# Games as Ontology Engines: AI and LLMs Invoke Spatiotemporal and Metaphysical Realities in Virtual Worlds

**Jasmine Roberts***
Microsoft & UC San Diego
jar053@ucsd.edu

**Andrzej Banburski-Fahey***
Microsoft
abanburski@microsoft.com

**Jaron Lanier**
Microsoft
jalani@microsoft.com

## Abstract

Contrary to common perception, games are not solely sources of entertainment; they are powerful mediums for modeling and communicating complex relationships. By design, games are composable systems for representing and manipulating interactions, especially the spatiotemporal. Historically, these interactions have been based around causal chains, which determine the sequencing of player actions and outcomes. In this paper, we repurpose games as 'ontology engines,' generating new metaphysical relationships and providing tools for experimenting with them. We explore how virtual worlds supported by large language models (LLMs) and real-time object-transformation systems can serve as frameworks for examining both culturally situated and universally recognizable ontologies.

## 1 Introduction

Previously, we investigated the level-editing and creative opportunities for games as generative and interactive systems for dynamically creating, manipulating and reconfiguring virtual worlds. As we explored how humans and AI could creatively generate and shape game spaces, we also showcased the potential for games to become much more than entertainment — interactive storytelling tools and world-building tools in real time. [1]

### 1.1 Ontological Modeling in Games

An ontology, in its simplest form, represents the structure of knowledge—how entities relate to each other and the meanings they convey [2]. Games, as rule-bound environments with explicit object interactions, naturally lend themselves to ontological modeling. However, the inclusion of generative language models like OpenAI's GPT [3] enables games to go beyond predefined interactions. Instead, they can dynamically generate, evaluate, and modify ontological rules based on player input, reflecting not just what is but what *could* be.

- **Spatiotemporal Relations**: Represented by physical movements, spatial arrangements, and cause-and-effect sequences.
- **Metaphysical Relations**: Expressed through symbolic meanings, metaphorical transformations, and emergent properties (e.g., turning a "knife" and a "fish" into "sushi").

*Both authors contributed equally to this work.

- **Cultural Grounding**: Generated ontologies often reflect inherent biases and cultural narratives within the language models, making games a testbed for these embedded structures. [4, 5]

### 1.2 Dynamic Ontology Generation and Exploration

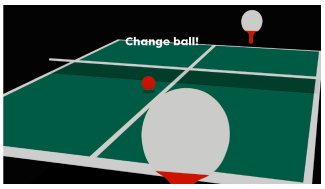
Traditional ontologies are static, representing a fixed structure of knowledge. In contrast, games can be used to generate ontologies dynamically, transforming the act of playing into a form of ontological exploration. For instance, a game could take a basic object like a "ball" and allow players to redefine its properties and relationships, turning it into a "planet" with its own gravity or a "thought" with semantic implications. This dynamic nature is achieved through a combination of AI-driven content generation, real-time rule construction, and player-driven interactions. Thus, games become not only worlds to be explored but systems for discovering new ontological truths.

## 2 Example Scenario and Use Cases

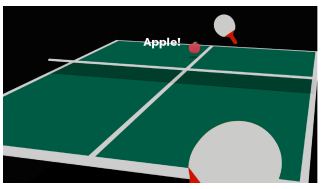
### 2.1 Sandbox Environments for Ontological Experimentation

Our game is designed as a sandbox-style environment that explores ontological relationships through object transformations, similar to titles like *Little Alchemy*. [6] Players are encouraged to experiment by combining, transforming, and modifying these entities. As players explore, the system generates a dynamic ontology that maps out discovered relationships (e.g., *"fire + water = steam"*).

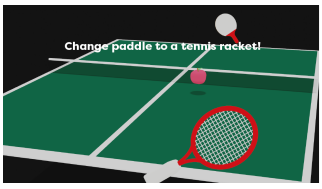
Table 1: ***Gameplay Overview*** Object Transformations and Interaction



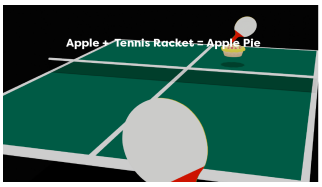

***(a)*** The system responds to *"Change ball!"* by altering the ball's form, initiating a transformation in the virtual environment.



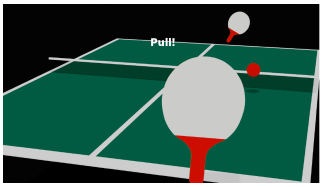

***(b)*** Following the *"Apple!"* command, the original ball is replaced by an apple, showcasing dynamic object switching.



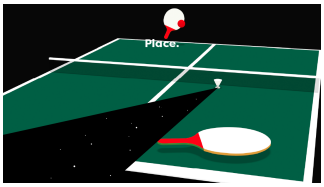

***(c)*** The paddle changes into a tennis racket, demonstrating the impact of commands that modify tools within the scene.




***(d)*** Combining the apple and racket results in a metaphorical *"Apple Pie"*, illustrating abstract reasoning and creative recombination.




***(e)*** The apple moves closer to the paddle in response to the *"Pull!"* command, highlighting object manipulation through interaction.




***(f)*** With the *"Place"* command, the apple is set down at a new location, showing precise spatial control.

### 2.2 System Architecture Overview

The architecture for LLM VR Pong is a modular integration of natural language processing (NLP), runtime code generation, and real-time scene manipulation inside the Unity game engine. The core implementation is based on the OpenAI GPT models [3], where the players' verbal instructions are interpreted and, in turn, used to generate C# code at runtime that can then be compiled and executed by Unity using the Roslyn compiler. For instance, when a player enters the command 'transform the ball to an egg', the GPT model generates and replace the existing ball visual using a 3D egg model found from the public API of Sketchfab [7]. The architecture maintains a loose coupling between the semantic understanding of the command, the code generation, and the scene management, which works and is flexible enough to add new objects or behaviours without predefined interactions for each.

Multiplayer capability is provided through Photon Unity Networking (PUN) [8], which synchronises gamestate across clients such that any transforms or object manipulations triggered by one player are reflected in another player's scene. This is accomplished via a shared state machine and networked object-identifiers, whilst allowing the system to handle multiple concurrent updates in a conflict-free manner. Real-time hand-tracking is provided by the Ultraleap Stereo IR 170 camera, by which a hand-gesture mapping is defined for each allowed transformation in the scene. Finally, voice commands are provided by Azure Cognitive Services [9], where speech is transcribed into text that is provided to Codex for further parsing. Collectively, these create an 'overhead' of software that separates user input from code generation and real-time object manipulation – making the whole system highly extensible with additional interactive behavior and dynamic content generation in a VR context.

### 2.3 Object Interaction Mechanics

The game's physics engine detects collisions and triggers a custom interaction logic system to determine the outcome based on object properties. For example, if a paddle (transformed into a frying pan) collides with a ball (transformed into an egg), the logic generates a new resultant object (e.g., a *fried egg*) based on predefined interaction rules.

Players can issue commands such as *"Change ball"* or *"Change paddle"* to dynamically alter in-game models, while additional commands like *"place"* (activating a head beam for precise placement) and *"pull"* (drawing objects toward the paddle) allow for more granular control during gameplay.

Table 2: Comparison of Objects and Transformations by GPT Models

| **Object 1** | **Object 2** | **GPT-3.5** | **GPT-4.0** | **GPT-4o** |
|---|---|---|---|---|
| Frying Pan | Egg | *Fried Egg* | *Omelet* | *Sunnyside Egg* |
| Pot | Fire | *Boiling Water* | *Steam* | *Vapor* |
| Glass | Sand | *Sand Timer* | *Hourglass* | *Hourglass* |
| Sun | Ice | *Puddle* | *Steam* | *Rainbow* |
| Scissors | Paper | *Confetti* | *Paper Shreds* | *Paper Shreds* |
| Whisk | Egg | *Omelet* | *Scrambled Egg* | *Meringue* |
| Blender | Milk | *Milk* | *Milk* | *Frothy Milk* |
| Sponge | Soap | *Bubbles* | *Soap Bubbles* | *Foam* |
| Knife | Fish | *Sushi* | *Sushi* | *Sashimi* |
| Tennis Racket | Apple | *Apple Pie* | *Apple Pie* | *Apple Pie* |
| Water | Rubber Duck | *Bath* | *Bubble Bath* | *Bubble Bath* |
| Clock | Egg | *Chicken* | *Hatched Chick* | *Phoenix* |
| Toaster | Bread | *Toast* | *Toast* | *Toast* |

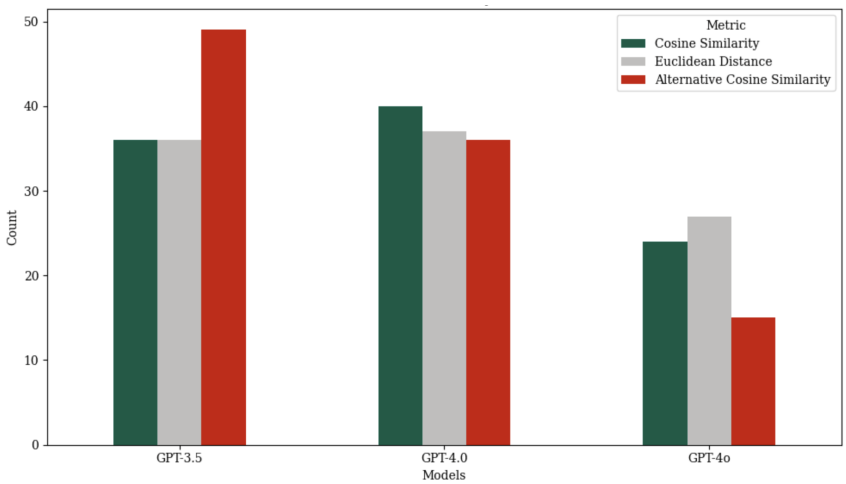
## 3 Discussion and Analysis

## 4 Analysis of GPT Models and Ontological Transformations

The table comparing object transformations by GPT models and the performance graph of different similarity metrics (Figure 1) illustrate how these models act as dynamic engines for creating and modifying ontological structures—mirroring the way games operate as ontology engines. In games, entities are defined not just by their static attributes but by how they interact and transform in response to in-game actions, much like the object pairings in the table (e.g., *Egg + Frying Pan = Fried Egg, Omelet, or Sunnyside Egg*). Each GPT model interprets these interactions differently, producing varying ontological states depending on its internal representation.

Figure 1 highlights the accuracy of GPT-3.5, GPT-4.0, and GPT-4o models under three different similarity metrics: **Cosine Similarity** [10], **Euclidean Distance** [10], and **Alternative Cosine Similarity** [11]. The graph reveals that while GPT-3.5 and GPT-4o perform strongly under specific metrics, GPT-4.0 shows a more balanced performance across all metrics, indicating that it captures a wider

Figure 1: Most Accurate Model Count by Different Metrics




range of transformation patterns. This suggests that GPT-4.0 has more versatile embeddings capable of handling both literal state changes and nuanced, context-based transformations. For example, the table shows that for the transformation *Egg + Frying Pan*, GPT-3.5 produces the straightforward result of a "Fried Egg," while GPT-4o outputs a more specialized result like "Sunnyside Egg." This difference in representation is reflected in Figure 1, where the high count for GPT-4o using Alternative Cosine Similarity indicates its strength in capturing more abstract or context-sensitive changes.

Moreover, the graph shows that **Alternative Cosine Similarity** favors more complex and contextually rich transformations, as GPT-4o achieves its highest performance under this metric. This is particularly evident when comparing outputs like *Clock + Egg* transforming into a "Phoenix," which would be challenging to capture using basic spatial similarity metrics. In contrast, Euclidean Distance, which performs best for GPT-4.0, is more effective at modeling transformations that involve straightforward state changes, such as *Ice* $\rightarrow$ *Steam* or *Water* $\rightarrow$ *Vapor*. This variation across metrics underscores the importance of selecting the right similarity measure depending on whether the goal is to capture **physical state transitions** or **conceptual context shifts**.

### 4.1 Conceptual Depth and Symbolic Understanding

The development of conceptual depth across these object transformations can also be seen in the way that derivation moves from being rather literal (e.g., *Whisk + Egg = Omelet*) to being more nuanced or specialised (e.g., *Meringue*), as shown in Figure 1.For GPT-3.5, note that we see higher numbers of generations that follow a type of schematic recipe, while the higher numbers of generations under Alternative Cosine Similarity seen for GPT-4o correspond to more abstract and symbolic outputs.

Completions that stay consistent – such as the *Toast* result – imply that some base-ontological associations are robust and well-structured. Those that change – as in *Puddle* to *Rainbow* – reveal how the models progress from natural responses to environmental stimuli to integrative models of complex phenomena that increase in internal complexity (quantitatively described by the metrics shown in Figure 1).

## 5 Conclusion

Games, when combined with LLM's and dynamic object transformation systems, become ontology engines—structures capable of rendering more than fun. They become mediums for discovering, modeling, and communicating complex metaphysical relations and cultural narratives.Figure 1, supplemented with the table of object transformations, shows that the efficiency of these systems is dependent on the proper selection of similarity measures useful and necessary for encoding and changing semantic structures in a dynamic way.

## References


[1] Roberts, J., Banburski-Fahey, A., & Lanier, J. (2022) Steps towards prompt-based creation of virtual worlds. *arXiv preprint arXiv:2211.05875*.

[2] Gruber, T.R. (1993) A translation approach to portable ontology specifications. *Knowledge Acquisition* **5**(2):199–220. https://doi.org/10.1006/knac.1993.1008.

[3] OpenAI. (n.d.) *OpenAI models documentation.* Retrieved September 30, 2024, from https://platform.openai.com/docs/models.

[4] Harrell, D.F. (2013) *Phantasmal Media: An Approach to Imagination, Computation, and Expression.* Cambridge, MA: MIT Press.

[5] Lanier, J. (2010) *You Are Not a Gadget: A Manifesto.* New York, NY: Alfred A. Knopf.

[6] Jakubowski, R., & Wojciechowski, M. (2010) *Little Alchemy* [Game]. Recloak. Retrieved September 30, 2024, from https://littlealchemy.com.

[7] Sketchfab. (n.d.) *Sketchfab: Discover, share, and buy 3D models.* Retrieved September 30, 2024, from https://sketchfab.com.

[8] Exit Games. (n.d.) *Photon: Real-time multiplayer game development framework.* Retrieved September 30, 2024, from https://www.photonengine.com.

[9] Microsoft. (2024). Azure Cognitive Services. https://azure.microsoft.com/en-us/services/cognitive-services/.

[10] Salton, G., Wong, A., & Yang, C.S. (1975). A vector space model for automatic indexing. *Communications of the ACM*, **18**(11):613–620. https://doi.org/10.1145/361219.361220.

[11] Sidorov, G., Velasquez, F., Stamatatos, E., Gelbukh, A., & Chanona-Hernández, L. (2014). Soft Similarity and Soft Cosine Measure: Similarity of Features in Vector Space Model. *Computación y Sistemas*, **18**(3):491-504. https://www.cys.cic.ipn.mx/ojs/index.php/CyS/article/view/2043.


## A Context for prompts

Here we collect the context appended to prompts for our experiments. Codex and GPT-3 are highly capable in the few-shot regime, in which we provide a few examples of desired operation.

### A.1 Prompt to GPT-3 for creative collisions

This is a magical game like ping pong, in which the players can change both the ball and their paddles, when the transformed ball object hits the transformed paddle, it changes the ball according to how you'd expect those two objects to interact.

When a spawned loaf of bread collides with spawned cheese it spawns A sandwich object.

When a spawned pen collides with spawned paper it spawns a notebook object.

When spawned meat collides with a spawned clock it spawns a bacteria object.

When a music note object collides with a cube object it spawns an instrument.

When water object collides with air object it spawns ice.

When a tree collides with a clock, it spawns a dead tree.

When an egg collides with a clock, it spawns a chicken.

When a cube collides with a wheel, it spawns a car.

When an egg collides with a frying pan, it spawns a fried egg.

When a balloon collides with a pin, it spawns a popped balloon.

When a bread collides with a clock, it spawns a moldy bread.

When a caterpillar collides with a clock, it spawns a butterfly.

When water collides with fire, it spawns steam.

When seed collides with water, it spawns a plant.

When egg collides with clock, it spawns