Multiverse Mechanica: A Testbed for Learning Game Mechanics via Counterfactual Worlds

Anonymous authors
Paper under double-blind review

Abstract

We study how generative world models trained on video games can go beyond mere reproduction of gameplay visuals to learning *game mechanics*—the modular rules that causally govern gameplay. We introduce a formalization of the concept of game mechanics that operationalizes mechanic-learning as a *causal counterfactual inference* task and uses the *causal consistency principle* to address the challenge of generating gameplay with world models that do not violate game rules. We present **Multiverse Mechanica**, a playable video game testbed that implements a set of ground truth game mechanics based on our causal formalism. The game natively emits training data, where each training example is paired with a set of causal DAGs that encode causality, consistency, and counterfactual dependence specific to the mechanic that is in play—these provide additional artifacts that could be leveraged in mechanic-learning experiments. We provide a proof-of-concept that demonstrates fine-tuning a pre-trained model that targets mechanic learning. Multiverse Mechanica is a testbed that provides a reproducible, low-cost path for studying and comparing methods that aim to learn game mechanics—not just pixels.

1 Introduction

Interactive world models have recently gained attention for their potential to simulate or extend video game experiences (Bruce et al., 2024; Parker-Holder & Fruchter, 2025; Decart et al., 2024; He et al., 2025; Che et al., 2025). SOTA models, typically leveraging deep autoregressive transformer architectures, are trained on large datasets containing sequences of visual frames, user inputs, or internal virtual states produced by a graphics engine. These models of video games (game world models) can generate gameplay sequences that are visually similar to original gameplay; an impressive feat considering modern video games often have cinematic levels of visual complexity.

A key motivation for a focus on video games is procedural generation of novel gameplay experiences (Gingerson et al., 2024). To this end, the ability to produce high-fidelity visuals is a necessity, as the novel experiences must look the part. But in addition to looking good, the game world model must generate gameplay that is **consistent** with the game's **mechanics** (Gingerson et al., 2024). In simple terms, if the generated gameplay violates game rules or logic, it breaks the gaming experience and, therefore, is not useful, regardless of how good it looks.

Authors of game world models often claim to have learned a game's *mechanics*—rules governing gameplay—through post-hoc observations of generated gameplay from the trained model, which visually demonstrates the mechanics in play. For example, in reference to their *World Models* framework, the authors Ha & Schmidhuber (2018) claim that "by learning only from raw image data collected from random episodes, [their model] learns how to simulate the essential aspects of the game, such as the game logic, enemy behavior, physics ...". Kim et al. (2020) claim their *GameGAN* model learned the collision and power pellet mechanics of PAC-MAN. Parker-Holder & Fruchter (2025) claimed that consistency was an "emergent" property of Genie 3.

There is a problem with such claims. An *a posteriori* observation that a game world model has learned a mechanic demonstrates that it is *possible* to learn *some* mechanics in *some* contexts with SOTA architectures. However, this does not tell us *a priori* that it is possible to reliably reproduce

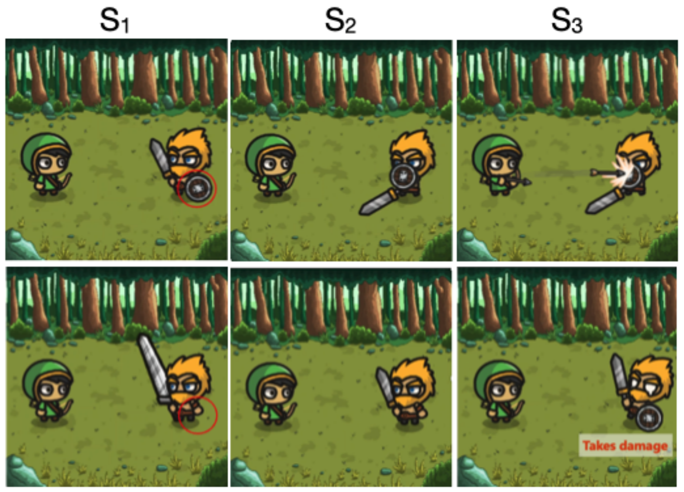


Figure 1: Clips from consistent contrasts sampled at their respective impact frames: each column contains two contrasting gameplay clips where differences are solely attributable to the shield mechanic. Left, middle, and right correspond to parallel world statements $\mathbf{S_1}$, $\mathbf{S_2}$, $\mathbf{S_3}$.

a *certain* context. To use a game world model to generate novel gameplay experiences in practical settings, we need this knowledge before expending on training and deploying a model.

Part of the difficulty is a lack of any formalized notion of what it means to learn a game mechanic. With a formal definition, we could determine if a mechanic is statistically identifiable from training data, and we could evaluate claims that a model has learned a mechanic. Moreover, we could define exactly what it is that we want to identify. The model needs to learn a representation of a mechanic sufficient to reliably reproduce it in generated gameplay, without going as far as reverse-engineering the mechanic's source code; a formal definition would tell us the level of detail required. Lacking this, we cannot know when we have disentangled the mechanic's representation from learning other representations that drive the game's visuals.

In this paper, we address this problem with the following contributions. Firstly, we provide a **causal formalization** of the concept of game mechanics using causal graphical modeling theory. We use this formalization to demonstrate data and inductive biases (in the form of causal graphs) that enable learning a game mechanic. We introduce Multiverse Mechanica, a playable game for use as a testbed for evaluating the learning of game mechanics. The game mechanics are implemented with our causal graphical formalization, providing a ground truth for evaluating mechanic learning. It provides causal graphical representations of the mechanic directly to the user for use in evaluation or to supervise mechanic-learning. Multiverse Mechanica is visually simple enough to facilitate inexpensive experimentation, while it enables generalization by using mechanics that are typical of fantasy combat games. Finally, we provide a proof-of-concept for fine-tuning a generative model on a specific game mechanism that leverages these causal representations in its objective function.

2 Background and Related Work

Defining Game Mechanics. Building on prior definitions (Lo et al., 2021), we define a *game mechanic* as a *modular subset* (Björk & Holopainen, 2004; Schaul, 2013; Thielscher, 2011; Zook & Riedl, 2019) of the *game rules* triggered by specific player/agent interactions (Lundgren & Bjork, 2003; Fullerton et al., 2004), producing changes in *game state* (Järvinen, 2008; Fabricatore, 2007) that shape gameplay visuals (Hunicke et al., 2004). These subsets entail causal relations with preconditions and effects, representable as logic, finite-state machines, behavior trees, or transition functions (Zook & Riedl, 2014; Thielscher, 2011; Schaul, 2013; Dormans, 2012; Zook & Riedl, 2019). We formalize this definition in Section 3.2.

Causal Framing. We adopt the *causal hierarchy*, which describes three levels of statements that employ causal logic: (level-1) observation, (level-2) intervention, and (level-3) counterfactual

Level	Statement
1) Observation	"The scene *might* generate to show an enemy taking damage from an attack."
2) Intervention	"*If* an enemy were given a shield and they blocked an incoming attack, *then* the scene *would* generate to show the enemy not taking damage."
3) Counterfactual	"*Given* an enemy did not have a shield and the scene generated showing them taking damage from an incoming attack, *if* they instead had a shield and they blocked, the scene *would have* generated to show the enemy not taking damage."

Table 1: Examples of the three causal hierarchy levels, in the context of game scene generation.

(Bareinboim et al., 2022). A predictive statement, such as "The scene would generate in *this* way," is a level-1 observational statement that asks questions about the distribution of possible game scene generations $P(\mathbf{V})$. A level-2 interventional statement targets outcomes under hypothetical interventions, such as "If an enemy were placed here (e.g., in a place it would not naturally appear), then the scene would generate in *this* way.", asks questions about $P(\mathbf{V} \mid \mathrm{do}(X{=}x))$. A level-3 counterfactual statement considers observed outcomes and how those outcomes might have been different under hypothetical interventions, and asks questions about joint outcomes under conflicting interventions, e.g., $(Y_{X=x}, Y_{X=x'})$ Consequently, Level-3 quantities encode "all-else-equal" comparisons across worlds. For example: "Given there was no enemy and the scene generated that way, if an enemy were placed here, the scene would generate in *this* way." Table 1 provides examples of the three levels of statements, grounded in the context of video game scene generation. Further background on the causal hierarchy is provided in Appendix D.1.

We use causal DAGs and *structural causal models* (SCMs) (Pearl, 2009) to model mechanics as mechanisms (rules) (Bongers et al., 2018). For focusing on a mechanic's variables, we reference *marginalized DAGs* (mDAGs) that preserve causal and interventional semantics while marginalizing others (Evans, 2016). The *causal consistency principle* states that variables not downstream of an intervention retain the same value across worlds (Pearl, 2010; Shpitser & Pearl, 2012); *counterfactual graphs* capture this cross-world consistency compactly (single nodes for shared variables; world-indexed nodes for affected ones) (Shpitser & Pearl, 2012). Construction details appear in Appendix D.5; additional background is in Appendix D.

Learning game mechanics. Empirical results highlight the challenges of learning game mechanics. A study performed by Gingerson et al. (2024) highlights the continuing challenge of generating *consistent* gameplay with SOTA architectures—even when generations look plausible, they frequently break the rules of the mechanic. Chen et al. (2025) report similar failures in spatial and numerical consistency, necessitating explicit corrective modules. More broadly, empirical studies of video prediction models show that while they excel in-distribution, they often rely on case-based mimicry and fail under distribution shift, violating simple physical principles (Kang et al., 2024; Riochet et al., 2021).

If we view a mechanic as a latent generative factor, then unsupervised learning of mechanics is provably impossible from video observations of gameplay alone (Locatello et al., 2019) without strong inductive biases (Mitchell, 1980; Wolpert, 1996). Prior work in this area shows that causal representations are at best only partially identifiable from observational data without intervention data or strong causal inductive biases (Spirtes et al., 2000; Bareinboim & Pearl, 2022; Schölkopf et al., 2021). Our work builds on prior work that employs these causal approaches to learning latent generative factors. But to our knowledge, our work is the first to apply this type of causal analysis to the problem of learning game mechanics during training, and generating consistent gameplay from a trained model.

Datasets, testbeds and environments. Existing testbeds, datasets, and environments for world models and video prediction largely emphasize intuitive reasoning about real-world Newtonian physics rather than explicitly defined game mechanics. For instance, IntPhys (Riochet et al., 2018) probes intuitive physics by testing whether models respect basic object permanence and motion, while Physion (Bear et al., 2021) provides simulated videos of collisions and stability events to evaluate physical prediction. However, game mechanics can encompass non-realistic "physics," such as spell casting and passing through portals. *Multiverse Mechanica* focuses on a broader set of game

Descriptions in Causal Logic	Formal Counterfactual Notation	Consistent Contrast Sample Data
S₁ All else equal, if the opponent has a light weapon, they may equip a shield; if heavy, they cannot.	$\mathcal{S}_1: \ P(S_{W=1}=1,\ S_{W=0}=0) \geq \epsilon_1$	$\mathcal{D}_1 = \{\omega_1,\ C, (S_{W=1}, B_{W=1}, D_{W=1}, V_{W=1}), (S_{W=0}, B_{W=0}, D_{W=0}, V_{W=0})\}$
S₂ All else equal, if the opponent has a shield, they may block; if no shield, they cannot.	$\mathcal{S}_2: \ P(B_{S=1}=1,\ B_{S=0}=0) \geq \epsilon_2$	$\mathcal{D}_2 = \{\omega_2,\ C,\ W, (B_{S=1}, D_{S=1}, V_{S=1}), (B_{S=0}, D_{S=0}, V_{S=0})\}$
S₃ Given a shield, if a block succeeds then no damage; if it fails, damage occurs.	$\mathcal{S}_3: \ P(D_{B=1}=0,\ D_{B=0}=1 \mid S=1) \geq \epsilon_3$	$\mathcal{D}_3 = \{\omega_3,\ C,\ W,\ S, (D_{B=1}, V_{B=1}), (D_{B=0}, V_{B=0})\}$

Table 2: The shield mechanic described as in natural language causal logic (column 1), which are then formalized with counterfactual notation (column 2), where strictly positive probabilities ($\epsilon_i > 0$) indicate bounded uncertainty due to other causal factors in the system. Column 3 shows *consistent-contrast* tuples (each row shares seed ω_i).

Variable / Operator	Description
$P(X)$	The probability distribution for random variable X
$P(X \mid Z{=}z)$	As above, but conditioned on observation $Z{=}z$
$\text{do}(X{=}x)$	An intervention taken to fix variable X to value x. Interrupts sampling from its conditional probability distribution given parents in the causal DAG: $P(X \mid Pa(X))$.
$P(Y_{X=x} \mid Z{=}z)$	Probability distribution for Y where an intervention is taken to fix variable X to value x $\big($i.e., $\text{do}(X{=}x)\big)$ (optionally conditioned on observation $Z{=}z$)
Sᵢ, $\mathcal{S}_i$	The i$^{\text{th}}$ contrast statement, in causal terms and counterfactual notation, respectively
$\mathcal{D}_i$	The i$^{\text{th}}$ *consistent-contrast* tuple, defining two worlds sharing a common seed ω_i
X_t, C_t, V_t	Game data at frame t: game state X_t, player controller input C_t, and video frame (image) vector.

Table 3: Key variables and operators used in the adopted formalism.

mechanics, and contributes a *playable generator* that emits data and artifacts that target learning of a formally defined ground-truth set of mechanics.

3 FORMALIZING AND LEARNING A GAME MECHANIC

In this section, we demonstrate the formalization of a game mechanic as well as how we would learn that mechanic from data. Then, in Section 3.2, we provide a general mathematical description of this approach.

3.1 ILLUSTRATING EXAMPLE

The Shield Mechanic Consider a scene from a stylized 90's fantasy turn-based combat game, where an archer battles a warrior. Like many games in this genre, there is a *shield mechanic*, as shown in Figure 1, where the warrior may raise a shield to block incoming attacks.

3.1.1 FORMALIZING THE SHIELD MECHANIC

How might we describe this shield mechanic in formal causal terms?

Step 0: Define Notation. As a preliminary step, we define the necessary variables and operators used in our formalism in a notation table (see Table 3).

Step 1: Describe the Mechanic with Causal Logic. We start by completely describing the shield mechanic using a series of causal hypothetical statements of the form "Given preconditions W, all else equal, if X, then Y." Specifically, we focus on level-3 multiverse logic statements that employ

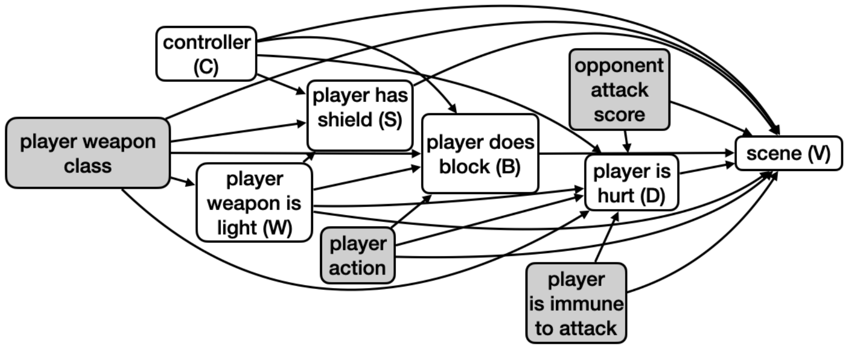


Figure 2: Causal DAG marginalized to focus on the nodes specific to the shield mechanic and their latent confounders (gray nodes)

conjunctions of conflicting conditions. Table 2 column 1 shows three statements, $\mathbf{S_1}$, $\mathbf{S_2}$, and $\mathbf{S_3}$, that fully describe the shield mechanic.

The columns of Figure 1 correspond to $\mathbf{S_1}$, $\mathbf{S_2}$, and $\mathbf{S_3}$. We could instead use *level-2* interventional statements, which are normally preferred because they are generally testable with experimental data. But the *level-3 parallel world* statements provide an additional constraint in the phrase "all else equal"; that outcomes unaffected by the conditions must remain *consistent* across the clauses. As we will see below, we can use that constraint to operationalize consistency in game generation. Secondly, we can leverage the gaming setting's rare opportunity to generate level-3 data to validate level-3 statements.

Step 2: Rewrite as Counterfactual Expressions. We can rewrite $\mathbf{S_1}$, $\mathbf{S_2}$, and $\mathbf{S_3}$ as mathematical expressions using counterfactual notation, capturing the contrasts more compactly. For simplicity, let W denote the weapon type, S indicate whether a shield is equipped, B indicate whether the shield is used to block, and D indicate whether damage occurs. We treat these as binary variables for clarity, without loss of generality. Let $W = 1$ and $W = 0$ denote light and heavy weapon, respectively. S, B, and D, let 1 mean *True* and 0 mean *False*.

We use counterfactual notation to denote variables under the influence of intervention, such that Y under an intervention that sets X to x is written as $Y_{X=x}$. We can formalize the parallel world statements $\mathbf{S_1}$, $\mathbf{S_2}$, and $\mathbf{S_3}$ as shown in column 2 of Table 2, where $\epsilon_i; \forall i \in \{1, 2, 3\}$ denotes strictly positive probabilities ($\epsilon_i > 0$), indicating bounded uncertainty due to other causal factors in the system.

With this, our shield mechanic is described in formal mathematical terms.

Step 3: Represent Mechanic with Causal Graphs Let V denote a full clip of gameplay. An outcome, denoted v, is a sequence of frames. Let C denote the controller input from a player at the start of the player's turn. Let G denote the full causal DAG for a single turn in the battle (see the full graph in Figure 6 in Appendix E). Let us assume we have access to this DAG, or that we could create it using knowledge of the game structure, analyzing causal dependence in the game's code (Winskel, 1986; Aho et al., 2006), or by applying causal discovery methods (Glymour et al., 2019). The variables implicated in our description of the shield mechanic are $Z = \{C, W, S, B, D, V\}$. The causal DAG G is quite large, so we derive the mDAG $G^{\mathcal{M}}$ that zooms in on Z (Figure 2) by marginalizing out the variables not in Z (see Appendix D.3 for a description of the algorithm). Next, we can combine the mDAG with each counterfactual expression in the mechanic's description to construct the counterfactual graphs in Figure 3.

The counterfactual graphs in Figure 3 encode a representation of *causal consistency*—variables that are not downstream of interventions and thus are consistent across worlds are unique, while inconsistent variables have nodes indexed by each world. Thus, the counterfactual graphs are representations of the shield mechanic that explicitly describe what should remain consistent when generating gameplay depicting the shield mechanic.

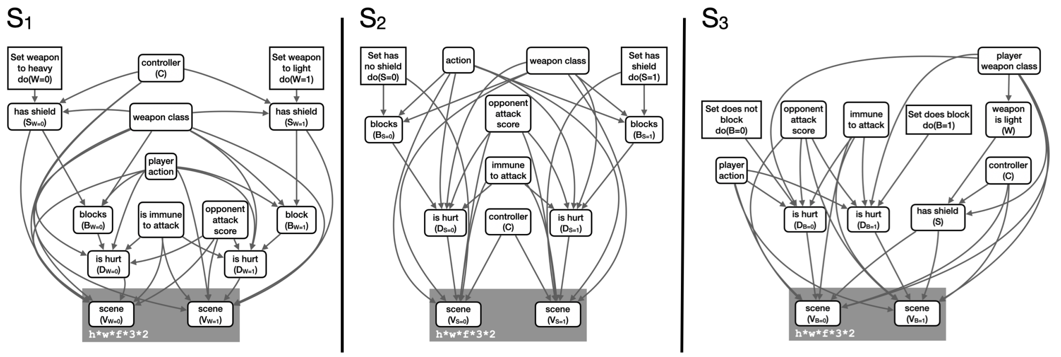


Figure 3: Counterfactual graphs for the shield mechanic. The video variables $V_{X=x}$ have shape height(h)*width(w)*frames(f)*channels(3)*worlds(2).

3.1.2 Generating Shield Mechanic Data

We can generate level-3 parallel world data consistent with $\mathcal{S}_1$, $\mathcal{S}_2$, and $\mathcal{S}_3$ by creating parallel runs with identical initial conditions and random seeds. We can intervene separately in each run, producing clips of parallel *virtual* worlds that differ only in their respective interventions. We call the tuple of these clips, combined with the outcomes of other mechanic-related variables under these interventions and their shared initial condition/seed, *consistent contrasts*. Figure 1 illustrates gameplay clips from the contrasts.

Let ω_1, ω_2, and ω_3 represent distinct sets of random seeds and initial conditions. Let C represent the controller input from the player. Let $V_{X=x}$ represent a video clip of gameplay under an intervention that sets X to x. Reading consistency from the graphs in Figure 3, we see $W_{S=1} = W_{S=0} = W$, $W_{B=1} = W_{B=0} = W$, and $S_{B=1} = S_{B=0} = S$. Let $\mathcal{D}_1$, $\mathcal{D}_2$, and $\mathcal{D}_3$ represent samples of consistent contrasts for $\mathbf{S_1}$, $\mathbf{S_2}$, and $\mathbf{S_3}$ respectively, as shown in Table 2 column 3.

3.1.3 Learning the Shield Mechanic from Data

Let $P_1 = P(S_{W=1}, S_{W=0})$, $P_2 = P(B_{S=1}, B_{S=0})$, and $P_3 = P(D_{B=1}, D_{B=0}, S)$ denote the distributions constrained by $\mathcal{S}_1$, $\mathcal{S}_2$, and $\mathcal{S}_3$. We can estimate these distributions through repeated sampling of consistent contrasts $\mathcal{D}_1$, $\mathcal{D}_2$, and $\mathcal{D}_3$, then averaging over the sampling distributions to obtain the sampling distributions $\hat{P}_1$, $\hat{P}_1$, and $\hat{P}_3$ (see Appendix D.6). In each case $\hat{P}_i$ converges almost surely to P_i. This provides a precise operationalization of what it means for a generative model to learn the shield mechanic: learning constraints $\{\mathcal{S}_1, \mathcal{S}_2, \mathcal{S}_3\}$ on distributions $\{P_1, P_2, P_3\}$ or modeling $\{P_1, P_2, P_3\}$ directly.

However, in the canonical case of training game world models, we assume that only the controller inputs and the video outputs are observed during training. Here, the inference of $\hat{P}_i$ becomes a task of unsupervised learning of a latent vector $\mathcal{D}\backslash\{C, V_{X=x}, V_{X=x'}\}$ using $\{C, V_{X=x}, V_{X=x'}\}$ as features, where $V_{X=x}, V_{X=x'}$ is a vector of shape 2 * frame height * frame width * 3 RGB channels * number of frames. Without further assumptions, disentangling the components of $\mathcal{D} \setminus \{C, V_{X=x}, V_{X=x'}\}$ is generally infeasible. However, the counterfactual graphs in Figure 3 already disentangle these variables for us. Using these, the problem reduces to training a latent variable model.

3.2 Formal Framework for Game Mechanics

We assume a causal DAG $G = (\mathsf{V}, \mathsf{E})$ for a single step of gameplay with $\mathsf{V} = \{C_t, X_t, C_{t+1}, X_{t+1}, V_t, V_{t+1}\}$, and typical edges $X_t \to C_t$, $X_t \to X_{t+1}$, $C_t \to X_{t+1}$, $X_t \to V_t$, $X_{t+1} \to V_{t+1}$. For a given mechanic, we restrict to the variable subset $M = \{C_t, X_t^M, X_{t+1}^M, V_{t+1}\}$, $X_t^M \subseteq X_t$, $X_{t+1}^M \subseteq X_{t+1}$, and form the marginalized DAG G^M by marginalizing variables outside M while preserving interventional semantics (Appendix D.3).

We work on a probability space with sample space Ω. Each variable $Z \in M$ is a measurable mapping $Z : \Omega \to \mathcal{X}_Z$, and each counterfactual $Z_{X=x} : \Omega \to \mathcal{X}_Z$, where $x \in \mathcal{X}_X$ is an intervention value in the state space of X. The ground-truth SCM, consistent with G^M, induces the family

$$\mathcal{F}(M) \;=\; \Big\{\, P\big(V_{t+1,X=x},\, Z_{X=x} \,\big|\, E\big) \;:\; X, Z \in M,\; x \in \mathcal{X}_X,\; E \in \sigma(M) \Big\},$$

where $\sigma(M)$ denotes the σ-algebra generated by the variables in M (in practice, E can be any measurable predicate on M, e.g., $S{=}1$ for "has shield").

We formalize a mechanic as the tuple $\langle G^M, \mathcal{M}\rangle$, where $\mathcal{M} = \{S_1, \ldots, S_k\}$ and each constraint $\mathcal{S}_i \;:\; P\Big(\bigwedge_{j=1}^{m_i} Y_{X=x_j} = y_j \,\Big|\, E\Big) \;\geq\; \epsilon_i$ binds counterfactuals of variables in M under interventions on $X \in M$, with $x_j \in \mathcal{X}_X$, $E \in \sigma(M)$, and $\epsilon_i \in (0,1]$ (allowing non-deterministic relations due to factors outside M). For each $\mathcal{S}_i$ we denote the targeted distribution by P_i (e.g., $P_1 = P(S_{W=1}, S_{W=0})$ in the shield example).

Data and Estimation. A consistent-contrast dataset for $\mathcal{S}_i$ of size N is

$$\mathcal{D}_i^{(N)} = \Big\{ \big(V_{X=x_1}(\omega_n), \ldots, V_{X=x_{m_i}}(\omega_n)\big) \;:\; \omega_n \in \Omega,\; n = 1, \ldots, N \Big\},$$

optionally restricted to ω_n satisfying E. Let $\hat{P}_i^{(N)}$ be the empirical distribution induced by $\mathcal{D}_i^{(N)}$. Under i.i.d. sampling of seeds ω_n, $\hat{P}_i^{(N)} \xrightarrow{\text{a.s.}} P_i$. For each $\mathcal{S}_i$ (and associated P_i), we construct a *counterfactual graph* $G_i^{M,\text{cf}}$. In partially observed settings (video + controller only), G_i^{cf} specifies which latent variables are shared across worlds and which differ, reducing estimation to a well-posed latent variable problem aligned with the counterfactual graph's structure.

4 MULTIVERSE MECHANICA: A PLAYABLE TESTBED FOR LEARNING MECHANICS

We introduce *Multiverse Mechanica*, a fantasy-style battle game designed as a testbed for learning game mechanics. Unlike static datasets, Multiverse Mechanica is a playable game that emits the artifacts required to study and evaluate whether models capture the game's mechanics—not just gameplay visuals. Its design integrates three innovations: (i) native support for level-3 parallel-world contrasts with consistency under the same ω; (ii) per-mechanic mDAGs $G^{\mathcal{M}}$, parallel world and counterfactual graphs, and specifications of $\mathcal{M}$; and (iii) explicit visual grounding, where stance and scene variables are rendered into pixels. We will release the game upon publication.

4.1 GAME OVERVIEW

Each episode consists of a pre-battle setup (character and equipment selection, random assignment of elemental buffs (e.g., fire, ice) followed by turn-based combat. The player occupies the left side of the screen, and the enemy occupies the right. On the player's turn to attack, a timing-based interaction yields an attack score; the enemy's turn samples an analogous attack score. Outcomes depend on weapons, defenses, the attack score, and buffs. See Appendix F for additional details.

4.2 IMPLEMENTED MECHANICS (V1.0)

Version 1.0 of Multiverse Mechanica includes the following mechanics, each with associated $G^{\mathcal{M}}$ and parallel-world data sufficient to estimate $\mathcal{M}$. The **shield mechanic** focuses on equipping and blocking with a shield, as discussed in Section 3.1. In the **elemental immunity mechanic**, "elemental" attributes (e.g., fire and ice) govern immunity and vulnerability to attacks. The **weapon range mechanic** governs melee vs. ranged combat. The **spell-casting mechanic** governs five submechanics that allow players to give themselves an advantage in battle (e.g., gain increased attack power, dodge ability), or their opponent a disadvantage (e.g., disarm them or lower their defense)—projectiles, self-levitation, enemy-levitation, self-transformation, and enemy-transformation. See Appendix E for detailed descriptions, including causal formalizations, DAGs, and illustrations.

4.3 Design Decisions

Mechanic-Specific Game Systems. Each mechanic is implemented with a unique instance of a *game system* (Nystrom, 2014; Gregory, 2018), independent of the others. This ensures parity between the mechanic and the code logic. Full game system details are given in Appendix F.1.1.

Impact Frames and Visual Conventions. We designed the game such that each turn contains an *impact frame*—the most visually and mechanically expressive phase of an interaction (e.g., the precise moment when an attack lands) Impact frames are not based on fixed time-points but are set according to specific conditions in the finite-state machines. See Appendix F.2 for details.

Simple yet Information Dense Visuals. To facilitate rapid, inexpensive experimentation, we focus on the ability to run experiments with episodes that have a minimal number of frames. To this end, we use a simple art style that aligns with the representational biases of pretrained vision models (luciI, 2024; Zhang et al., 2023) and animation conventions that emphasize dynamic information, such as speed lines ("zip ribbons") to depict fast motion, trajectory lines for projectiles, curved swipes for melee attacks, and burst lines and explosion visual effects for collisions or blocked strikes (McCloud, 2020; Eisner, 2008; Cohn, 2013). In Section 5, we highlight this ability by limiting our analysis to single time-point snapshots at the impact frame, chosen as the impact frame of the clip. The images in Figure 1 are all sampled at their respective impact frames.

4.4 Data Generation

Multiverse Mechanica is not a dataset but a generator. To generate data, an automated agent repeatedly plays the game to produce clips. Users can select a number N of generations. The generation process can randomly generate N clip examples, constituting level-1 data. The user can also specify interventions on specific game state variables and generate N clip examples where those interventions are applied, constituting level-2 data. Finally, the user can specify interventions and assign them to multiple game instances with a shared "ω" (same random seed and initial conditions) and generate N consistent contrasts (tuples of clips), constituting level-3 data. Each mechanic has presets for level-2 and level-3 generation. Each clip is a 512x512 MP4 video averaging 4 seconds at 50 FPS. Each generated example is a tuple consisting of a clip, controller inputs, game-state variable outcomes, and a random seed for reproducibility. See Appendix G for additional details related to data generation.

Summary. Multiverse Mechanica provides a compact yet expressive testbed for studying whether generative models can capture mechanics. Its design couples causal structure with visual grounding, leverages art and animation conventions for low-cost training, and enables reproducible creation of parallel-world contrasts.

5 Proof-of-Concept: Learning a Mechanic with Diffusion Fine-Tuning

One advantage of *Multiverse Mechanica*'s design (Section 4.3) is that mechanics are rendered into *impact frames* using simple yet information-dense visuals. This allows experiments with supervision from clips as short as a single frame. We illustrate this with a case study: fine-tuning a pretrained image diffusion model to target mechanic learning. Specifically, we fine-tune the model to learn the full mechanic set (i.e., shield, elemental immunity, weapon range, and spell-related submechanics), as discussed in Section 4.2.

We fine-tuned the latent diffusion model *OpenJourney-v4* (PromptHero, 2022) on $N = 1000$ impact frame *consistent contrasts* generated from the game. Each consistent contrast consists of paired images $(V_{X=x_0}, V_{X=x_1})$ from parallel worlds that share a random seed ω but differ by interventions on a mechanic variable X. This setup directly instantiates the causal consistency principle: non-descendant variables of X should remain invariant across the pair.

We introduce a **multiverse alignment** objective—a modification of the standard diffusion loss—to enforce the *causal consistency principle*. In a consistent contrast, all elements share a common sample $\omega \in \Omega$, mirroring the reverse process in deterministic diffusion variants that generate data

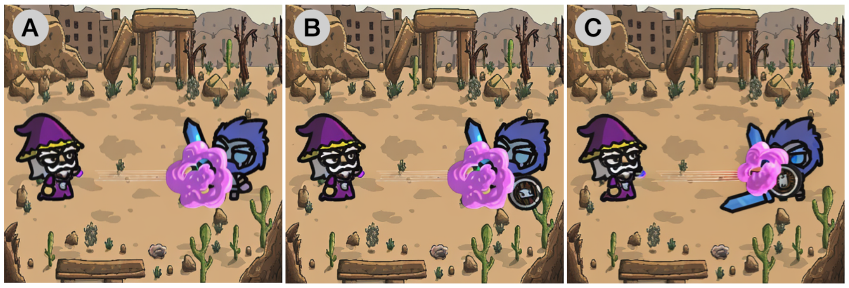


Figure 4: Proof-of-concept of shield mechanic learning with a diffusion model trained on game data. Mechanic learning is shown by sampling counterfactuals: unchanged image elements remain consistent with the original. (A) Held-out game image: wizard attacks warrior (blue due to an ice elemental buff). (B) Generated counterfactual with warrior holding a shield but not blocking ($V_{S=1,B=0}$). (C) Generated counterfactual with shield and block active ($V_{S=1,B=1}$) (aliased sword is a generation error).

from latent noise. We therefore initialize each contrast trajectory with the *same* noise: sample $\omega \sim \mathcal{N}(0, I)$ and set $Z_{T,X=x_0} = Z_{T,X=x_1} = \omega$.

Let $\{X = x_0, V_{X=x_0}, X = x_1, V_{X=x_1}\} \sim \hat{P}$, where $X \in \mathcal{X}_X$ is a game-state variable under intervention and $V_{X=x_j}$ is the corresponding impact frame snapshot under intervention $X = x_j$. With timesteps $t \in [0, T]$, let $Z_{t,X=x_0}, Z_{t,X=x_1}$ denote the noisy latent representations of $V_{X=x_0}$ and $V_{X=x_1}$, respectively. Conditioned on controller input c, the *denoiser* $\epsilon_\theta(\cdot)$ iteratively transforms the shared seed ω into a clean latent $Z_{0,X=x_j}$ that decodes into the impact frame $V_{X=x_j}$. Our task is to train the denoiser's weights θ with a loss that enforces causal consistency across contrasts.

The **multiverse alignment** loss has two components:

$$\mathcal{L} \;=\; \lambda_1 \mathcal{L}_1 \;+\; \lambda_2 \mathcal{L}_2, \quad \lambda_1, \lambda_2 \geq 0,\ \lambda_1 + \lambda_2 = 1.$$

$\mathcal{L}_1$: seed-consistency loss. Let Abduct_θ denote an inversion procedure that estimates the Gaussian seed from an observed impact-frame latent $Z_{0,X=x_j}$ and controller input c_j. Given $Z_{0,X=x_j}$ with controller input c_j, Abduct_θ uses the denoiser $\epsilon_\theta(\cdot)$ to trace $Z_{t,X=x_j}$ backward through the noise schedule, producing an estimate $\hat{\omega}_j$ of the exogenous seed ω. Under a deterministic sampler (e.g., DDIM), this corresponds to following the reverse trajectory from the observed latent to the initial noise (see Appendix I.10.5 for details). The seed-consistency loss then enforces agreement between abducted seeds across a consistent contrast:

$$\mathcal{L}_1(Z_{0,X=x_0}, c_0, Z_{0,X=x_1}, c_1) \;=\; \big\| \mathsf{Abduct}_\theta(Z_{0,X=x_0}, c_0) - \mathsf{Abduct}_\theta(Z_{0,X=x_1}, c_1) \big\|_2^2.$$

Remark. With deterministic sampling, shared ω implies shared non-descendant content. Minimizing $\mathcal{L}_1$ suppresses nuisance differences and attributes variation to mechanic-specific interventions.

$\mathcal{L}_2$: structure-alignment loss. Let $S \subset \{1, \ldots, T\}$ be a subset of high-noise timesteps. For each $t \in S$, the denoiser predicts noise $\epsilon_\theta(Z_{t,X=x_j}, t, c_j)$. We align predictions across the contrast:

$$\mathcal{L}_2 \;=\; \sum_{t \in S} \big\| \epsilon_\theta(Z_{t,X=x_0}, t, c_0) - \epsilon_\theta(Z_{t,X=x_1}, t, c_1) \big\|_2^2.$$

Remark. Early reverse steps encode coarse layout. Aligning them enforces global semantic identity across contrasts, while leaving mechanic-specific differences to emerge in later, low-noise steps.

Evaluation Results.

Figure 4 shows qualitative counterfactual generations: the model preserves non-mechanic-related content while toggling the targeted shield mechanic. From a factual image v with $X = x$ we abduct the exogenous seed ω, then generate a counterfactual image v' with $X = x'$ while keeping ω fixed. This corresponds to sampling from $P(V_{X=x'} \mid X = x, V = v)$, i.e. counterfactual reconstruction with shared ω. This provides visual evidence that the fine-tuned model has learned aspects of the mechanic, not merely pixels.

To systematize this evaluation, we report quantitative results in Table 4. The causal consistency metrics provide a systematic approach to evaluating consistency (Hessel et al., 2021; Radford et al., 2021), and we include metrics for image quality and reconstruction quality (Wang et al., 2004; Wolf et al., 2009) to serve as table-stakes baselines. While the model demonstrates reasonable image-to-text alignment (CLIP score: 24.45) and strong image similarity (0.89) between factual and counterfactual pairs, the reconstruction metrics reveal challenges in perfectly inverting the diffusion process, with reconstruction PSNR at 10.73 dB. The high exogenous distance (0.67) suggests that our current implementation requires further optimization to achieve tighter alignment between parallel worlds. Despite these limitations, the model successfully generates semantically meaningful counterfactuals, as evidenced by the transfer CLIP score of 20.77.

Table 4: Evaluation metrics for diffusion fine-tuning on 1000 consistent-contrast pairs. Causal consistency is emphasized as the key criterion for mechanic learning; reconstruction and image quality provide baseline checks.

Metric Category	**Metric**	**Value**
Causal Consistency	CF Transfer CLIP Score	20.77
	Exogenous Distance (MSE)↓	0.671
Reconstruction	Reconstruction PSNR	10.73 dB
	Reconstruction SSIM	0.183
	Reconstruction CLIP Score	21.45
Image Quality	PSNR (Factual vs CF)	17.66 dB
	SSIM (Factual vs CF)	0.433
	CLIP Image-Text Score	24.45
	CLIP Image-Image Similarity	0.892

6 Scalability & Limitations

Our descriptions of mechanics assume parallel-world statements (Section 3.1) with discrete differences across worlds; the same logic extends to incremental changes, but we have not yet characterized which mechanics require that expressivity. In our proof-of-concept, we operate on impact frame snapshots (Section 5), which makes experiments tractable but sidesteps temporal dynamics. Although our current evaluation is limited to settings with similar temporal structure, in principle, the approach applies to any generative video model capable of conditioning across worlds. As future work, we plan to benchmark video-capable world models and longer horizons.

Contrast generation scales linearly with the number of worlds per mechanic and seeds, whereas composing multiple mechanics can grow contrasts combinatorially. Our simplified 2D domain reduces compute expense and aids clarity, but limits transfer to photorealistic 3D and other genres, where new rendering conditions and longer horizons demand more complex models. Evaluation of consistency currently relies on human or VLM-based checks; the robustness of automated evaluators in this setting remains an open challenge.

Beyond video games, our causal-consistency formulation—which expresses mechanics as counterfactual constraints over an SCM and enforces cross-world consistency—extends to other controllable simulators and content-generation pipelines. Many physics engines, animation systems, and VFX toolchains expose structured state, deterministic updates, and seed-controlled stochasticity, providing the ingredients needed to construct parallel-world contrasts. In these settings, “mechanics” may correspond to physical laws, interaction rules, or rendering effects. However, realistic simulators involve high-dimensional latent state, longer temporal dependencies, and rendering stacks whose stochasticity may not align cleanly across worlds; likewise, real-world video lacks explicit multiverse structure, often requiring synthetic contrasts or approximate consistency objectives. Thus, while *Multiverse Mechanica* provides a “best-case” environment where contrasts are explicit, extending these ideas to more complex simulators or cinematic pipelines will require domain-specific approximations and strategies for generating or inferring multiverse-like structure.

7 Conclusion

We formalized game mechanics as causal counterfactual inference and introduced *Multiverse Mechanica*, a playable testbed that emits parallel-world contrasts and per-mechanic causal graphs. Building on this foundation, we proposed a multiverse-alignment objective and demonstrated a proof-of-concept fine-tuning that learns targeted mechanics. Together, these components provide a reproducible path to assessing whether world models learn *mechanics—not just pixels*.

References

Alfred V. Aho, Monica S. Lam, Ravi Sethi, and Jeffrey D. Ullman. *Compilers: Principles, Techniques, and Tools*. Addison-Wesley, Boston, MA, 2nd edition, 2006. ISBN 9780321486813.

E. Bareinboim and J. Pearl. *Causal Inference and the Data-Fusion Problem*. Cambridge University Press, 2022. ISBN 9781108844984.

Elias Bareinboim, Juan D. Correa, Duligur Ibeling, and Thomas Icard. On pearl's hierarchy and the foundations of causal inference. In Hector Geffner, Rina Dechter, and Joseph Y. Halpern (eds.), *Probabilistic and Causal Inference: The Works of Judea Pearl*, pp. 507–556. Association for Computing Machinery, New York, NY, USA, 2022. ISBN 9781450395861. doi: 10.1145/3501714.3501737.

Daniel M Bear, Elias Wang, Damian Mrowca, Felix J Binder, Hsiao-Yu Fish Tung, RT Pramod, Cameron Holdaway, Sirui Tao, Kevin Smith, Fan-Yun Sun, et al. Physion: Evaluating physical prediction from vision in humans and machines. *arXiv preprint arXiv:2106.08261*, 2021.

Eli Bingham, Jonathan P Chen, Martin Jankowiak, Fritz Obermeyer, Neeraj Pradhan, Theofanis Karaletsos, Rohit Singh, Paul Szerlip, Paul Horsfall, and Noah D Goodman. Pyro: Deep universal probabilistic programming. *Journal of machine learning research*, 20(28):1–6, 2019.

Staffan Björk and Jussi Holopainen. *Patterns in Game Design*. Charles River Media, Hingham, MA, 2004. ISBN 1-58450-354-8.

Stephan Bongers, Tineke Blom, and Joris M Mooij. Causal modeling of dynamical systems. *arXiv preprint arXiv:1803.08784*, 2018.

Jake Bruce, Michael D Dennis, Ashley Edwards, Jack Parker-Holder, Yuge Shi, Edward Hughes, Matthew Lai, Aditi Mavalankar, Richie Steigerwald, Chris Apps, et al. Genie: Generative interactive environments. In *Forty-first International Conference on Machine Learning*, 2024.

Haoxuan Che, Xuanhua He, Quande Liu, Cheng Jin, and Hao Chen. Gamegen-x: Interactive open-world game video generation. In *International Conference on Learning Representations*, 2025. URL `https://openreview.net/forum?id=8VG8tpPZhe`.

Z. Chen, T. Xu, Y. Zhang, et al. Model as a game: On numerical and spatial consistency for generative games. *arXiv preprint arXiv:2503.21172*, 2025. URL `https://arxiv.org/abs/2503.21172`.

Neil Cohn. *The Visual Language of Comics: Introduction to the Structure and Cognition of Sequential Images*. Bloomsbury Advances in Semiotics. Bloomsbury Academic, London, 2013. ISBN 9781472542175.

Decart, Julian Quevedo, Quinn McIntyre, Spruce Campbell, Xinlei Chen, and Robert Wachen. Oasis: A universe in a transformer. `https://oasis-model.github.io/`, 2024. Accessed: 2025-09-24.

Joris Dormans. *Engineering Emergence: Applied Theory for Game Design*. PhD thesis, University of Amsterdam, 2012. URL `https://eprints.illc.uva.nl/id/eprint/2118/1/DS-2012-12.text.pdf`.

Will Eisner. *Comics & Sequential Art*. W. W. Norton & Company, New York, revised edition edition, 2008. ISBN 9780393331264. First published 1985; expanded edition 1990; revised edition 2008.

Robin J Evans. Graphs for margins of bayesian networks. *Scandinavian Journal of Statistics*, 43(3): 625–648, 2016.

Carlo Fabricatore. Gameplay and game mechanics: A key to quality in videogames. In *Proc. OECD Expert Meeting on Videogames and Education*, 2007. URL `https://eprints.hud.ac.uk/id/eprint/20927/`.

Tracy Fullerton, Chris Swain, and Steven Hoffman. *Game design workshop: Designing, prototyping, & playtesting games*. CRC Press, 2004.

J. Gingerson, S. Amershi, et al. World and human action models towards gameplay ideation. *Nature*, 2024. URL `https://www.nature.com/articles/wham2024`. Microsoft Research technical report also available.

Clark Glymour, Kun Zhang, and Peter Spirtes. Review of causal discovery methods based on graphical models. *Frontiers in genetics*, 10:524, 2019.

Jason Gregory. *Game Engine Architecture*. A K Peters/CRC Press, Boca Raton, FL, 3 edition, 2018. ISBN 9781138035454.

David Ha and Jürgen Schmidhuber. World models. *arXiv preprint arXiv:1803.10122*, 2018. URL `https://arxiv.org/abs/1803.10122`.

Xianglong He, Chunli Peng, Zexiang Liu, Boyang Wang, Yifan Zhang, Qi Cui, Fei Kang, Biao Jiang, Mengyin An, Yangyang Ren, Baixin Xu, Hao-Xiang Guo, Kaixiong Gong, Cyrus Wu, Wei Li, Xuchen Song, Yang Liu, Eric Li, and Yahui Zhou. Matrix-game 2.0: An open-source, real-time, and streaming interactive world model. *arXiv preprint arXiv:2508.13009*, 2025.

Mariya Hendriksen, Tabish Rashid, David Bignell, Raluca Georgescu, Abdelhak Lemkhenter, Katja Hofmann, Sam Devlin, and Sarah Parisot. Adapting vision-language models for evaluating world models. *arXiv preprint arXiv:2506.17967*, 2025.

Jack Hessel, Ari Holtzman, Maxwell Forbes, Ronan Le Bras, and Yejin Choi. Clipscore: A reference-free evaluation metric for image captioning. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 7514–7528, 2021. doi: 10.18653/v1/2021.emnlp-main.595.

Paul W. Holland. Statistics and causal inference. *Journal of the American Statistical Association*, 81(396):945–960, 1986. doi: 10.2307/2289064.

Charles Tapley Hoyt, Craig Bakker, Richard J. Callahan, Joseph Cottam, August George, Benjamin M. Gyori, Haley M. Hummel, Nathaniel Merrill, Sara Mohammad Taheri, Pruthvi Prakash Navada, Marc-Antoine Parent, Adam Rupe, Olga Vitek, and Jeremy Zucker. Causal Identification with Y_0, 2025. URL `https://doi.org/10.48550/arXiv.2508.03167`. arXiv:2508.03167.

Robin Hunicke, Marc LeBlanc, and Robert Zubek. MDA: A formal approach to game design and game research. In *Proc. AAAI Workshop on Challenges in Game AI*, 2004. URL `https://aaai.org/papers/ws04-04-001-mda-a-formal-approach-to-game-design-and-game-research/`.

Aki Järvinen. *Games without Frontiers: Theories and Methods for Game Studies and Design*. PhD thesis, University of Tampere, 2008. URL `https://ocw.metu.edu.tr/pluginfile.php/4468/mod_resource/content/0/ceit706/week3_new/AkiJarvinen_Dissertation.pdf`.

D. Kang, Y. Wu, et al. How far is video generation from world models? *arXiv preprint arXiv:2402.19014*, 2024. URL `https://arxiv.org/abs/2402.19014`.

Tero Karras, Miika Aittala, Timo Aila, and Samuli Laine. Elucidating the design space of diffusion-based generative models. *Advances in neural information processing systems*, 35:26565–26577, 2022.

Seung Wook Kim, Yuhao Zhou, Jonah Philion, Antonio Torralba, and Sanja Fidler. Learning to simulate dynamic environments with gamegan. In *Proc. IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020. URL `https://openaccess.thecvf.com/content_CVPR_2020/papers/Kim_Learning_to_Simulate_Dynamic_Environments_With_GameGAN_CVPR_2020_paper.pdf`.

Priscilla Lo, David Thue, and Elin Carstensdottir. What is a game mechanic? In *International Conference on Entertainment Computing*, pp. 336–347. Springer, 2021.

F. Locatello, S. Bauer, M. Lucic, G. Raetsch, S. Gelly, B. Schölkopf, and O. Bachem. Challenging common assumptions in the unsupervised learning of disentangled representations. In *Proceedings of the 36th International Conference on Machine Learning (ICML)*, 2019. URL `https://arxiv.org/abs/1811.12359`.

luciI. Chibi artstyle model. `https://civitai.com/models/22820/chibi-artstyle`, 2024. Community submission on Civitai. Accessed: 2025-09-24.

Sus Lundgren and Staffan Bjork. Game mechanics: Describing computer-augmented games in terms of interaction. In *Proceedings of TIDSE*, volume 3, 2003.

Scott McCloud. *Understanding Comics: The Invisible Art*. Harper Perennial, reprint edition edition, 2020. ISBN 9780060976255.

T. M. Mitchell. *The Need for Biases in Learning Generalizations*. Technical Report CBM-TR-117, Rutgers University, 1980.

Miguel Monteiro, Fabio De Sousa Ribeiro, Nick Pawlowski, Daniel C Castro, and Ben Glocker. Measuring axiomatic soundness of counterfactual image models. *arXiv preprint arXiv:2303.01274*, 2023.

Robert Nystrom. *Game Programming Patterns*. Genever Benning, 2014. ISBN 9780990582908.

Jack Parker-Holder and Shlomi Fruchter. Genie 3: A new frontier for world models. Google DeepMind Blog, aug 2025. URL `https://deepmind.google/discover/blog/genie-3-a-new-frontier-for-world-models/`. Accessed 24 September 2025.

Judea Pearl. *Causality*. Cambridge university press, 2009.

Judea Pearl. On the consistency rule in causal inference: axiom, definition, assumption, or theorem? *Epidemiology*, 21(6):872–875, 2010.

PromptHero. Openjourney v4. `https://huggingface.co/prompthero/openjourney-v4`, 2022. Diffusion model hosted on Hugging Face.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning transferable visual models from natural language supervision. In *Proceedings of the 38th International Conference on Machine Learning (ICML)*, volume 139, pp. 8748–8763. PMLR, 2021. URL `http://proceedings.mlr.press/v139/radford21a.html`.

J. Riochet, H. Le Borgne, E. Ricci, et al. Intphys 2: A benchmark for physical consistency in video prediction. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2021. URL `https://arxiv.org/abs/2107.06227`.

Ronan Riochet, Mario Ynocente Castro, Mathieu Bernard, Adam Lerer, Rob Fergus, Véronique Izard, and Emmanuel Dupoux. Intphys: A framework and benchmark for visual intuitive physics reasoning. *arXiv preprint arXiv:1803.07616*, 2018.

Tom Schaul. A video game description language for model-based or interactive learning. In *Proceedings of the IEEE Conference on Computational Intelligence in Games*, 2013. URL `https://schaul.site44.com/publications/pyvgdl.pdf`.

B. Schölkopf, F. Locatello, S. Bauer, N. R. Ke, N. Kalchbrenner, A. Goyal, and Y. Bengio. Towards causal representation learning. *Proceedings of the IEEE*, 109(5):612–634, 2021. doi: 10.1109/JPROC.2021.3058954.

Ilya Shpitser and Judea Pearl. What counterfactuals can be tested. *arXiv preprint arXiv:1206.5294*, 2012.

P. Spirtes, C. Glymour, and R. Scheines. *Causation, Prediction, and Search*. MIT Press, 2nd edition, 2000. ISBN 9780262194402.

Michael Thielscher. The general game playing description language is universal. In *Proceedings of the 22nd International Joint Conference on Artificial Intelligence (IJCAI)*, 2011. URL `https://www.ijcai.org/Proceedings/11/Papers/189.pdf`.

Zhou Wang, Alan C. Bovik, Hamid R. Sheikh, and Eero P. Simoncelli. Image quality assessment: From error visibility to structural similarity. *IEEE Transactions on Image Processing*, 13(4): 600–612, 2004. doi: 10.1109/TIP.2003.819861.

Glynn Winskel. Event structures. In *Advanced course on Petri nets*, pp. 325–392. Springer, 1986.

Stephen Wolf, Margaret Pinson, et al. Reference algorithm for computing peak signal to noise ratio (psnr) of a video sequence with a constant delay. Technical Report NTIA/ITS / COM-9 C.6, National Telecommunications and Information Administration (NTIA) / ITS, 2009. Proposed reference implementation and calibration procedures for PSNR.

D. H. Wolpert. The lack of a priori distinctions between learning algorithms. *Neural Computation*, 8(7):1341–1390, 1996. doi: 10.1162/neco.1996.8.7.1341.

Lvmin Zhang, Anyi Rao, and Maneesh Agrawala. Adding conditional control to text-to-image diffusion models. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 3813–3824. IEEE, 2023. ISBN 9798350307184. doi: 10.1109/ICCV51070.2023.00355.

Alexander Zook and Mark Riedl. Automatic game design via mechanic generation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 28, 2014.

Alexander Zook and Mark O. Riedl. Automatic game design via mechanic generation. *arXiv preprint arXiv:1908.01420*, 2019. URL `https://arxiv.org/abs/1908.01420`.

A Use of Large Language Models (LLMs)

GPT-5 was used for editing and polishing during the writing of this paper. The LLM was not used to make any technical or scientific contributions to the paper writing process; for example: writing the literature review or background sections, creating citations, or analyzing data.

B Ethics Statement

There are no ethics concerns raised by this paper.

C Reproducibility Statement

To ensure reproducibility of this paper, we make the following efforts:

- We will release Multiverse Mechanica upon publication to provide researchers a game for use as a testbed for evaluating learning of game mechanics.
- We will make available publicly upon publication the dataset (including mechanism-specific graphical artifacts) used for the proof-of-concept training presented in Section 5.

C.1 Proof-of-Concept Dataset

Here, we describe the dataset used for the proof-of-concept training presented in Section 5, used to learn the **shield gameplay mechanic**.

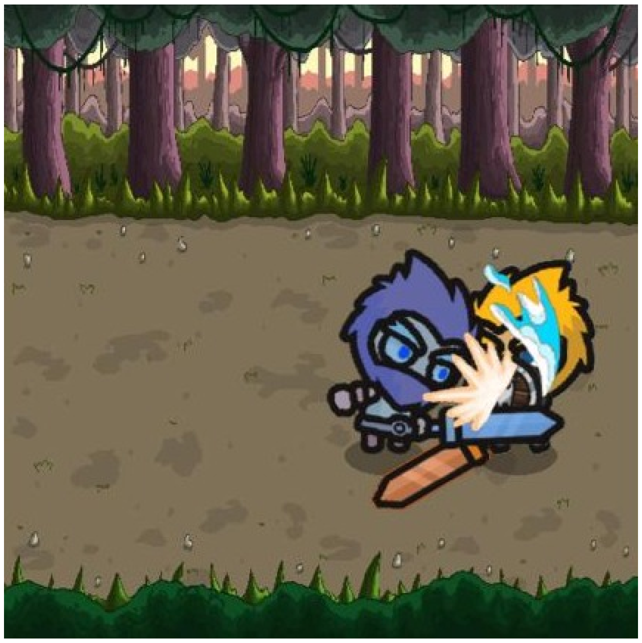


Figure 5: Impact frame generated by Multiverse Mechanica for a world in a parallel world tuple.

C.1.1 Generation

We generate a level-3 dataset, as described in Section 4.4. We automatically derive parallel world interventions based on the full game DAG, targeting the variables relevant to the parallel world contrast statements for the shield mechanic. The set of derived interventions define multiple parallel world tuples, enumerated such that we sufficiently cover the support of joint distribution of the mDAG induced by the query variables associated with the level-3 parallel world statements for the mechanic.

We assign these interventions to multiple game instances with a shared "ω" (same random seed and initial conditions) and generate $N = 1000$ consistent contrasts, constituting level-3 data. Since we are training on text and images (i.e., not gameplay clips), for each we generate only a subset of training artifacts consisting of these two modalities: 1) the impact frame as a 512x512 PNG image, and 2) the game-state variable outcomes (converted into a caption in the pre-processing method, outlined below).

C.1.2 Pre-Processing

To align our artifact modalities with the text-to-image latent diffusion architecture used in our proof-of-concept, we must perform a pre-processing step to convert from the game-state variable outcomes—a dictionary of variable-name/value pairs—to a text caption. We implement a templated captioning process for each parallel world tuple, by which plain-text captions are deterministically constructed based on the values of the game-state variables.

Consider, for example, the following impact frame image and game-state variable artifacts generated by Multiverse Mechanica for one of the worlds in a parallel world tuple.

For this world, Multiverse Mechanica has generated the impact frame image shown in Figure 5 and the following game-play state:

```
gameplay_state = {
    'background': 'forest',
    'player action': 'melee attack',
    'player class': 'warrior',
    'player does block': False,
    'player element': 'ice',
    'player has shield': False,
    'player is hurt': False,
    'player is immune to attack': False,
    'player weapon': 'short sword',
    'player weapon class': 'light',
    'player weapon element': 'ice',
    'player weapon is light': True,
    'player weapon range': 'melee',
    'opponent action': 'defend',
    'opponent class': 'warrior',
    'opponent does block': True,
    'opponent element': 'none',
    'opponent has shield': True,
    'opponent is hurt': False,
    'opponent is immune to attack': False,
    'opponent weapon': 'short sword',
    'opponent weapon class': 'light',
    'opponent weapon element': 'fire',
    'opponent weapon is light': True,
    'opponent weapon range': 'melee',
}
```

Given this game-play state, our pre-processing step constructs the following caption:

> "A 1-on-1 battle between two warriors. The warrior on the left has a ice element buff. The warrior on the left's weapon is a short sword with a ice buff. The warrior on the left is launching a melee attack. The warrior on the right's weapon is a short sword with a fire buff and they have a shield. The warrior on the right is blocking with their shield."

The captions generated by the pre-processing step is combined with the generated image to thus create training data suitable for text-to-image latent diffusion architecture used in our proof-of-concept.

D Brief Primer on Core Causal Concepts

D.1 The Causal Hierarchy

The *causal hierarchy* describes three levels of statements that employ causal logic: (level-1) observation, (level-2) intervention, and (level-3) counterfactual (Bareinboim et al., 2022). A predictive statement, such as "The scene would generate in *this* way," is a level-1 observational statement. A level-2 interventional statement targets outcomes under hypothetical interventions, such as "If an enemy were placed here (e.g., in a place it would not naturally appear), then the scene would generate in *this* way." A level-3 counterfactual statement considers observed outcomes and how those outcomes might have been different under hypothetical interventions. For example: "Given there was no enemy and the scene generated that way, if an enemy were placed here, the scene would generate in *this* way."

Formally, level-k statements describe events in the sample space of a level-k distribution (Bareinboim et al., 2022). Level-k data can be viewed as samples from such a distribution, and with i.i.d. sampling, the empirical distribution converges to the true sampling distribution. Ordinary gameplay logs are level-1 data; data where variables are artificially fixed before sampling (as in experiments) is level-2 data. In most settings, level-3 data does not exist due to the *fundamental problem of causal inference*—it is impossible to observe outcomes for the same effect variables across worlds (Holland, 1986). In this work, we leverage the video game setting's ability to observe level-3 data across *virtual worlds*, i.e., multiple instances of a game run with shared initial conditions.

Models also align with this hierarchy. A causal DAG is a level-2 model: when combined with a generative model, it encodes the family of interventional distributions over the DAG's variables. A structural causal model (SCM) is a level-3 model: it additionally encodes the family of counterfactual distributions over the DAG's variables (Pearl, 2009). Level-k data is sufficient to identify level-k models, but the *causal hierarchy theorem* states that level-k statements cannot, in general, be inferred from data below level k (Bareinboim et al., 2022).

Thus, we follow the three-level hierarchy (Bareinboim et al., 2022): (L1) *observational* questions about $P(\mathbf{V})$; (L2) *interventional* questions about $P(\mathbf{V} \mid \mathrm{do}(X{=}x))$; and (L3) *counterfactual* (parallel-world) questions about joint outcomes under conflicting interventions, e.g., $(Y_{X=x}, Y_{X=x'})$. Level-3 quantities encode "all-else-equal" comparisons across worlds.

D.2 SCMs and Rules

An SCM $\mathcal{M} = (\mathbf{U}, \mathbf{V}, \mathbf{F}, P(\mathbf{U}))$ specifies exogenous factors $\mathbf{U}$, endogenous variables $\mathbf{V}$, structural assignments $\mathbf{F}$, and a distribution over $\mathbf{U}$ (Pearl, 2009). In our context, $\mathbf{F}$ plays the role of executable rules; see Bongers et al. (2018) for SCMs as formal rule systems.

D.3 Marginalization of Causal DAGs

We implement Evans (2016) approach to DAG marginalization. Given a causal DAG and a set of nodes to marginalize out of the DAG, Evans (2016) creates a marginalized DAG with *hyper-edges* that represent the footprint of latent common causes. Directed edges encode parent–child relationships among observed variables as usual. Hyper-edges encompassing a set of observed nodes, indicate that all nodes in the set share an unobserved exogenous influence. For example, a hyper-edge touching nodes X, Y, Z represents the fact that there is some unobserved variable that acts as a common cause to X, Y, Z simultaneously. The mDAG maintains the causal interpretation, intervention model, and implications to conditional independence as the full DAG.

While mDAGs are convenient for reasoning about marginalized structures, we require a fully explicit DAG representation for generating parallel-world and counterfactual graphs, as well as for using standard graph serialization. We therefore modify the algorithm such that, for each hyperedge, a d-separating sets of shared ancestors that entail the hyperedge are explicitly added back into the model. This implements the expansion of the canonical mDAG by making the latent causes explicit expansion as discussed in Evans (2016).

D.4 Multiverse Reasoning.

With *multiverse* counterfactual reasoning, we envision one world where observed outcomes occurred, and separate "parallel" worlds where hypothetical interventions lead to outcomes that differ from the observed outcomes Shpitser & Pearl (2012). For example, consider the statement "Given there was no enemy and the scene generated *that* way, if an enemy were placed here, the scene would generate in *this* way". With this statement, we can envision two parallel worlds, one where there was no enemy and the scene generated *that* way, and one where there was an enemy and the scene generated *this* way. Level-3 data, therefore, is a data tuple representing outcomes across parallel worlds. The "fundamental problem of causal inference" (Holland, 1986) is that level-3 data is unobservable—in real world settings, it is impossible to observe potential outcomes for the same variable across parallel worlds. A key insight of our work is that in *virtual world* settings, the data can exist by creating parallel instances of the same world with the same initial conditions.

D.5 Counterfactual Graphs and Consistency

Counterfactual graphs are based on *parallel world graphs*. The parallel world graph clones a causal DAG across parallel worlds and uses graph surgery to represent hypothetical conditions in certain worlds (Shpitser & Pearl, 2012). Variables that are *not* descendants of the intervention share a single node across worlds (consistency); variables affected by the intervention are duplicated and indexed by world. This graph encodes which quantities must remain identical across worlds and which may differ, providing a compact template for supervision and evaluation. The counterfactual graph collapses nodes that must be consistent across worlds into single nodes, creating a graph that (unlike the parallel world graph) encodes conditional independence across parallel worlds. In this work, we formalize the concept of game mechanics such that, for a given mechanic, we can generate a set of parallel world graphs and counterfactual graphs that explicitly encode its structure and which variables should remain consistent while the mechanic is in play. These can be used in training alongside the level-3 gameplay data.

D.6 Estimating Counterfactual Distributions

In Section 3.1.3, we let $P_1 = P(S_{W=1}, S_{W=0})$, $P_2 = P(B_{S=1}, B_{S=0})$, and $P_3 = P(D_{B=1}, D_{B=0}, S)$ denote the distributions constrained by $\mathcal{S}_1$, $\mathcal{S}_2$, and $\mathcal{S}_3$. We can estimate these distributions with repeated sampling of consistent contrasts $\mathcal{D}_1$, $\mathcal{D}_2$, and $\mathcal{D}_3$, then averaging over sampling distributions to obtain sampling distributions $\hat{P}_1$, $\hat{P}_1$, and $\hat{P}_3$.

- $\hat{P}_1$: $\sum_{j\in 0,1} \sum_{b^{(i)}_{W=j}, d^{(i)}_{W=j}, v^{(i)}_{W=j}} \hat{P}\big(S_{W=1}, B_{W=1}, D_{W=1}, V_{W=1}, S_{W=0}, B_{W=0}, D_{W=0}, V_{W=0}\big)$
- $\hat{P}_2$: $\sum_{j\in 0,1} \sum_{w, d_{S=j}, v_{S=j}} \hat{P}\big(W, B_{S=1}, D_{S=1}, V_{S=1}, B_{S=0}, D_{S=0}, V_{S=0}\big)$
- $\hat{P}_3$: $\sum_{j\in 0,1} \sum_{w, s, v_{B=j}} \hat{P}\big(W, S, D_{B=1}, V_{B=1}, D_{B=0}, V_{B=0}\big)$

In each case $\hat{P}_i$ converges almost surely to P_i.

E Mechanics Implemented in Multiverse Mechanica v1.0

In this section, we describe the ground truth causal structure and mechanics Multiverse Mechanica.

We describe each game mechanic in terms of:

- *level-2* interventional statements: A set of causal hypothetical statements of the form "Given preconditions W, all else equal, if X, then Y."
- A set of causal Markov kernels encompassing the relevant conditional probability distributions in the causal DAG.
- *level-3 parallel world* statements: A set of counterfactual statements of the form "Given preconditions W, all else equal, if X, then Y AND if X' then Y'."
- A set of counterfactual probability expressions defining the induced counterfactual outcome probabilities.

Particularly, in the case of the level-3 parallel world statements, we can enumerate counterfactual worlds in which we make a change and the target outcome variable(s) change, with guarantees on consistency, with respect to the game mechanic.

We formulate the counterfactual cases by taking interventions to change variables (**bold**), from the factual case (Case 0). Downstream changes are shown in *italics*.

Full Causal DAG Figure 6 shows the full causal DAG of a turn in Multiverse Mechanica v1.0.

E.1 Shield Mechanic

The warrior character can equip a shield they can use to block incoming attacks, if they have a free hand and perform the "defend" action.

E.1.1 level-2 Interventional Definition

We can completely define this mechanic by a set of contrasting statements:

- If a player has a small weapon, they may hold a shield, otherwise they cannot.
- If a player has a shield, they may block incoming attacks, otherwise they cannot.
- If a player blocks an incoming attack, they avoid taking damage, otherwise they may take damage.

Involved variables:

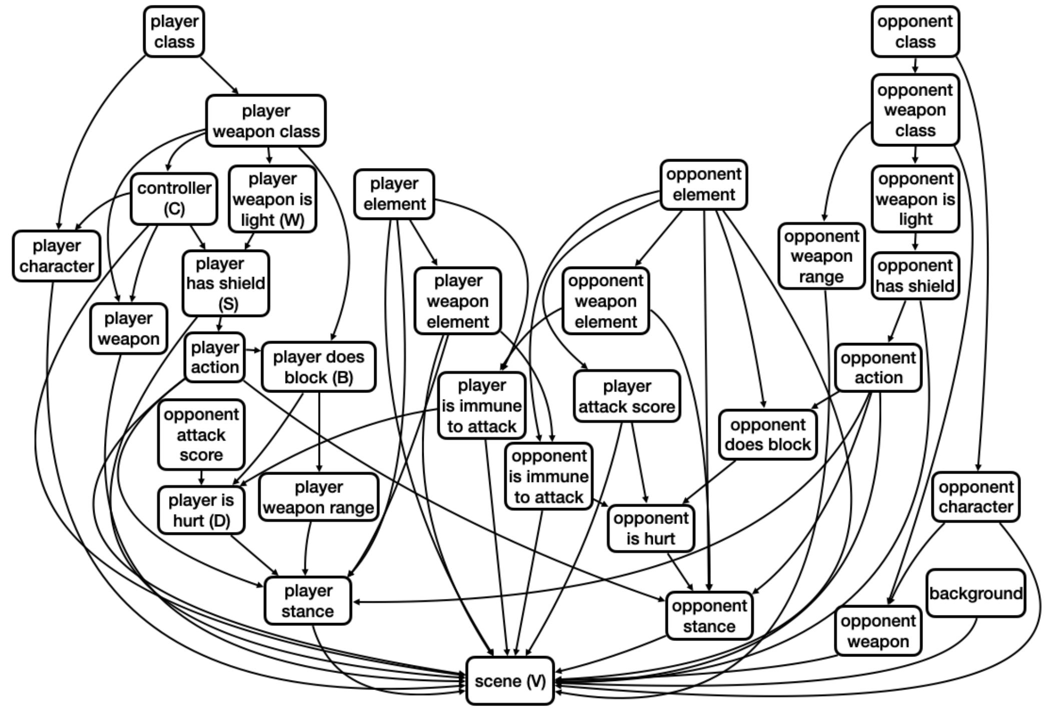
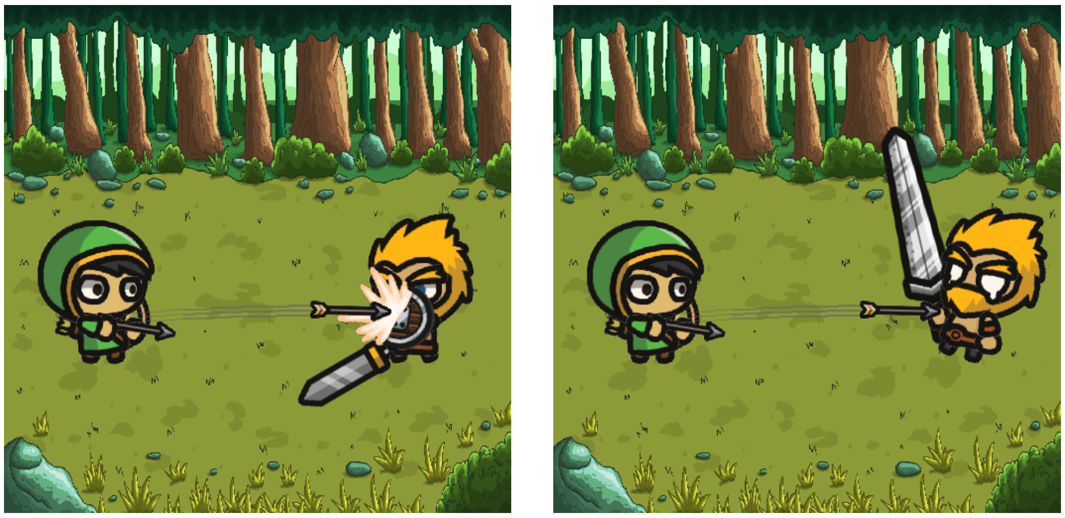


Figure 6: Full causal DAG of a turn in Multiverse Mechanica



(a) Factual: The warrior is equipped with a one-handed short-sword and a shield, and blocks the arrow with a shield.

(b) Counterfactual: The warrior is equipped with a two-handed long-sword, and no shield, and is unable to block the attack.

Figure 7: Example counterfactual contrast statement for the shield mechanic.

- `player action` : [idle, melee attack, defend]
- `opponent weapon class` : [small, large]
- `opponent has shield` : [True, False]
- `opponent action` : [idle, melee attack, defend]
- `opponent is hurt` : [True, False]

Latent variables:

- `opponent weapon is heavy` : [one-handed, two-handed]

E.1.2 Causal Markov Kernels

In our model, *block* is one of the actions. Thus, we can reason about the shield game mechanic in terms of the player/opponent action being sampled as *block*.

The causal Markov kernel for the opponent action variable encompasses the shield game mechanic:

$$P\big(\text{opponent action} \,\big|\, \text{opponent has shield, opponent weapon}\big).$$

To reason about the opponent blocking, we consider the probability that the action variable is *block*:

$$P\big(\text{opponent action} = \text{block} \,\big|\, \text{opponent has shield, opponent weapon}\big).$$

To reason about the opponent getting hurt, we consider the probability that the opponent gets hurt, given the opponent action and player action:

$$P\big(\text{opponent is hurt} \,\big|\, \text{opponent action, opponent has shield, opponent weapon}\big).$$

An important detail to take note of is that the causal DAG does not have a variable to represent the character's outcome (e.g. getting hit, blocking, dodging)—it only has a notion of final stance. However, the stance variable is currently unused after sampling; the battle video game handles the underlying outcome logic and renders the scene.

E.1.3 level-3 Counterfactual Parallel World Statements

We can rewrite the level-2 interventional statements as the following level-3 parallel world statements:

1. Given a character had a heavy weapon and they did not have a shield; if the character would have had a light weapon instead, then they could have equipped a shield.
2. Given a character had not had a shield and did not block; if the character had a shield instead, then they could have blocked.
3. Given a character had a shield but did not block and took damage; if the character had blocked instead, then they would have avoided taking damage from the attack.

E.1.4 Counterfactual Probability Expressions

We can formulate these natural-language statements in mathematical notation.

Notation. Let:

- WC = weapon class,
- HS = has shield,
- B = does block,
- H = is hurt.

Then we have:

$$P\big(HS_{WC=\text{Light}} = \text{True} \,\big|\, WC = \text{Heavy},\ HS = \text{False}\big) > 0,$$

$$P\big(B_{HS=\text{True}} = \text{True} \,\big|\, HS = \text{False},\ B = \text{False}\big) > 0,$$

$$P\big(H_{B=\text{True}} = \text{False} \,\big|\, HS = \text{True},\ B = \text{False},\ H = \text{True}\big) = 1.$$

E.2 Elemental Immunity Mechanic

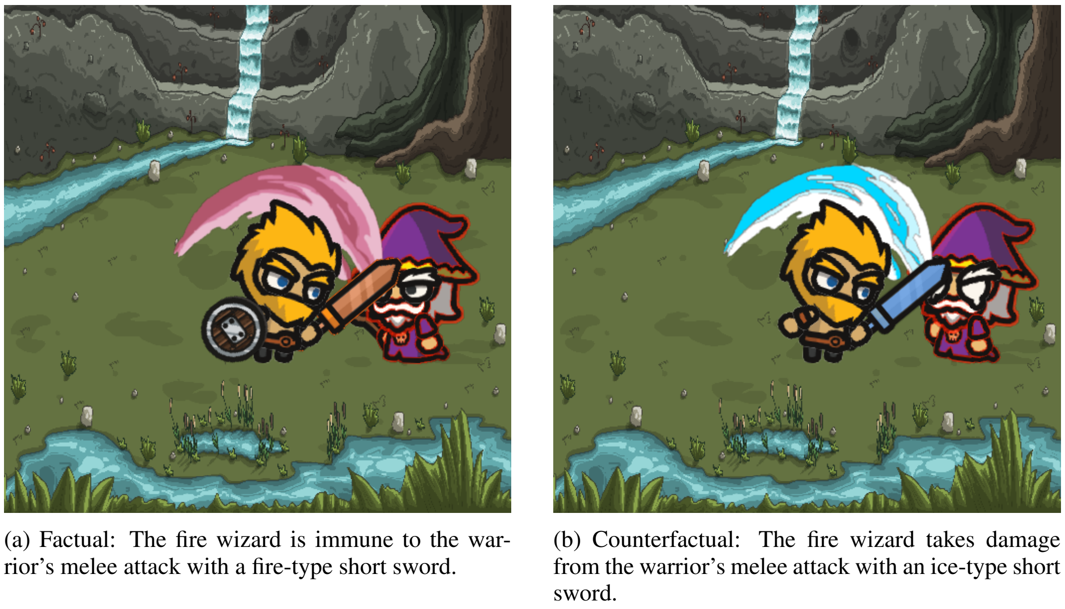
Under the elemental immunity game mechanic, players may have elemental attributes; either fire or ice. Weapons may also have elemental attributes. If the attacking player weapon element and opponent element are the same (and non-none), the opponent is granted elemental immunity, and thus avoids taking damage from the attack, since they are already imbued with the element the attacker is using to attack them.

The elemental attributes of characters are indicated by variations from their base appearance:

- *none*: Characters appear in their standard appearances
- *fire*: Characters appear with orange shading and/or red outline
- *ice*: Characters appear with light blue shading and/or blue outline

The elemental attributes of weapons are similarly indicated by variations from their base appearance:

- *none*: Weapons appear in their standard appearances
- *fire*: Weapons appear with orange shading and/or red outline. Melee weapon slash visual effects appear with red shading. Ranged weapon projectiles (e.g., arrows, spell attacks) and explosions appear with red shading.
- *ice*: Weapons appear with light blue and/or blue outline. Melee weapon slash visual effects appear with blue shading. Ranged weapon projectiles and explosions appear with blue shading.


(a) Factual: The fire wizard is immune to the warrior's melee attack with a fire-type short sword.

(b) Counterfactual: The fire wizard takes damage from the warrior's melee attack with an ice-type short sword.

Figure 8: Example counterfactual contrast statement for the elemental immunity mechanic.

E.2.1 level-2 Interventional Definition

We can completely define this mechanic by a set of contrasting statements:

- If a player has an elemental attribute, they may wield a weapon with the same elemental attribute but they cannot wield one of an opposing element in the fire–ice dichotomy. For example, a player with a fire-elemental attribute may wield a fire-elemental weapon (or non-elemental weapon); they cannot wield an ice-elemental weapon.
- If a player does not have an elemental attribute, they may wield any elemental-type weapon; otherwise the rule above applies.

- If a player is hit by an incoming attack from a weapon with the same elemental attribute, they are immune and thus avoid taking damage; otherwise they may take damage.

Involved variables:

- `player element` : [none, fire, ice]
- `player weapon element` : [none, fire, ice]
- `opponent element` : [none, fire, ice]
- `opponent is immune to attack` : [True, False]
- `opponent is hurt` : [True, False]

Latent variables: None.

E.2.2 Causal Markov Kernels

The element of the player and the opponent are both independent variables:

$$P(\text{player element}),$$
$$P(\text{opponent element}).$$

The causal Markov kernel for the player weapon elemental is as follows:

$$P\big(\text{player weapon element} \mid \text{player element}\big).$$

To reason about the opponent being immune to the attack, we consider the conditional probability given the opponent's element and player's weapon element:

$$P\big(\text{opponent is immune to attack} \mid \text{player weapon element, opponent element}\big).$$

To reason about the opponent getting hurt, we consider the probability that the opponent gets hurt, given the opponent being immune:

$$P\big(\text{opponent is hurt} \mid \text{opponent is immune to attack}\big).$$

E.2.3 level-3 Counterfactual Parallel World Statements

We can rewrite the level-2 interventional statements as the following level-3 parallel world statements:

1. Given a character had a fire elemental attribute and they had a fire elemental weapon; if the character would have had an ice elemental attribute instead, then they could have had an ice- or none-elemental weapon but not a fire elemental.
2. Given a character had a fire elemental attribute and they had a fire elemental weapon; if the character would have had no elemental attribute instead, then they could have had any elemental or non-elemental type weapon.
3. Given a character had an ice elemental attribute and was hit with an attack from an ice elemental weapon, and thus was immune and did not take damage; if the character had a fire elemental attribute instead, then they would have taken damage from the attack.

E.2.4 Counterfactual Probability Expressions

We can formulate these natural language statements in mathematical notation.

Notation. Let:

- $PE =$ player element,
- $PWE =$ player weapon element,
- $OE =$ opponent element,
- $I =$ is immune to attack,
- $H =$ is hurt.

Then we have:

Statement 1.

$$P\big(PWE_{PE=\text{Ice}} = \text{Fire} \,\big|\, PE = \text{Fire},\ PWE = \text{Fire}\big) = 0,$$

$$P\big(PWE_{PE=\text{Ice}} = \text{Ice} \,\big|\, PE = \text{Fire},\ PWE = \text{Fire}\big) > 0,$$

$$P\big(PWE_{PE=\text{Ice}} = \text{None} \,\big|\, PE = \text{Fire},\ PWE = \text{Fire}\big) > 0.$$

Statement 2.

$$P\big(PWE_{PE=\text{None}} = \text{Fire} \,\big|\, PE = \text{Fire},\ PWE = \text{Fire}\big) > 0,$$

$$P\big(PWE_{PE=\text{None}} = \text{Ice} \,\big|\, PE = \text{Fire},\ PWE = \text{Fire}\big) > 0,$$

$$P\big(PWE_{PE=\text{None}} = \text{None} \,\big|\, PE = \text{Fire},\ PWE = \text{Fire}\big) > 0.$$

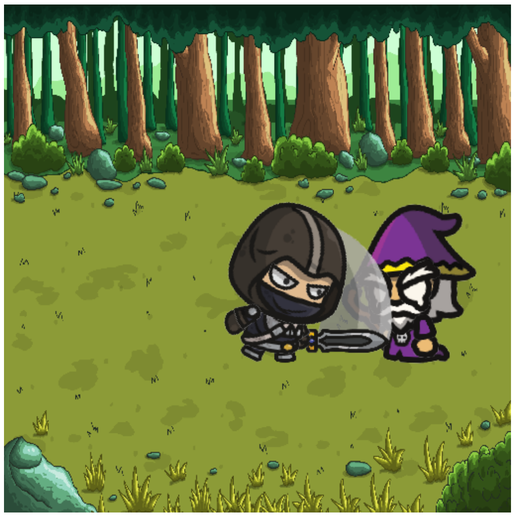
Statement 3.

$$P\big(H_{OE=\text{Fire}} = \text{True} \,\big|\, PWE = \text{Ice},\ OE = \text{Ice}\big) = 1.$$

E.3 Weapon Range Mechanic

Under the weapon range game mechanic, weapon classes have range attributes; either melee or ranged. Each weapon class is either a melee or ranged type. For example, the light sword, the heavy sword, and the dagger are melee weapons; while the bow, the staff, and the throwing knife are ranged weapons. The weapon range attribute affects primarily visual elements of the battle scene:

1. **Stance:** a character attacking with a melee weapon will be depicted in the snapshot physically swinging the melee weapon at the opponent's location, while a character attacking with a ranged weapon will be depicted shooting/throwing/casting from their position.
2. **Scene:** a character attacking with a melee weapon will be shown in the game scene (video) first approaching the opponent's position before physically swinging the melee weapon, while a character attacking with a ranged weapon will be depicted shooting/throwing/-casting from their position and the emitted projectile will be shown traveling towards the opponent from left to right (possibly on a parabolic trajectory, if affected by gravity, e.g. the arrow shot from the bow).


(a) Factual: The assassin performs a melee attack on a wizard with a sword.

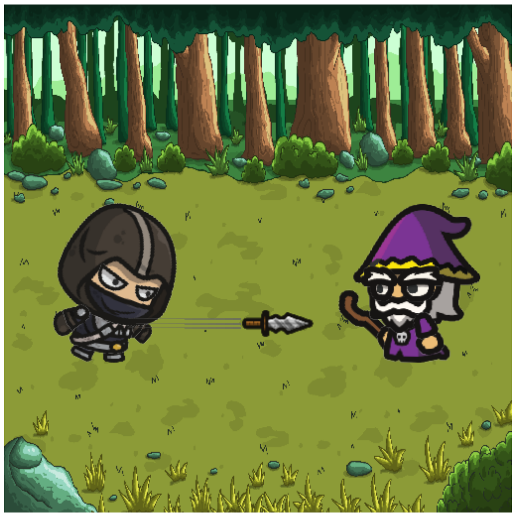

(b) Counterfactual: The assassin performs a ranged attack on a wizard with a throwing knife.

Figure 9: Example counterfactual contrast statement for the weapon range mechanic.

E.3.1 LEVEL-2 INTERVENTIONAL DEFINITION

We can completely define this mechanic by a set of contrasting statements:

- If a weapon is one of the weapon classes *light sword*, *heavy sword*, or *dagger*, it is a melee weapon; otherwise if it is *bow*, *staff*, or *throwing knife* it is a ranged weapon.
- If a player wields a melee weapon, they will be depicted in the snapshot as performing a melee attack at the opponent's location; otherwise they will be depicted as performing a ranged attack from their own location.
- If a player wields a melee weapon, they will be shown in the game scene (video) first approaching the opponent's position before physically swinging the melee weapon; otherwise they will be depicted shooting/throwing/casting from their position and the emitted projectile will be shown traveling towards the opponent from left to right.

Involved variables:

- `player weapon class` : [light sword, heavy sword, dagger, bow, staff, throwing knife]
- `player weapon range` : [melee, ranged]
- `player stance` : pixels in the rendered image snapshot
- `scene` : pixels in the rendered gameplay video

Latent variables: None.

E.3.2 CAUSAL MARKOV KERNELS

The causal Markov kernel for the player weapon range is as follows:

$$P\big(\text{player weapon range} \mid \text{player weapon class}\big).$$

To reason about the player stance rendered in the image snapshot, we consider the conditional probability given the player weapon and player weapon range:

$$P\big(\text{stance} \mid \text{player weapon, player weapon range}\big).$$

Similarly, to reason about the scene rendered in the gameplay video, we consider the conditional probability given the player weapon, player weapon range, and player stance:

$$P\big(\text{scene} \mid \text{player weapon, player weapon range, stance}\big).$$

E.3.3 LEVEL-3 COUNTERFACTUAL PARALLEL WORLD STATEMENTS

We can rewrite the level-2 interventional statements as the following level-3 parallel world statements:

1. Given a character had a light sword and they had a melee weapon; if the character would have had a bow instead, then they would have had a ranged weapon.
2. Given a character had a melee weapon and they were depicted in the snapshot as performing a melee attack at the opponent's location; if the character would have had a ranged weapon instead, then they would have been depicted as performing a ranged attack from their own location.
3. Given a character had a melee weapon and they were shown in the game scene video first approaching the opponent's position before physically swinging the melee weapon; if the character would have had a ranged weapon instead, then they would have been depicted shooting/throwing/casting from their position and the emitted projectile would have been shown traveling towards the opponent from left to right.

E.3.4 COUNTERFACTUAL PROBABILITY EXPRESSIONS

We can formulate these natural language statements in mathematical notation.

Notation. Let:

- $PWC =$ player weapon class,
- $PWR =$ player weapon range,
- $PS =$ player stance,
- $S =$ scene.

Then we have:

Statement 1.

$$P\big(PWR_{PWC=\text{bow}} = \text{ranged} \,\big|\, PWC = \text{light sword},\ PWR = \text{melee}\big) = 1.$$

Statement 2.

$$P\Big(PS_{PWR=\text{ranged}} = \text{depicted performing ranged attack} \,\Big|\, PWR = \text{melee},\ PS = \text{depicted performing melee attack}\Big) = 1, \tag{1}$$

$$P\Big(PS_{PWR=\text{ranged}} = \text{depicted performing melee attack} \,\Big|\, PWR = \text{melee},\ PS = \text{depicted performing melee attack}\Big) = 0. \tag{2}$$

Statement 3.

$$P\Big(S_{PWR=\text{ranged}} = \text{shown performing ranged attack} \,\Big|\, PWR = \text{melee},\ PS = \text{shown approaching the opponent and performing melee attack}\Big) = 1, \tag{3}$$

$$P\Big(S_{PWR=\text{ranged}} = \text{shown performing melee attack} \,\Big|\, PWR = \text{melee},\ PS = \text{shown approaching the opponent and performing melee attack}\Big) = 0. \tag{4}$$

E.4 Spell-Casting Mechanic

Under the spell-casting mechanic, the wizard character may perform one of five spells:

1. **Spawn Magic Projectile Spell to Perform Ranged Attack**
2. **Summon Cloud Platform Spell to Dodge Attack**
3. **Self-Transform Spell to Increase Melee Strength**
4. **Opponent Transform Spell to Lower Enemy Defense**
5. **Levitation Spell to Disarm Opponent**

None of these spell actions can be mitigated by the enemy. Even in the case of the self-transform spell, the sheer size of the transformed wizard (into a golem) renders any block action by the opponent useless. Each spell action choice affects the *stance* and *scene* variables in the rendered game.

E.4.1 level-2 Interventional Definition.

All spells share the same base level-2 contrasting statements:

- If the player character is a wizard, they may wield a magical staff, otherwise they cannot.
- If the character is equipped with a magical staff, they may cast spells, otherwise they cannot.
- If the character casts a spell, they gain a particular offensive or defensive benefit to help them in the battle, otherwise they do not.

E.4.2 Causal Markov Kernels

Notation. Let:

- PC = player character
- PW = player weapon
- PA = player action
- PF = player form
- PS = player stance
- OC = opponent character
- OW = opponent weapon
- OA = opponent action (e.g., defend)
- OF = opponent form
- OS = opponent stance
- B = opponent successfully blocks (semantic variable, only true if the defend action actually succeeds)
- D = opponent dodges
- H = opponent is hurt
- Stance and Scene are the rendered variables

Latent variables:

- OIA = opponent incoming attack indicator

Shared enabling kernels.

$$P(PW = \text{staff} \mid PC = \text{wizard}) > 0$$

$$P(PA \in \{\text{spell actions}\} \mid PW \neq \text{staff}) = 0$$

$$P(PA \mid PW = \text{staff, state}) \text{ is exogenous (agent policy)}$$

Shared state & rendering kernels.

$$P(PF, PS, OF, OS \mid PA, OA, PC, PW, OC, OW)$$

$$P(\text{Stance} \mid PF, PS, OF, OS, PA, OA)$$

$$P(\text{Scene} \mid \text{Stance}, PF, PS, OF, OS, PA, OA)$$

Shared interaction/outcome kernels.

$$P(B \mid PF, OA)$$

$$P(D \mid PS, OIA)$$

$$P(H \mid B, D, PA)$$

E.4.3 level-3 Counterfactual Parallel World Statements.

All spells also share a common set of counterfactual statements:

1. Given a character was not a wizard and therefore could not wield a magical staff; if the character had been a wizard instead, then they could have wielded a magical staff.
2. Given a character was not equipped with a magical staff and therefore could not cast spells; if the character had been equipped with a staff instead, then they could have cast spells.
3. Given a character did not cast a spell and therefore received no benefit in the battle; if the character had cast a spell instead, then they would have gained the corresponding offensive or defensive benefit.

E.4.4 COUNTERFACTUAL PROBABILITY EXPRESSIONS (SHARED).

Not wizard → Wizard ⇒ Staff access

$$P\big(PW_{PC=\text{wizard}} = \text{staff} \,\big|\, PC \neq \text{wizard},\ PW \neq \text{staff}\big) > 0.$$

No staff → Staff ⇒ Spell casting possible

$$P\big(PA_{PW=\text{staff}} \in \{\text{spell actions}\} \,\big|\, PW \neq \text{staff},\ PA \notin \{\text{spell actions}\}\big) > 0.$$

E.5 SPAWN MAGIC PROJECTILE SPELL TO PERFORM RANGED ATTACK

The wizard can cast a spell to conjure and launch a magical projectile directly at the opponent. This action is a ranged attack, and its visual and mechanical dynamics follow the same principles as other ranged weapons described in Section E.3.

E.5.1 LEVEL-2 INTERVENTIONAL DEFINITION.

- If the wizard casts the spawn projectile spell, a magical projectile is launched towards the opponent (i.e., OIA = True), otherwise no projectile is spawned (OIA = False).
- If a magical projectile strikes the opponent and is not successfully blocked or dodged, the opponent is hurt, otherwise they are not.

Involved variables:

- PA = player action
- OA = opponent action (e.g., defend, dodge)
- PS = player stance
- B = opponent successfully blocks
- D = opponent dodges
- H = opponent is hurt

Latent variables:

- OIA = opponent incoming attack indicator

E.5.2 CAUSAL MARKOV KERNELS.

$$P(OIA \mid PA)$$
$$P(B \mid PF, OA)$$
$$P(D \mid PS, OIA)$$
$$P(H \mid B, D, PA)$$

E.5.3 LEVEL-3 COUNTERFACTUAL PARALLEL WORLD STATEMENTS.

1. Given the wizard idled and there was no incoming attack; if the wizard had cast the spawn projectile spell instead, then the opponent would have faced an incoming attack (OIA = True).
2. Given the wizard cast the spawn projectile spell and the projectile hit (i.e., no successful block or dodge) and the opponent was hurt; if the opponent had successfully blocked or dodged instead, then they would not have been hurt.

E.5.4 Counterfactual Probability Expressions.

Idle → Projectile spell ⇒ Incoming attack present

$$P\big(OIA_{PA=\text{projectile spell}} = \text{True} \,\big|\, PA = \text{idle},\, OIA = \text{False}\big) = 1.$$

Projectile hits → Successful block ⇒ Not hurt

$$P\big(H_{B=\text{True}} = \text{False} \,\big|\, PA = \text{projectile spell},\, OIA = \text{True},\, B = \text{False},\, H = \text{True}\big) = 1.$$

Projectile hits → Successful dodge ⇒ Not hurt

$$P\big(H_{D=\text{True}} = \text{False} \,\big|\, PA = \text{projectile spell},\, OIA = \text{True},\, D = \text{False},\, H = \text{True}\big) = 1.$$

E.5.5 Summon Cloud Platform Spell to Dodge Attack

The wizard has a spell they can use to summon a levitating cloud platform and use it to raise themselves up above the battlefield to dodge incoming enemy attacks.

level-2 Interventional Definition.

- If the wizard casts the cloud platform spell, a platform is summoned; the wizard moves onto it and is levitated upwards above the battlefield, otherwise they remain on the ground.
- If the wizard is in an elevated position on a platform during an incoming attack, they successfully dodge and avoid damage, otherwise if they are on the ground they remain vulnerable and may take damage.

Involved variables:

- PA = player action
- OA = opponent action
- OS = opponent stance (grounded, elevated)
- H = opponent is hurt

Latent variables: None.

Causal Markov Kernels.

$$P(OS \mid PA)$$
$$P(D \mid OS, OIA),$$
$$P(H \mid D).$$

level-3 Counterfactual Parallel World Statements.

1. Given the wizard performed the idle action and remained on the ground; if the wizard had cast the cloud platform spell instead, then they would have been elevated.
2. Given the wizard remained on the ground during an incoming attack and was hurt; if the wizard had cast the cloud platform spell instead, then they would have been elevated and dodged the attack, and thus not been hurt.

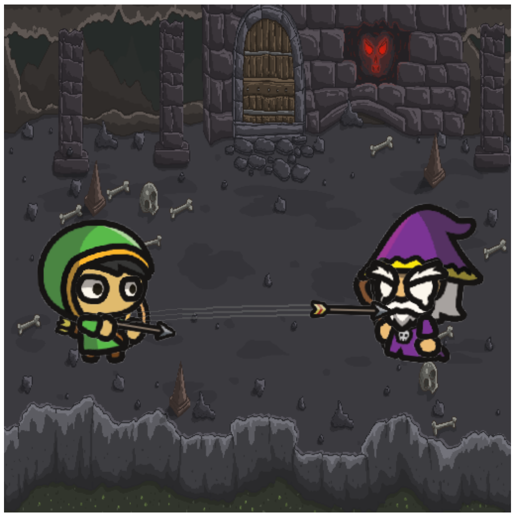
Counterfactual Probability Expressions.

Idle → Cloud Platform ⇒ Elevated stance

$$P\big(OS_{PA=\text{cloud platform}} = \text{elevated} \,\big|\, PA = \text{idle},\, OS = \text{grounded}\big) = 1.$$

Grounded, hurt during incoming attack → Cloud Platform ⇒ Dodge, not hurt

$$P\big(H_{OS=\text{elevated}} = \text{False} \,\big|\, PA = \text{idle},\, OS = \text{grounded},\, OIA = \text{True},\, H = \text{True}\big) = 1.$$


(a) Factual: The wizard idles and is struck by the archer's arrow.

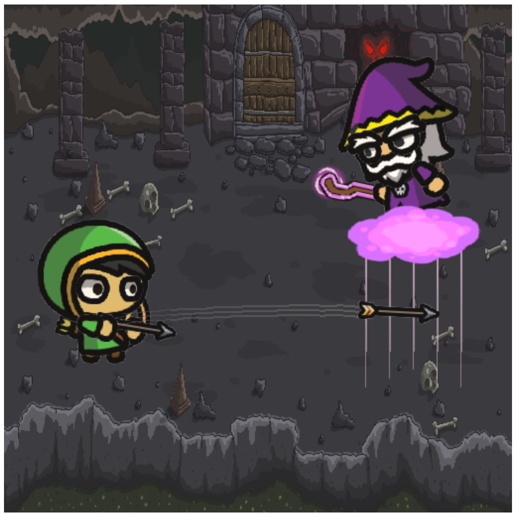

(b) Counterfactual: The wizard summons a magical cloud platform to gain an elevated position and dodge the archer's arrow.

Figure 10: Example counterfactual contrast statement for the summon cloud platform spell mechanic.

E.5.6 Self-Transform Spell to Increase Melee Strength

The wizard can cast a spell to transform themselves into a large golem. In this form, their melee attacks are enormously strengthened and cannot be blocked by opponents.

level-2 Interventional Definition.

- If the wizard casts the self-transform spell, they transform into a golem, otherwise they remain in their normal form.
- If the wizard is transformed into a golem, their melee attack cannot be blocked due to sheer size and strength, otherwise it can.
- If the wizard's golem-form melee attack is not successfully blocked, the opponent will be hurt, otherwise they will not be.

Involved variables:

- PA = player action
- OA = opponent action (e.g., defend)
- B = opponent successfully blocks
- H = opponent is hurt

Latent variables:

- PF = player form (normal, golem)

Causal Markov Kernels.

$$P(PF \mid PA)$$

$$P(B \mid PF, OA)$$

$$P(H \mid B, PA)$$

level-3 Counterfactual Parallel World Statements.

1. Given the wizard performed the idle action and their form remained unchanged; if the wizard had cast the self-transform spell instead, then they would have transformed into a mighty golem.
2. Given the wizard cast a projectile spell and it was blocked by the opponent; if the wizard had cast the self-transform spell instead, then they would have transformed into a golem and the melee attack would not have been blocked by the opponent.
3. Given the wizard cast a projectile spell and it was blocked by the opponent and thus they were not hurt; if the wizard had cast the self-transform spell instead, then they would have transformed into a golem and the melee attack would have bypassed the block and hurt the opponent.

Counterfactual Probability Expressions.

Idle $\rightarrow$ Self-Transform $\Rightarrow$ Golem form

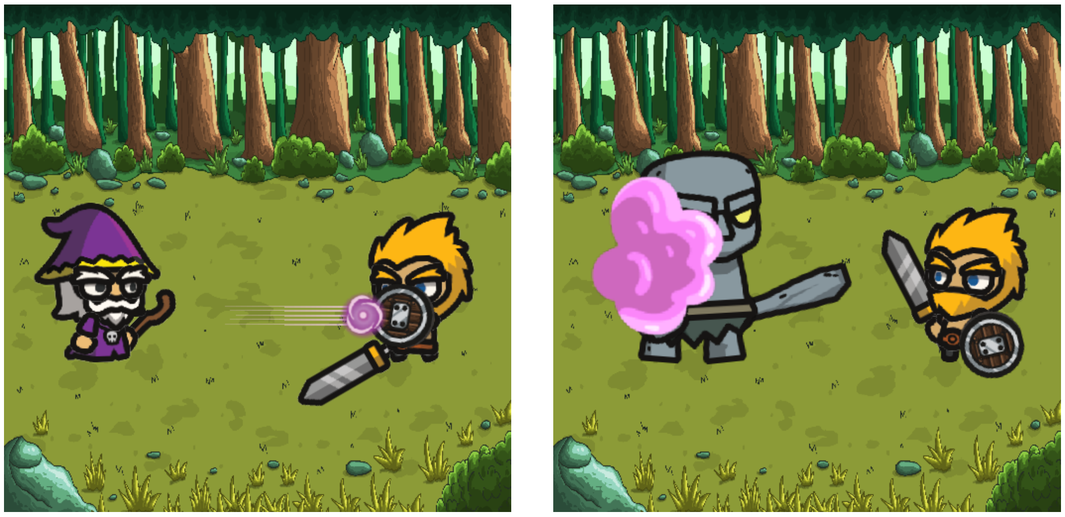
$$P\big(PF_{PA=\text{self-transform}} = \text{golem} \,\big|\, PA = \text{idle},\ PF = \text{normal}\big) = 1.$$

Magic projectile attack blocked $\rightarrow$ Self-Transform $\Rightarrow$ Block attempt failed

$$P\big(B_{PF=\text{golem}} = \text{True} \,\big|\, PA = \text{magic projectile attack},\ PF = \text{normal},\ OA = \text{defend}, B = \text{True}\big) = 0.$$

Magic projectile attack blocked, not hurt $\rightarrow$ Self-Transform $\Rightarrow$ Hurt despite block action

$$P\big(H_{PF=\text{golem}} = \text{True} \,\big|\, PA = \text{magic projectile attack},\ PF = \text{normal},\ OA = \text{defend},\ H = \text{False}\big) = 1.$$



(a) Factual: The wizard casts a magic projectile spell but the warrior blocks it.

(b) Counterfactual: The wizard casts a spell to transform itself into a large golem to perform a powerful melee attack.

Figure 11: Example counterfactual contrast statement for the self-transform spell mechanic.

E.5.7 Opponent Transform Spell to Lower Enemy Defense

The wizard can cast a spell to transform an opponent into a weak or harmless creature (i.e., a snail), disarming them and preventing them from defending against incoming attacks.

level-2 Interventional Definition.

- If the wizard casts no spell, the opponent retains their normal form and can block or defend as usual.
- If the wizard casts the opponent transform spell, the opponent is transformed (e.g. into a snail), otherwise they remain in their normal form.
- If the opponent is transformed, they are disarmed and cannot block; otherwise they may block.

Involved variables:

- PA = player action
- OA = opponent action
- B = opponent successfully blocks
- H = opponent is hurt

Latent variables:

- OF = opponent form (normal, transformed)

Causal Markov Kernels.

$$P(OF \mid PA),$$
$$P(H \mid OF, B).$$

level-3 Counterfactual Parallel World Statements.

1. Given the wizard cast a projectile spell and the opponent remained in their normal form; if the wizard had cast the opponent transform spell instead, then the opponent would have been transformed into a snail.
2. Given the wizard cast a projectile spell and it was blocked by the opponent, and thus the opponent was not hurt; if the wizard had cast the opponent transform spell instead, then the opponent would have been unable to block and would have been hurt by a follow-up attack.

Counterfactual Probability Expressions.

Idle or projectile spell → Opponent Transform ⇒ Opponent form changes

$$P\big(OF_{PA=\text{opponent transform}} = \text{transformed} \,\big|\, PA \neq \text{opponent transform},\ OF = \text{normal}\big) = 1.$$

Projectile spell blocked, not hurt → Opponent Transform ⇒ Block disabled (opponent disarmed)

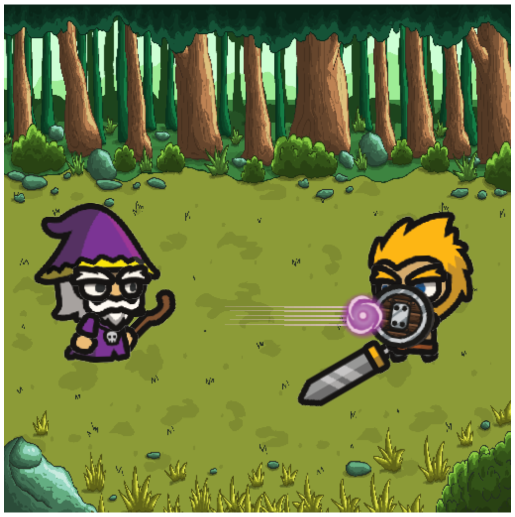
$$P\big(B_{OF=\text{transformed}} = \text{True} \,\big|\, PA = \text{magic projectile attack},\ OF = \text{normal},\ B = \text{True}\big) = 0.$$

E.5.8 Levitation Spell to Disarm Opponent

The wizard can cast a spell that lifts the opponent into the air, leaving them unable to defend or block effectively, and rendering them disarmed.

level-2 Interventional Definition.

- If the wizard casts no spell, the opponent remains grounded and may block normally.
- If the wizard casts the levitation spell, the opponent is lifted into the air, otherwise they remain grounded.
- If the opponent is levitated, they cannot block and are effectively disarmed; otherwise they may block.


(a) Factual: The wizard casts a magic projectile spell but the warrior blocks it.

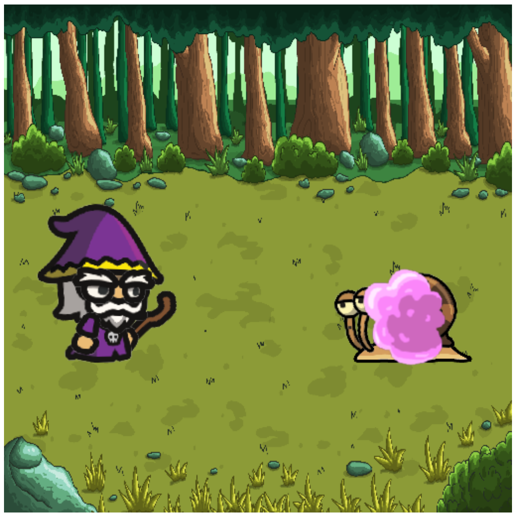

(b) Counterfactual: The wizard casts a spell to transform the warrior into a snail to disarm them.

Figure 12: Example counterfactual contrast statement for the opponent transform spell mechanic.

Involved variables:

- PA = player action
- OA = opponent action
- OS = opponent stance (grounded, levitating)
- B = opponent successfully blocks
- H = opponent is hurt

Latent variables: None.

Causal Markov Kernels.

$$P(OS \mid PA),$$
$$P(H \mid OS, B)$$

level-3 Counterfactual Parallel World Statements.

1. Given the wizard cast a projectile spell and the opponent remained grounded; if the wizard had cast the levitation spell instead, then the opponent would have been levitated.
2. Given the wizard cast a projectile spell and it was blocked by the opponent, and thus the opponent was not hurt; if the wizard had cast the levitation spell instead, then the opponent would have been unable to block and left vulnerable.

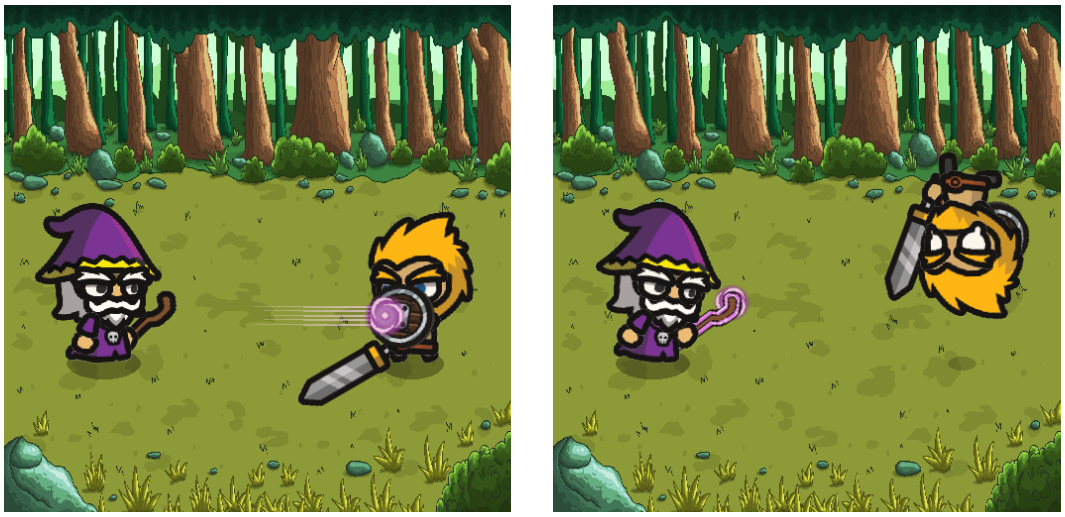
Counterfactual Probability Expressions.

Idle or projectile spell → Levitation ⇒ Opponent stance changes

$$P\big(OS_{PA=\text{levitation}} = \text{levitating} \,\big|\, PA \neq \text{levitation},\, OS = \text{grounded}\big) = 1.$$

Projectile spell blocked, not hurt → Levitation ⇒ Block disabled (opponent disarmed)

$$P\big(B_{OS=\text{levitating}} = \text{True} \,\big|\, PA = \text{magic projectile attack},\, OS = \text{grounded},\, B = \text{True}\big) = 0.$$



(a) Factual: The wizard casts a magic projectile spell but the warrior blocks it.

(b) Counterfactual: The wizard casts a levitation spell to lift the warrior off the ground, disarming them.

Figure 13: Example counterfactual contrast statement for the levitation spell mechanic.

F GAME DESIGN DECISIONS

To ensure that the datasets produced by our simulation respect the causal assumptions of our model, we designed the game architecture with consistency guarantees as a primary objective. This appendix documents how these guarantees were realized in practice, focusing on two complementary aspects: (i) the modular, system-based design of game mechanics that mirrors the structure of the causal model, and (ii) the temporal alignment of captured frames to ensure semantic consistency in dynamic interactions.

F.1 BRIDGING GAME MECHANICS AND CAUSAL MECHANISMS WITH SYSTEM-BASED DESIGN

At the foundation of our data generation pipeline lies a design principle: game mechanics must be implemented in a way that respects and preserves the structure of the causal model they instantiate. To achieve this, we adopted a fully modular *Entity–Component–System* (ECS) architecture (Nystrom, 2014; Gregory, 2018), which enforces locality of causal mechanisms and supports reproducibility across runs.

F.1.1 ENTITIES, COMPONENTS, AND SYSTEMS AS CAUSAL UNITS.

In our implementation, *entities* represent the units of analysis (e.g., characters, projectiles, platforms), *components* represent their attributes and state (e.g., position, animation phase, shield presence), and *systems* encapsulate the transformation rules that govern state evolution (e.g., combat resolution, kinematics, physics, or screenshot scheduling). This separation guarantees that each causal mechanism is expressed locally, without entanglement with unrelated processes.

Each system implements a distinct causal mechanism: for example, the `GameCombatSystem` maps action states of attacker and defender into outcomes such as *hit*, *block*, or *immune*, while the `GamePhysicsSystem` governs the motion of projectiles according to deterministic physical rules. Systems are generally designed to operate frame-by-frame using component data as inputs, but they may also maintain local state when required (for example, the `GameAnimationSystem` manages a per-entity priority queue of animation requests). This design ensures that each causal transformation is encapsulated and modular, while still supporting the persistent state needed for realistic simulation.

F.1.2 Alignment with Causal Graph Structure

The ECS architecture was chosen deliberately to reflect the modular structure of our causal model. Nodes in the causal graph correspond to entity attributes (e.g., *weapon class*, *stance*, *elemental immunity*), while edges correspond to the update dependencies realized through system logic.

For example, the transition of a defender into a *blocking* or *hurt* state depends on multiple upstream components: the presence of a shield component, the character's action selection component (indicating whether the chosen action is to block), and downstream trigger flags such as `needs_to_block` or `is_hurt`. The shield and action selection components establish the potential for blocking, while the trigger flags are set based on situational context (e.g., proximity of an incoming attack). Only when both preconditions and triggers align does the `GameCombatSystem` update the defender's state to *blocking*; otherwise, the state transitions to *hurt*.

This design directly encodes the causal mechanism:

$$\text{shield component} + \text{action selection} \quad \longrightarrow \quad \{\texttt{needs\_to\_block}, \texttt{is\_hurt}\} \quad \longrightarrow \quad \text{outcome (block or hurt).}$$

By structuring dependencies in this way, the system preserves the logic of the causal graph within the mechanics of the game engine. This mapping ensures that system update rules correspond closely to the assignment functions of the causal model.

F.1.3 Locality and Modularity for Consistency

By localizing mechanics in dedicated systems, the architecture prevents hidden confounding across game features. For instance, animation timing is managed exclusively by the `GameAnimationSystem`, while collision and trajectory updates are confined to the `GamePhysicsSystem`. This guarantees that modifying one mechanism (e.g., projectile gravity) does not inadvertently alter another (e.g., collision detection or blocking). Such modularity enforces a form of "causal isolation," allowing dataset generation to reflect the true structure of the designed model.

F.1.4 Reproducibility and Connection to Interventions

The ECS structure also guarantees reproducibility: since each system applies deterministic update rules to the current component state, identical initial conditions yield identical traces. Importantly, the systems themselves do not support interventions in the sense of directly overriding assignment functions during simulation. Instead, interventions are handled at the model level: sampled values from the causal model are passed into the simulation as inputs that determine which branches of the `GameBehaviorTree` are executed (e.g., sampled values specifying whether a character *does block*). The behavior tree then orchestrates the scene by triggering the appropriate system updates, while each system executes its assignment function deterministically given the requested state changes. In this way, the game engine acts as a faithful executor of causal mechanisms, while the intervention logic is confined to the sampling layer above.

Through this design, the simulation environment operates as a direct computational analogue of the causal model, where each mechanism is encapsulated in a corresponding system. This guarantees that generated training data inherits the same modularity and independence properties as the underlying causal graph, thereby supporting consistency-guaranteed counterfactual analysis.

F.2 Impact Frames: Defining Semantic Consistency in Dynamic Causal Model Traces via *Point of Maximum Action* Concept

While the system-based architecture guarantees local causal consistency at the level of game logic, temporal alignment must also be addressed to preserve counterfactual consistency in dynamic interactions. To this end, our system generates gameplay clips of contrasting player turns, each designed to include a canonical *impact frame*: the instant an attack connects, a shield block occurs, or a projectile visibly misses. Ensuring that these per-turn impact frames correspond to semantically equivalent points in the causal process is critical; otherwise, contrasts risk reflecting phase misalignment rather than true causal differences. We therefore formalize alignment using the *Point of Maximum Action*

(PoMA) principle, which anchors impact frames to the most visually and mechanically expressive phase of the interaction.

F.2.1 Conceptual Framing: Temporal Alignment for Counterfactual Comparisons

In static structural causal models (SCMs), counterfactuals are evaluated at a single time index, and semantic alignment across factual and counterfactual worlds is immediate. In dynamic SCMs, the meaning of an event depends on *when* it occurs relative to the unfolding process. For example, a melee strike might connect later than a projectile impact due to differences in action duration. If frames are extracted at fixed indices, we risk capturing non-equivalent phases of these interactions (e.g., an attack wind-up in one turn versus a point of contact in another). If we extract training artifacts (e.g., impact frames) at fixed indices, we risk capturing non-equivalent phases of these interactions (such as a wind-up in one run versus a point of contact in another). This undermines the validity of counterfactual comparisons by introducing differences that are artifacts of temporal phasing rather than consequences of the intervention.

We therefore treat each player turn as a temporal *causal trace* and align counterfactual observations to the *most informative temporal locus* of the relevant event class. We formalize this with the principle of the *Point of Maximum Action* (PoMA).

F.2.2 Point of Maximum Action (PoMA).

Let $A(S_{t'}) \in \mathbb{R}_{\geq 0}$ score how *action-intense* the counterfactual state $S_{t'}$ is with respect to the target event class (e.g., impact, block). The PoMA alignment selects

$$t'_{\mathrm{PoMA}} \;=\; \arg\max_{t' \in T'} A(S_{t'}).$$

PoMA frames are then extracted at t'_{PoMA}, aligning the impact frame to the point of maximum expressivity. This guarantees that contrasts correspond to the same semantic phase of the interaction, regardless of variation in action duration.

F.2.3 Four Alignment Methods: Brief Summary with Pros and Cons

We summarize four practical approaches for temporal alignment of dynamic counterfactuals. Each provides a distinct trade-off between simplicity, robustness, and semantic fidelity.

1) Constant Time Interval. *Rule.* Evaluate the counterfactual variable at the same nominal time as the factual: $t' \equiv t$. *Pros.* Conceptually simple; trivial to implement; deterministic. *Cons.* Fails when action durations differ; risks capturing non-equivalent phases (e.g., mid-swing vs. impact); unsuitable for dynamic interactions where event timing adapts to interventions.

2) Equilibrium-Based. *Rule.* Evaluate once the counterfactual dynamics have reached a steady state or absorbing condition (e.g., $\|S'_{t'+1} - S'_{t'}\| < \varepsilon$ or a domain predicate holds). *Pros.* Appropriate for tasks where long-run properties matter; robust to transient phasing differences; alignment invariant to small shifts in sequence length. *Cons.* Inapplicable to inherently transient events (e.g., impacts); some episodes may not converge; equilibrium may erase the very distinctions needed to analyze acute causal effects.

3) Point of Maximum Action (PoMA). *Rule.* Align to the counterfactual time of peak action intensity for the event class:

$$t'_{\mathrm{PoMA}} \;=\; \arg\max_{t'} A(S'_{t'}).$$

Pros. Directly targets the most salient phase of the interaction; robust to differences in sequence length; naturally accommodates variable-duration actions by abstracting to their peak. *Cons.* Requires a well-defined intensity score A; can be non-trivial for abstract or multi-agent interactions; may require smoothing when peaks are brief or noisy.

4) Semantic Consistency. *Rule.* Align by maximizing semantic similarity between factual and counterfactual states. *Pros.* General and flexible in principle; useful in settings where semantic descriptors are available. *Cons.* Not used in our implementation. It requires an additional similarity metric and embedding design, which introduces complexity and potential bias.

In practice, we adopt the PoMA approach, extending it with event-specific scoring functions and event-defined windows to handle variable-duration interactions. Constant Time and Equilibrium serve primarily as conceptual baselines, while semantic similarity was considered but not implemented.

F.2.4 Implementation in Our Game: Event-Based Windows and Weighted Scoring

To instantiate these principles in a reproducible data pipeline, our engine implements an event-driven *GameScreenshotSystem* that schedules captures at semantically aligned moments:

1. **Event Detection.** The simulation raises structured events for interactions of interest (e.g., `melee_impact`, `projectile_hit`, `shield_block`). Each event is associated with the participating entities and their current states.
2. **Event-Based Scoring Windows.** For each interaction type, we define a start event and a stop event that bound a scoring window (e.g., `swing_start` → `impact`, or `projectile_cast` → `collision`). These windows are managed directly in the behavior tree, ensuring that scoring only occurs during the semantically relevant phase of the interaction.
3. **Weighted Scoring.** Within each window, frames are scored according to event-specific weights. For example, projectile impact events may be given higher weight than projectile flight, and shield contact may be prioritized over shield raise. The capture frame is then chosen as
$$\widehat{t'} \in \operatorname*{arg\,max}_{t' \in \text{window}} A(S'_{t'}),$$
with deterministic tie-breaking for reproducibility.

By anchoring artifact capture to event-defined scoring windows and applying weighted intensity scoring, our pipeline produces semantically aligned visual data across simulations, despite natural variability in action durations. This guarantees that counterfactual comparisons reflect genuine causal differences, rather than artifacts of capturing frames at arbitrary, non-equivalent time indices.

F.2.5 Summary.

Taken together, these design choices preserve consistency at both structural and temporal levels. The ECS architecture ensures local mechanics map cleanly to modular causal mechanisms, while the screenshot system anchors each player turn contrast to a canonical *impact frame* selected via PoMA. By aligning contrasts at the most expressive phase of interaction, the system guarantees that observed differences reflect genuine causal effects rather than timing artifacts, producing impact frames that are both causally and semantically consistent.

G Specifications of Generated Video Clips

Generated video clips have following technical specifications:

G.1 Resolution and Format

All video clips are rendered at a resolution of 512×512 pixels in a square aspect ratio. Videos are encoded in MP4 format using the H.264 codec with yuv420p pixel format to ensure broad compatibility across video players and analysis frameworks.

G.2 Frame Rate, Duration, and Timing

Videos are captured at 50 frames per second (FPS) using a fixed-interval delta time method to ensure consistent temporal sampling across all generated clips. This approach decouples game simulation time from wall-clock time, enabling reproducible frame timing essential for dataset generation and comparative analysis.

Video clip duration is variable and event-driven, spanning the complete battle sequence from initialization to completion. Clip length is determined by the termination of both player behavior trees, typically ranging from several seconds to longer sequences depending on the complexity of actions performed (e.g., melee attacks, spell casting, projectile interactions, defensive maneuvers).

G.3 Dataset Organization

The system generates consistent contrasts, enabling direct comparison between observed battle outcomes and alternative scenarios under modified conditions. Each video file follows the naming convention `battle_XXXXXX-TIMESTAMP.mp4`, where `XXXXXX` represents a zero-padded 6-digit battle identifier. Accompanying JSON metadata files provide technical details including encoding parameters, frame timing information, and battle configuration data.

G.4 Implementation

Video generation utilizes the imageio library with PyAV backend for efficient encoding. The rendering pipeline captures frame sequences from the game's screenshot buffer system, which also maintains temporal consistency in impact frames through the behavior tree execution framework.

G.5 Variables generated

In addition to clips, each generated example includes:

- Controller inputs C_t
- Full or partial state X_t
- Mechanic-specific CL-DAG $G^{\mathcal{M}}$
- Set of mechanic-specific parallel world DAGs
- Set of mechanic-specific counterfactual DAGs
- Seed / ω identifiers

H Suggested Metrics

Task	**Description**	**Metric**
Fully observed mechanic inference	Infer $\mathbf{P_i}$ using clips, game state variables, and controller inputs	$D_{\mathrm{KL}}\big(P_i \,\|\, \hat{P}_i\big)$
Partial (canonical) mechanic inference	Infer $\mathbf{P_i}$ using clips and controller inputs, assuming game state is unobserved	$D_{\mathrm{KL}}\big(P_i \,\|\, \hat{P}_i\big)$
Generate consistent contrasts	Generate multiple consistent contrast examples for each d_i associated with the mechanic, and validate that they are visually accurate and consistent	Human or VLM validation
Counterfactual abduction	Generate $\{v_{X=x}, v_{X=x'}\}$ from game, obtain $\hat{v}_{X=x'} = \mathbb{E}[V_{X=x'} \mid V_{X=x} = v_{X=x}]$ using the model, then compare $\hat{v}_{X=x'}$ to $v_{X=x'}$	Human or VLM validation

Table 5: Tasks that can be used to evaluate mechanic learning with Multiverse Mechanica.

Table 5 summarizes the four tasks for evaluating mechanic-learning with *Multiverse Mechanica*. We expand here on how each metric is operationalized and its relation to prior work.

H.1 Mechanic Inference.

The fully- and partially-observed inference tasks require estimating the distributions $\hat{P}_i$ implied by mechanic-specific constraints $\mathcal{M}$ and comparing them against the ground truth distributions P_i. We adopt KL-divergence as the primary quantitative measure, explicitly reporting $D_{\text{KL}}(P_i \,\|\, \hat{P}_i)$ for each mechanic. This aligns with classical practice in distributional evaluation, where divergence to the ground truth provides a direct measure of whether the model has captured the stochastic relations induced by the mechanic.

H.2 Consistency in Contrast Generation.

The contrast-generation and counterfactual-abduction tasks require evaluating whether pairs of clips $(v_{X=x}, v_{X=x'})$ are *consistent*, i.e. differing only in outcomes attributable to the intervention variable X while holding non-descendant factors fixed. Prior work has typically relied on human evaluation: for example, Gingerson et al. (2024) collected large-scale human judgments of whether generated gameplay videos adhered to intended mechanics. Recent work has investigated whether VLMs can serve as automated evaluators of model generations. Hendriksen et al. (2025) show that pre-trained VLMs can be adapted for evaluating world model rollouts, e.g. scoring whether predicted videos match textual descriptions of target outcomes. However, to our knowledge, no prior work has specifically used VLMs to evaluate *consistency* across parallel-world contrasts, as defined by causal counterfactual principles. This remains an open direction, and *Multiverse Mechanica* provides level-3 parallel-world data where such evaluations can be systematically explored.

Work by Monteiro et al. (2023) introduced *composition* metrics that effectively evaluate consistency in image-based counterfactuals by checking whether attributes remain unchanged under controlled edits. These metrics capture aspects of counterfactual faithfulness that parallel our notion of consistency. However, they have not yet been extended to video data, nor applied to generative modeling of game mechanics. We view *Multiverse Mechanica* as a platform to develop such extensions, allowing both human and VLM-based evaluation methods to be compared side-by-side.

H.3 Summary.

In short, our metrics combine classical distributional divergences for inference tasks with human- or VLM-based consistency checks for contrastive generation tasks. This dual approach ensures that models are evaluated not only on reproducing distributions of state variables but also on capturing the causal consistency of gameplay dynamics under controlled interventions.

I Theoretical and Implementation Details for Proof-of-concept

I.1 Background on Diffusion Models and Reverse-Sampling

Denoising diffusion probabilistic models (DDPMs) define a forward Markov chain that gradually adds Gaussian noise to an image and a learned reverse process that denoises it step by step. Given a data sample $x_0 \sim q(x_0)$, the forward process produces x_t by directly adding noise to the data: $q(x_t|x_{t-1}) = \mathcal{N}\big(x_t; \sqrt{\alpha_t}x_{t-1}, (1-\alpha_t)I\big)$ for $t = 1, \ldots, T$. Here $0 < \alpha_t < 1$ are predefined variances. In the reverse generation, one starts from pure noise $x_T \sim \mathcal{N}(0, I)$ and iteratively denoises to x_0 using a model ϵ_θ trained to predict the injected noise. At each step, the model predicts $\hat{\epsilon} \approx \epsilon$ such that x_{t-1} can be estimated by removing noise: e.g., $x_{t-1} = \frac{1}{\sqrt{\alpha_t}}\big(x_t - (1-\alpha_t)\hat{\epsilon}\big)$ (with additional variance for stochastic sampling). In practice, one can also use the continuous-time formulation and solve a reverse stochastic differential equation or its deterministic counterpart (as in DDIM), yielding a mapping from an initial noise u directly to an image $x = G(u, c)$. Importantly, this exogenous noise u acts as the stochastic latent that accounts for random variation in generated images. Using a deterministic sampler (e.g.,setting $\eta = 0$ in DDIM), one obtains a one-to-one mapping between u and the output x, and can invert a given image to its corresponding u for a particular conditioning c.

We will leverage this invertibility to extract the latent noise u_a from the original image x and the latent noise u_b from the counterfactual image x_{cf}, where x denotes the factual image generated under the original conditioning, x_{cf} represents the counterfactual image generated under modified conditioning, u_a is the inverted latent noise corresponding to the factual image x, and u_b is the inverted latent noise corresponding to the counterfactual image x_{cf}.

I.2 Background on Causal Counterfactuals in Image Generation

In causal terms, we can view the generative model as a structural causal model $x := f_\theta(u, c)$, where c is a cause (e.g.,textual description or a set of discrete random variables) and u is an unobserved exogenous variable accounting for randomness. A counterfactual image aims to answer: "What would the image look like if we change c from c_A to c_B, while keeping all other latent factors the same?" The classical procedure for generating such counterfactuals is the three-step abduction-action-prediction: (1) Abduction: infer the exogenous noise u_a that produced the factual image x under c_A (this u_a captures the instance-specific variations of x); (2) Action: intervene by setting the prompt to c_B (while keeping u_a fixed); (3) Prediction: generate the new image as $x_{\text{cf}} = f_\theta(u_a, c_B)$. This procedure, if the model perfectly captures the true causal mechanism, would change only the aspects of the image directly affected by c and leave other details intact (satisfying causal consistency that no non-descendant features of c change). In practice, directly using u_a with a new prompt c_B can produce a reasonable edited image, but it may fail or produce artifacts if c_B demands alterations that conflict with the original latent factors.

By contrast, many non-SCM image editing approaches do not explicitly enforce the same latent noise. For example, one might simply prompt the model with c_B and generate a new sample (different u), or apply heuristics like latent interpolation, attention refocusing, or mask-based noising of only certain regions. These approaches can produce plausible results, but often lack guarantees that only the intended changes occur—the model might inadvertently change unrelated details because the random draw u or other generation conditions differ. Our goal is to incorporate causal principles into the diffusion editing process to maximize counterfactual faithfulness (only c-dependent changes) while still allowing the model flexibility to implement the edit realistically.

I.3 Contrastive Training via Alignment Losses

In this section, we provide the necessary background to help understand our method.

I.4 Notation

We first define the notation used in our counterfactual editing framework. Let x denote the original (factual) image, and let x_{cf} denote the counterfactual image we aim to generate (the edited image after an intervention). We model image generation via a diffusion model as $x = G(u, c)$, where u is an initial exogenous noise (drawn from a Gaussian prior, typically $u \sim \mathcal{N}(0, I)$) and c is the conditioning (in our case, a text prompt). We use c_A for the original prompt and c_B for the counterfactual prompt. Using an inversion technique (e.g., reverse ODE or deterministic DDIM inversion), we can obtain u_a as the noise that generates x under c_A, and similarly u_b as the noise corresponding to x_{cf} under c_B. We denote by x_t the (noisy) latent image at diffusion timestep t when evolving toward x (with $x_0 = x$ and $x_T = u_a$ for the forward noising process), and likewise $x_{\text{cf},t}$ for the counterfactual trajectory. The diffusion model's denoiser is denoted $\epsilon_\theta(x_t, c, t)$, which predicts the added noise at step t for latent x_t and conditioning c. For brevity, we write $\epsilon(x_t, c, t)$ when θ is clear from context. Finally, $\mathcal{L}_1$, $\mathcal{L}_2$, $\mathcal{L}_{\text{text}}$, and $\mathcal{L}_{\text{sub}}$ will denote different loss terms introduced below.

I.5 Method 1: L_1 – Consistency Alignment

Our first new loss function enforces consistency alignment between the factual and counterfactual generations. We obtain the noise u_a and u_b corresponding to x and x_{cf} respectively (via the inversion process described above). The consistency alignment loss is then defined as,

$$\mathcal{L}_1 = \|u_a - u_b\|_2^2, \quad u_* = H_\theta^{T \leftarrow 0}(x_*, c_*) \tag{5}$$

where $H_\theta^{T \leftarrow 0}$ represents the inversion function that maps from image space back to noise space at timestep T.

Given the diffusion model $x = f_\theta(u, c)$, we have:

$$x = f_\theta(u_a, c_A) \tag{6}$$

$$x_{\text{cf}} = f_\theta(u_b, c_B) \tag{7}$$

The inversion takes the image back to the noise, which yields:

$$u_a = H_\theta^{T \leftarrow 0}(x, c_A) \tag{8}$$

$$u_b = H_\theta^{T \leftarrow 0}(x_{\text{cf}}, c_B) \tag{9}$$

The $\mathcal{L}_1$ is added to the training loss as a regularization term to enforce ***exogenous invariance***. In the ideal case where $u_b = u_a$, the counterfactual generation becomes:

$$x_{\text{cf}} = f_\theta(u_a, c_B) \tag{10}$$

which ensures that all variations between x and x_{cf} are attributed solely to the conditioning change $c_A \to c_B$, while preserving the exogenous factors encoded in u_a.

This loss directly penalizes any differences between the underlying noise vectors of the original and edited image. The motivation is to ensure that x and x_{cf} share the same source of variation, so that as much of the scene's random details as possible remain unchanged. This approach extracts editing-related information from the seed, enabling differences to be more expressed by the conditioning c rather than random variations. The primary change affects the reverse mapping/generator's decomposition of conditions, belonging to "seed-level" invariance.

Intuitively, if u_a and u_b are identical, the only differences between x_{cf} and x will come from the changed conditioning c_B vs c_A. In the ideal case, $\mathcal{L}_1 = 0$ means we are generating the counterfactual with the exact same "random seed" as the factual image (the pure SCM counterfactual). This encourages maximal consistency: the background, lighting, style, and other incidental attributes should stay the same unless the new prompt explicitly demands their change. Enforcing $\mathcal{L}_1$ provides strong alignment that leads to stable edits. It prevents the edited image from drifting in appearance or composition: the counterfactual will tend to have the same objects and layout as the original, only differing in the aspects dictated by the prompt change. This is beneficial for preserving identity (e.g.,the same person's face before and after an edit) and ensuring only the intended attributes change.

However, this strict constraint can also have a few limitations. If the counterfactual prompt c_B is significantly different from c_A, using exactly the same noise u_a might overly constrain the generation, resulting in artifacts or an incomplete edit. The model might struggle to reconcile u_a (which was optimal for the original content) with the new prompt, leading to implausible images or failure to fully achieve the desired change.

I.6 Method 2: L_2 – Structure Preservation at High-Noise

As a more relaxed alternative, we propose to align the diffusion model's behavior for the two images at the high-noise stages of generation, rather than forcing the initial noises to be identical. Concretely, let S be a set of diffusion time steps focusing on the high-noise region (e.g.,the latter half or last third of the diffusion schedule, when x_t is still highly noisy). We define the structure preservation loss $\mathcal{L}_2$ as:

$$\mathcal{L}_2 = \sum_{t \in S} \|\epsilon_\theta(x_t, c_A, t) - \epsilon_\theta(x_{\text{cf},t}, c_B, t)\|_2. \tag{11}$$

$\mathcal{L}_2$ measures the disparity between the model's denoising predictions for the factual versus counterfactual image trajectories, but only at very noisy states (where x_t is mostly noise). By penalizing this difference, we encourage the denoiser's reaction to the two inputs to be the same in the early stages of generation. This effectively steers $x_{\text{cf},t}$ to evolve in a similar direction as x_t while the image is still coarse and noisy, ensuring the two generation processes start out aligned in terms of global structure. Importantly, L_2 does not enforce that the latent noises x_t themselves are exactly equal, only that the predicted noise residuals (or equivalently, the score vectors) are similar. This distinction makes L_2 a partial relaxation of the L_1 constraint. It nudges the counterfactual to have a similar high-level appearance without locking in all the exact stochastic details.

When using L_2, the model is free to adjust u_b as needed, but it will still preserve large-scale aspects of u_a. For example, if x depicts a particular scene layout, L_2 will bias x_{cf} to keep that layout, since early denoising steps (which shape the overall composition) will be similar for both. As t gets smaller (less noise), x_{cf} can gradually diverge more to realize the new content c_B specifies.

This approach maintains structure and identity better than an unconstrained edit, while granting more flexibility than L_1 for the model to incorporate the new prompt. In essence, L_2 focuses on aligning the coarse-grained features (which are determined in high-noise stages) and lets the fine details emerge freely.

One might consider applying a spatial mask so that structure preservation is enforced only on certain regions (for instance, only aligning the background areas that should remain unchanged). In our approach, we generally did not require an explicit mask for L_2. Since L_2 operates on high-noise (low-detail) states, it inherently affects global structure more than specific fine features. We found that a well-balanced L_2 encourages overall consistency without needing per-pixel restrictions—the model naturally preserves unedited parts of the image. However, if a particular application demands strict locality (e.g.,editing only a small region while leaving everything else exactly as is), a mask could be introduced to further ensure no influence of L_2 on the region to be changed (or conversely, to focus L_2 only on the region to preserve). In summary, L_2 already provides a soft, global consistency constraint, and masking is an optional refinement rather than a necessity in most cases.

I.7 Abduction–Action–Prediction and Its Diffusion Emulation

I.7.1 SCM Setup.

Let $\mathcal{M} = (\mathbf{U}, \mathbf{V}, \mathbf{F}, P(\mathbf{U}))$ be a structural causal model with exogenous variables $\mathbf{U}$, endogenous variables $\mathbf{V}$, structural assignments $\mathbf{F}$, and exogenous distribution $P(\mathbf{U})$ (Pearl, 2009). We single out: (i) the *mechanic variables* $X \subseteq \mathbf{V}$ that we will intervene on; (ii) the *controller input* C; and (iii) the visual variable V whose realization is a impact frame snapshot v. Throughout, we explicitly treat our generation seed ω as a realization of the SCM exogenous variables, i.e., $\omega \sim P(\mathbf{U})$, and we treat all other shared rendering conditions as part of ω (while C is held fixed explicitly).

I.7.2 Abduction–Action–Prediction (AAP).

Given a factual observation v_0 obtained under $(X{=}x_0, C{=}c)$, the AAP recipe for the two-world case proceeds as:

(Abduction)	Infer $P(\mathbf{U} \mid V{=}v_0, X{=}x_0, C{=}c)$, choose a representative $\hat{\omega}$.
(Action)	Form the intervened model $\mathcal{M}_{X=x_1}$ while keeping $C{=}c$ and $\mathbf{U}{=}\hat{\omega}$ fixed.
(Prediction)	Evaluate the counterfactual $V_{X=x_1}(\hat{\omega})$.

Equivalently, in distributional form,

$$\hat{v}_1 \;\sim\; P\big(V_1 \,\big|\, V_0{=}v_0,\, X_0{=}x_0,\, C{=}c\big),$$

which is shorthand for propagating $\hat{\omega} \sim P(\mathbf{U} \mid V_0{=}v_0, X_0{=}x_0, C{=}c)$ through $\mathcal{M}_{X=x_1}$.

Algorithm 1 Abduction–Action–Prediction (two-world case)

```
Require: SCM $\mathcal{M} = (\mathbf{U}, \mathbf{V}, \mathbf{F}, P(\mathbf{U}))$, factual $(v_0, x_0, c)$, target $x_1$
1: Abduction: Infer $\hat{\omega} \leftarrow$ MAP/mean/sample from $P(\mathbf{U} \mid V{=}v_0, X{=}x_0, C{=}c)$
2: Action: Construct $\mathcal{M}_{X=x_1}$; hold $C{=}c$ and $\mathbf{U}{:=}\hat{\omega}$
3: Prediction: Compute $\hat{v}_1 \leftarrow V_{X=x_1}(\hat{\omega})$
4: return $\hat{v}_1$
```

I.8 Diffusion-Based Emulation of AAP

In our latent diffusion setting, we emulate abduction–action–prediction (AAP) by identifying the SCM exogenous variables with the model's initial latent noise:

$$\mathbf{U} \;\longleftrightarrow\; \omega \sim \mathcal{N}(0, I).$$

Abduction (DDIM inversion). We estimate $\hat{\omega}$ from a factual impact frame $V_{X=x_0}$ under (x_0, c) via deterministic sampler inversion (DDIM, $\eta{=}0$). Let $Z_{t,X=x_0}$ denote its noisy latents across

$t \in [0, T]$. At each step, the denoiser $\epsilon_\theta(Z_{t,X=x_0}, t, c_0)$ predicts the noise, and inversion equations (Appendix I.10.5) are used to propagate forward in the schedule, yielding an estimate $\hat{\omega} = \mathsf{Abduct}_\theta(Z_{0,X=x_0}, c_0)$.

Action. Hold c fixed and change the mechanic state from x_0 to x_1.

Prediction (deterministic reverse). Initialize the trajectory with $Z_{T,X=x_1} := \hat{\omega}$ and run the deterministic reverse process conditioned on (x_1, c) to obtain a counterfactual latent $Z_{0,X=x_1}$, then decode it to the counterfactual impact frame $\hat{V}_{X=x_1}$.

Algorithm 2 Diffusion-based AAP via DDIM (η=0)

```
Require: Diffusion model ($\epsilon_\theta$, scheduler, decoder), factual ($V_{X=x_0}, x_0, c$), target $x_1$
 1: Abduction: $\hat{\omega} \leftarrow \mathsf{Abduct}_\theta(Z_{0,X=x_0}, c_0)$        // DDIM inversion in latent space
 2: Action: Keep $c$ fixed, set $X := x_1$
 3: Prediction: $Z_{T,X=x_1} := \hat{\omega}$; for $t = T, \dots, 1$: $Z_{t-1,X=x_1} \leftarrow \mathrm{DDIMStep}\big(Z_{t,X=x_1}, t, x_1, c; \epsilon_\theta\big)$;
    $\hat{V}_{X=x_1} \leftarrow \mathrm{decoder}(Z_{0,X=x_1})$
 4: return $\hat{V}_{X=x_1}$
```

I.8.1 Connection to Our Training Losses.

The AAP framing clarifies the roles of our loss components. (i) *Exogenous alignment* $\mathcal{L}_1$ encourages shared-seed invariance by driving

$$\mathsf{Abduct}_\theta(Z_{0,X=x_0}, c_0) \;\approx\; \mathsf{Abduct}_\theta(Z_{0,X=x_1}, c_1),$$

for elements of the same contrast that share the true ω. This ensures the abduction step produces consistent seeds across worlds. (ii) *Structure preservation* $\mathcal{L}_2$ aligns denoiser outputs $\epsilon_\theta(Z_{t,X=x_j}, t, c_j)$ at high-noise steps $t \in S$, enforcing agreement on coarse global structure during the early reverse process. This mirrors AAP's assumption that exogenous factors (ω) are held fixed while only X changes.

I.8.2 SCM $\leftrightarrow$ Diffusion mapping (two-world case).

SCM concept		**Diffusion instantiation**
Exogenous $\mathbf{U}$	$\leftrightarrow$	Initial latent noise seed $\omega \sim \mathcal{N}(0, I)$
Abduction $P(\mathbf{U} \mid V_{X=x_0}, x_0, c)$	$\leftrightarrow$	DDIM (η=0) latent inversion $\hat{\omega} = \mathsf{Abduct}_\theta(Z_{0,X=x_0}, c_0)$
Action $X = x_1$	$\leftrightarrow$	Condition reverse process on (x_1, c)
Prediction $V_{X=x_1}(\hat{\omega})$	$\leftrightarrow$	Deterministic reverse to $Z_{0,X=x_1}$ then decode $\hat{V}_{X=x_1}$
Causal consistency	$\leftrightarrow$	Shared seed $\hat{\omega}$; early-step structure preservation ($\mathcal{L}_2$)

Abduction via DDIM inversion yields an *estimate* $\hat{\omega}$ whose fidelity depends on the schedule and conditioning; see Appendix I.10.6 for caveats and tuning guidance.

I.9 Additional Regularizers

Beyond the core losses L_1 and L_2, our framework can incorporate additional terms to improve consistency and fidelity:

I.9.1 Subspace Consistency Loss.

We can encourage the factual and counterfactual images to remain close in certain intermediate representations of the diffusion model. For example, one may align hidden latents or cross-attention maps at corresponding diffusion steps. By penalizing differences in these subspaces (e.g.,the model's multi-head attention maps for background tokens, or feature maps in a particular UNet layer), we enforce that the internal generation pathways for x and x_{cf} stay similar. This helps preserve layout and identity at a semantic level, complementing the pixel-space alignment enforced by

L_1/L_2. Formally, if $F_t(x)$ denotes some feature (such as a latent embedding or attention tensor) computed during the denoising of x at step t, we can define a loss $L_{\text{sub}} = \sum_{t \in \mathcal{T}} \|F_t(x) - F_t(x_{\text{cf}})\|_2$ for some chosen set of layers or timesteps $\mathcal{T}$. This subspace consistency loss encourages the edited image to differ only minimally in features unrelated to the intervention.

Text Consistency Loss. To ensure the edited image indeed reflects the counterfactual prompt c_B, we include a text-image consistency term. We rely on a pretrained image-text similarity model (such as CLIP) to measure alignment between x_{cf} and the description c_B. Let $\text{sim}(x_{\text{cf}}, c_B)$ be a similarity score (higher means the image matches the prompt better). We define a loss $L_{\text{text}} = -\text{sim}(x_{\text{cf}}, c_B)$ (or equivalently $1 - \text{sim}$, depending on the normalization) so that minimizing L_{text} maximizes the agreement between the generated image and the desired attributes. This ensures that while preserving content, we do not under-shoot the edit: the new image should clearly exhibit the prompted change. The text consistency loss guides the generation to remain faithful to the user's request, especially when L_1 or L_2 are pulling towards the original image. It helps avoid the outcome where the edit is so conservative that the difference between x_{cf} and x is hard to discern.

I.10 Loss Combination

Our full counterfactual editing objective combines these components in a weighted sum:

$$L_{\text{total}} = \lambda_1 L_1 + \lambda_2 L_2 + \lambda_3 L_{\text{text}} + \lambda_4 L_{\text{sub}}, \tag{12}$$

where λ_i are tunable weights that control the influence of each loss term. In practice, we choose these weights to balance identity preservation against effective editing. Typical values and trade-offs are as follows:

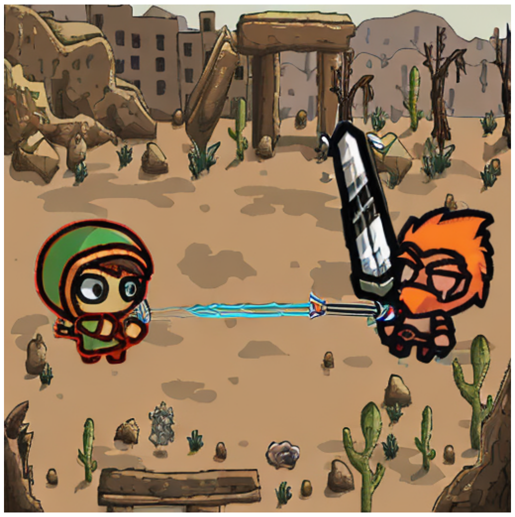
λ_1 **(Consistency Alignment):** This is often kept relatively small (e.g., λ_1 in the range 0 to 0.5) unless the edit is very minor. A small λ_1 nudges the initial noise vectors closer without forcing identity completely. Increasing λ_1 leads to more literal counterfactuals (very high consistency with the original image's details), but if set too high it may prevent the new attributes from appearing strongly. There is a trade-off between maintaining background/identity (higher λ_1 favors this) and allowing change (lower λ_1 gives more freedom).

λ_2 **(Structure Preservation):** We usually give L_2 a moderate to high weight (on the order of 1.0) as it is the principal mechanism to preserve structure. λ_2 in a range roughly 0.5 to 2.0 works well. A larger λ_2 tightly constrains the high-level layout and style to match the original, which is good for identity preservation; however, if λ_2 is excessively large, it can act almost as strictly as L_1, potentially impeding necessary changes. Reducing λ_2 allows the counterfactual generation to deviate more in composition if needed, but too low λ_2 might result in unwanted alterations in background or other objects.

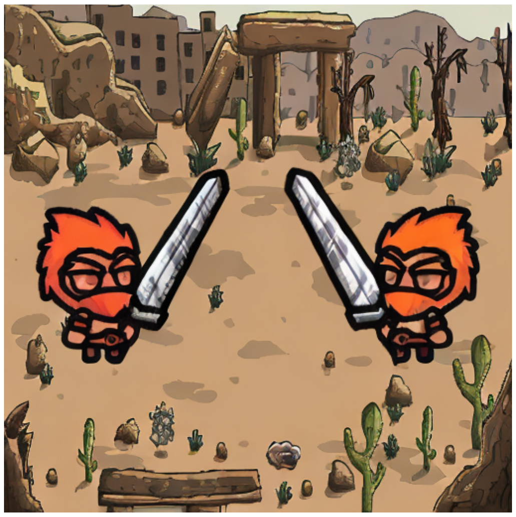
λ_3 **(Text Consistency):** This weight should be high enough to ensure the edit actually happens (especially for subtle changes), but not so high that it overrides the preservation losses. In practice λ_3 is often set around 0.5 to 1.0 (assuming similarity is scaled to a comparable range) so that the image aligns with the prompt without artifacts. If λ_3 is set too low, the edit might be too conservative (the model might simply regenerate the original image to satisfy L_1/L_2). If λ_3 is too high, the model may introduce exaggerated or incorrect features to satisfy the prompt, possibly compromising identity or visual quality.

λ_4 **(Subspace Consistency):** If used, this is typically a small auxiliary weight (e.g.,0.1). Since L_{sub} operates on internal features, it can strongly bind the generation if overweighted. A modest λ_4 helps reinforce structural consistency without conflicting with the primary losses. Tuning λ_4 involves checking that it indeed improves preservation of details like face identity or scene layout, without, for example, freezing the image in an early-state that ignores the new prompt. In some cases, we might set $\lambda_4 = 0$ (i.e., not use this term) if we find L_1 and L_2 are sufficient; when used, it serves as an extra regularizer.

In summary, L_1 and L_2 provide a spectrum between strict and loose alignment, λ_3 drives the fidelity to the requested counterfactual change, and λ_4 can bolster consistency on a feature level. We recommend starting with a balanced combination (for instance, $\lambda_1 = 0.2, \lambda_2 = 1.0, \lambda_3 = 0.5, \lambda_4 = 0.1$).



(a) Factual: A 1-on-1 battle. Foe 1 type is archer, element is fire, weapon is bow, weapon element is none, has shield is false, action is shoot. Foe 2 type is warrior, element is fire, weapon is long sword, weapon element is none, has shield is false, action is idle. Foe turn is foe 1.



(b) Counterfactual: A 1-on-1 battle. Foe 1 type is warrior, element is fire, weapon is long sword, weapon element is none, has shield is false, action is idle. Foe 2 type is warrior, element is fire, weapon is long sword, weapon element is none, has shield is false, action is idle. Foe turn is foe 1.

Figure 14: Example counterfactual image pairs generated from finetuned text-to-image diffusion model.

I.10.1 Diffusion Backbone.

We use a pretrained Stable Diffusion variant with a deterministic sampler [1], which makes the mapping between exogenous noise u and image v approximately invertible. This enables abduction of u from an image and consistent reuse across parallel worlds. Classifier-free guidance is applied with scale 7.5, and the scheduler uses $T = 50$ steps.

I.10.2 Conditioning.

In the main text we describe conditioning directly on game state variables (e.g., shield, weapon type, block outcome). For implementation, these variables and outcomes are converted into natural-language captions for compatibility with CLIP text encoders. This is a nuisance parameterization: the underlying conditioning remains the game state variables.

I.10.3 Training Setup.

We fine-tune the network backbone (UNet) only, keeping the VAE and text encoder frozen. Batch size is 4, training runs for 50 epochs, with a cosine learning-rate schedule, weight decay 0.01, and gradient clipping. Images are peak-action snapshots extracted at canonical times in each episode to minimize temporal ambiguity.

I.10.4 Alignment Losses.

We mainly use two loss functions for finetuning:

- $\mathcal{L}_1$: consistency alignment. After inverting both factual and counterfactual images to latent noise (u_a, u_b), we penalize $\|u_a - u_b\|^2$, enforcing invariance of exogenous factors. Note that this loss function is very expensive due to it needs to inverse two sampling pathes for each counterfactual data pair. In practice, we only apply $\mathcal{L}_1$ to one data point per batch of training data.

[1] specifically we use DDIM $\eta = 0$, also the deterministic samplers in Karras et al. (2022)

- $\mathcal{L}_2$: structure preservation at high noisy region (large SNRs). At early diffusion timesteps, we penalize discrepancies between denoiser predictions for factual vs. counterfactual trajectories, encouraging global structural consistency.

Both terms can be weighted with coefficients λ_1, λ_2.

I.10.5 Seed Abduction with Deterministic Diffusion.

Our multiverse alignment objective requires that consistent contrasts share the same exogenous noise ω. In deterministic samplers (e.g., DDIM), this means that two parallel reverse processes $(Z_{t,X=x_0})_{t=0}^{T}$ and $(Z_{t,X=x_1})_{t=0}^{T}$ can be initialized with the same ω, ensuring non-descendant content is consistent.

Given an observed impact frame $V_{X=x}$ with controller input c, let $Z_{0,X=x}$ denote its clean latent before decoding and $Z_{t,X=x}$ the noisy latent at step t. At each step, the denoiser $\epsilon_\theta(\cdot)$ predicts noise $\epsilon_\theta(Z_{t,X=x}, t, c)$, which is then used to trace the trajectory backward through the noise schedule. Iterating these updates recovers an estimate $\hat{\omega} = \mathsf{Abduct}_\theta(V_{X=x}, c)$.

Applying this inversion procedure to both members of a contrast yields $\hat{\omega}_{x_0}$ and $\hat{\omega}_{x_1}$, which ideally coincide. The seed-consistency loss $\mathcal{L}_1$ penalizes their distance, providing a concrete operationalization of the causal consistency principle within deterministic diffusion.

I.10.6 Caveats.

Deterministic inversion. We use DDIM with $\eta = 0$ for Abduct_θ, yielding an approximately bijective mapping between seed and latent trajectory under fixed conditioning and schedule. In practice, invertibility is approximate and sensitive to: (i) the precise noise schedule; (ii) classifier-free guidance settings; and (iii) conditioning (x, c). Hence $\hat{\omega}$ should be treated as a consistent *estimate* rather than a ground-truth latent.

I.10.7 Choosing the High-Noise Set S for $\mathcal{L}_2$.

We select S as either (i) the last k steps of the schedule (empirically $k \in [\frac{T}{3}, \frac{T}{2}]$), or (ii) all t with $\mathrm{SNR}(t) \le \tau$ for a threshold τ. A weighted variant uses $w_t \propto \mathrm{SNR}(t)^{-\gamma}$ and

$$\mathcal{L}_2^{\mathrm{w}} = \sum_t w_t \left\| \epsilon_\theta(z_{0,t}, t, x_0, c) - \epsilon_\theta(z_{1,t}, t, x_1, c) \right\|_2^2.$$

Under mild conditions, aligning score predictions at high-noise is connected to alignment in data space via Stein's identity.

I.10.8 Scope.

This is a feasibility study. We claim no pixel-level counterfactual identification and provide only qualitative illustrations. Future work may extend this approach to video sequences and more complex mechanics.

J Software Dependencies

J.1 Game Engine.

The game itself is implemented in `Pygame`, a lightweight Python library for 2D graphics and interaction. We chose Pygame because it enables rapid prototyping of turn-based combat mechanics, frame-accurate rendering of impact frames, and reproducible control of random seeds, all within a Python environment that integrates smoothly with machine learning workflows.

J.2 Causal Modeling.

To formalize and simulate the causal generative process underlying gameplay, we use the `Pyro` probabilistic programming library (Bingham et al., 2019). Pyro provides the primitives required to implement SCMs consistent with the game's causal DAG, including stochastic functions for exogenous variables, deterministic assignments for endogenous variables, and intervention operators. This allows us to align the game engine's execution trace with an explicit causal model, and to sample parallel-world contrasts in a principled manner.

J.3 REPRODUCIBILITY.

Both components are integrated in a unified Python codebase, ensuring that gameplay, causal modeling, and data generation can be run deterministically from a single seed.

J.4 GRAPH LIBRARIES.

We generate mDAGs, parallel-world graphs and counterfactual graphs using the `Y0` library (Hoyt et al., 2025).

J.5 GRAPH SERIALIZATION.

All graphs are serialized as directed graphs in JSON using a node-link format. Nodes are represented as JSON objects with keys for the node identifier and attributes, while edges are represented as objects with source and target identifiers and an edge type field. In the case of mDAGs and parallel world graphs, exogenous nodes are marked with the attribute `"exogenous": true`. For example, the above example of an mDAG with observed nodes $\{A, B, C\}$, one directed edge $A \to B$, and one hyper-edge $\{B, C\}$, is converted to a CL-DAG and serialized as follows:

```
{
  "nodes": [
    {"id": "A"},
    {"id": "B"},
    {"id": "C"},
    {"id": "N_{B,C}", "exogenous": true}
  ],
  "links": [
    {"source": "A", "target": "B", "type": "directed"},
    {"source": "N_{B,C}", "target": "B", "type": "directed"},
    {"source": "N_{B,C}", "target": "C", "type": "directed"}
  ]
}
```