that compelling virtual world games could leverage the techniques presented.

7.1 Generative assets

7.2 World rendering

7.3 Visual Perception Loops

Description of the methods employed for rendering visual elements and how they interact with the AI agents.

7.4 Generative AI Components

Detailing the generative algorithms used for creating images, videos, and audio within the ChatWorld system.

7.5 Memory Subsystem

Explaining the mechanism for storing and retrieving information within ChatWorld, enabling continuity and a personalized experience.

7.6 Real-time Rendering Interface

Discussion on the real-time rendering capabilities of ChatWorld, particularly in browser environments.

7.7 Applications

7.7.1 Interactive Game World ("metaverse")

7.7.2 Infinite TV Show

Discussion on the application of ChatWorld in creating an interactive and dynamic anime series.

7.7.3 Autonomous Programming Agent ("Citrine")

Description of the Citrine agent, its capabilities, and how it functions within ChatWorld.

7.8 Multi-actor Decision Agent ("Jedi Council")

Exploration of how ChatWorld can facilitate complex multi-agent decision-making processes.

7.9 Blockchain integration

Minted tokens, give agents their own crypto.

7.10 Technical Performance

Analysis of the technical aspects of ChatWorld, such as rendering speed, response time, etc.

8 Future Directions

Discussion on potential future developments, enhancements, and areas for research within the ChatWorld system.

9 Conclusion

Summarizing the key contributions of this paper, reflecting on the implications of ChatWorld, and considering its potential impact on the field of virtual worlds and AI-driven simulations.

10 Acknowledgements

Any acknowledgements to collaborators, funding sources, etc.

11 References

References

[1] avaer m00n avaer. *Upstreet: The upstreet package provides a set of tools and handy abstractions for interacting with Upstreet*. GitHub repository. 2023. URL: https://github.com/M3-org/upstreet.

[2] Danny Driess and Pete Florence. *PaLM-E: An embodied multimodal language model*. Posted by Danny Driess, Student Researcher, and Pete Florence, Research Scientist, Robotics at Google. Mar. 10, 2023. URL: https://ai.googleblog.com/2023/03/palm-e-embodied-multimodal-language.html.

[3] Sirui Hong et al. "MetaGPT: Meta Programming for Multi-Agent Collaborative Framework". In: *arXiv preprint arXiv:2308.00352* (2023). DOI: 10.48550/arXiv.2308.00352. URL: https://arxiv.org/abs/2308.00352.

[4] TAI Base by Humi et al. *SillyTavern: a mobile-friendly, Multi-API (KoboldAI/CPP, Horde, NovelAI, Ooba, OpenAI, OpenRouter, Claude, Scale), VN-like Waifu Mode, Horde SD, System TTS, WorldInfo (lorebooks), customizable UI, auto-translate, and more prompt options than you'd ever want or need*. Based on a fork of TavernAI 1.2.8. 2023. URL: https://github.com/SillyTavern/SillyTavern.

[5] Ruoshi Liu et al. "Zero-1-to-3: Zero-shot One Image to 3D Object". In: *arXiv preprint arXiv:2303.11328* (2023). URL: https://arxiv.org/pdf/2303.11328.pdf.

[6] Philipp Maas et al. *To Infinity and Beyond: SHOW-1 and Showrunner Agents in Multi-Agent Simulations*. 2023. URL: https://fablestudio.github.io/showrunner-agents/.

[7] OpenAI. *Function calling and other API updates*. We're announcing updates including more steerable API models, function calling capabilities, longer context, and lower prices. 2023. URL: https://openai.com/blog/function-calling-and-other-api-updates.

[8] Joon Sung Park et al. "Generative Agents: Interactive Simulacra of Human Behavior". In: *arXiv preprint arXiv:2304.03442* (2023). arXiv:2304.03442v2 [cs.HC]. DOI: 10.48550/arXiv.2304.03442. arXiv: 2304.03442 [cs.HC]. URL: https://arxiv.org/abs/2304.03442.

[9] Guanzhi Wang et al. "Voyager: An Open-Ended Embodied Agent with Large Language Models". In: *arXiv preprint arXiv:2305.16291* (2023). arXiv:2305.16291v1 [cs.AI]. DOI: 10.48550/arXiv.2305.16291. arXiv: 2305.16291 [cs.AI]. URL: https://arxiv.org/abs/2305.16291.

[10] webmixedreality. *Perception stack demo tweet*. Twitter post. 2023. URL: https://twitter.com/webmixedreality/status/1685621040087552001.

12 Appendix

12.1 Image gallery

Figure 1: Citrine AGI Agent Running Live on Twitch

Figure 2: Adventure Mode for ChatWorld

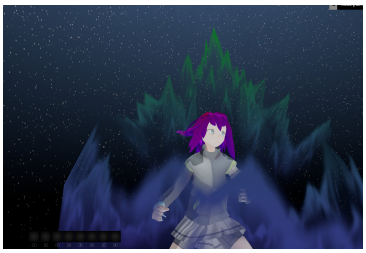


Figure 3: A game engine with interactive avatars

Figure 4: Imaginative story dialogue

Figure 5: A procedurally generated game world

Figure 6: In-engine lighting effects 1

Figure 7: In-engine lighting effects 2

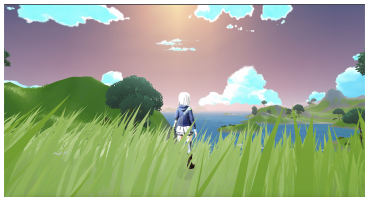


Figure 8: In-world day-night cycle

Figure 9: In-world custom avatars

Figure 10: Animatediff can generate compelling custom animations

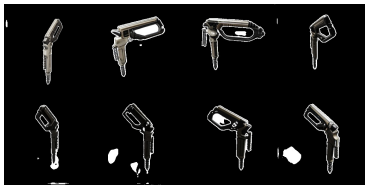


Figure 11: 3D items generated with stable diffusion and Zero-123

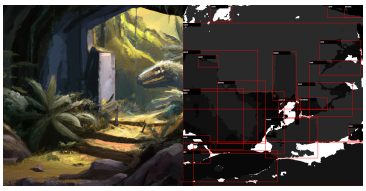


Figure 12: Image segmentation allows for visual understanding of worlds

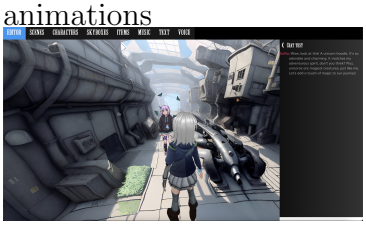


Figure 13: Blockade labs depth-based skyboxes can generate infinite linked worlds

Figure 14: Embodied VRM characters can exist in Blockade Labs worlds

Figure 15: The scale of Blockade Labs worlds can be vast

Figure 16: Avatars can wear objects in the world

Figure 17: "Citrine" artificial general intelligence agent on Twitch 1

Figure 18: "Citrine" artificial general intelligence agent on Twitch 2

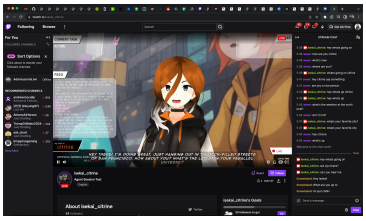


Figure 19: "Citrine" artificial general intelligence agent on Twitch 3

Figure 20: Items can be generated with Midjourney 1

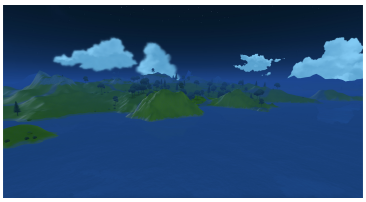


Figure 21: Procedurally generated world base

Figure 22: The engine supports character facial expressions

Figure 23: Image segmentation can be used to perceive the world

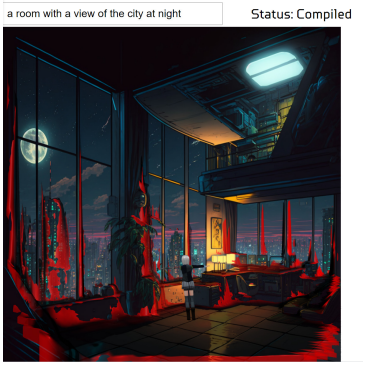


Figure 24: Images depth can be used to place characters into artwork

Figure 25: Stable diffusion character generation 1

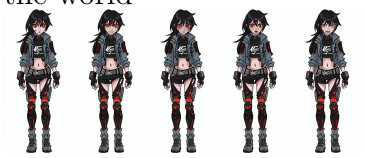


Figure 26: Stable diffusion character generation 2

Figure 27: Stable diffusion character generation 3

Figure 28: Character base image generation

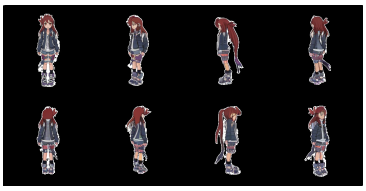


Figure 29: Character 360 spritesheet generation

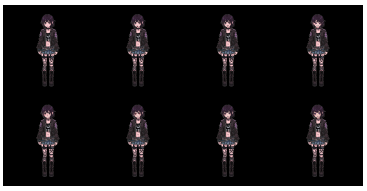


Figure 30: Character expressions spritesheet generation

Figure 31: Guided stable diffusion 1

Figure 32: Guided stable diffusion 2

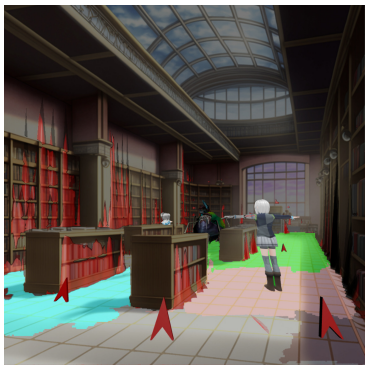


Figure 33: World generation can support scene occlusion

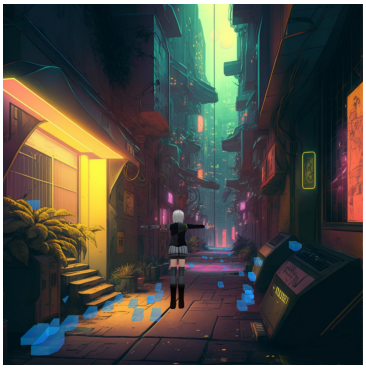


Figure 34: Image depth extrustion can support a large range of depths

Figure 35: SAM + BlIP models can be used for in-world understanding

Figure 36: Stable diffusion character generation 1

Figure 37: Stable diffusion character generation 2

Figure 38: Stable diffusion character generation 3

Figure 39: Stable diffusion character generation 4

Figure 40: Stable diffusion character generation 5

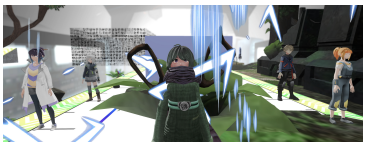


Figure 41: Particle effects

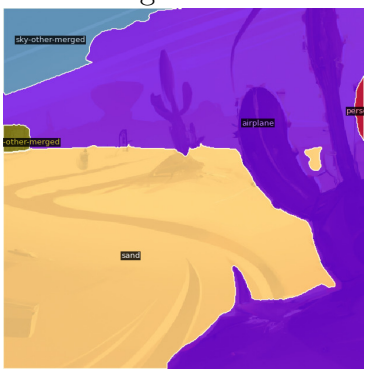


Figure 42: Image segmentation over art

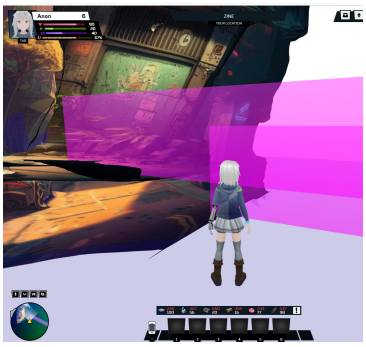


Figure 43: 2D images can be processed into 3D worlds

Figure 44: 2D -¿ 3D comic panel processing pipeline 1

Figure 45: 2D -¿ 3D comic panel processing pipeline 2

Figure 46: 2D -¿ 3D comic panel processing pipeline 3

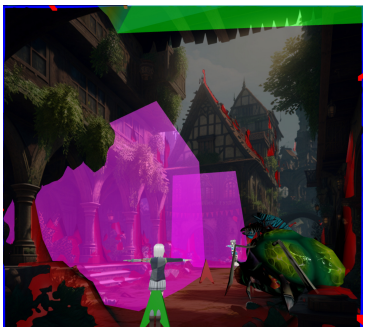


Figure 47: 2D -¿ 3D comic panel processing pipeline 4

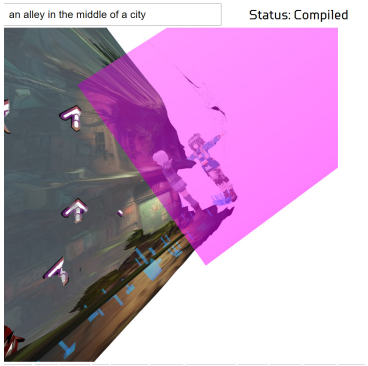


Figure 48: 2D -¿ 3D image pipeline editing

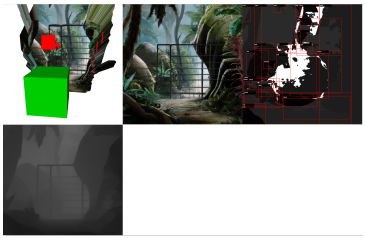


Figure 49: 2D -¿ 3D image processing pipeline

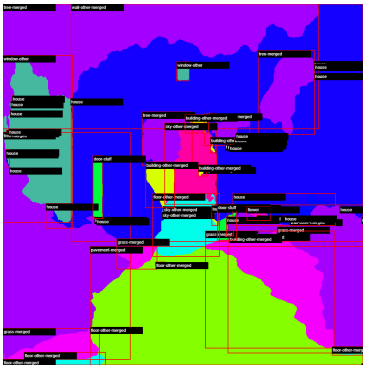


Figure 50: 2D scene segmentation

Figure 51: Procedurally generated story scenes 1

Figure 52: Procedurally generated story scenes 2

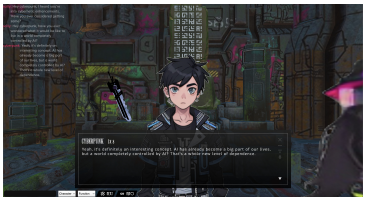


Figure 53: Procedurally generated story scenes 3

Figure 54: Realtime speech input in engine

Figure 55: Characters scene

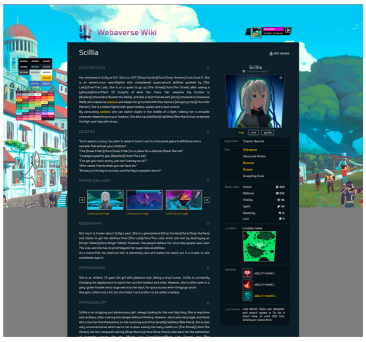


Figure 56: Procedurally generated wiki 1

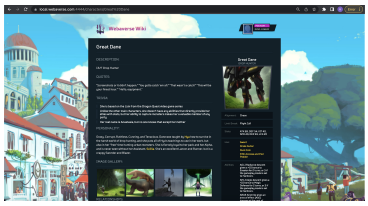


Figure 57: Procedurally generated wiki 2

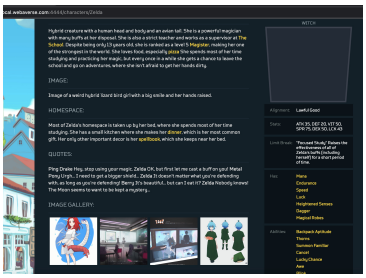


Figure 58: Procedurally generated wiki 3