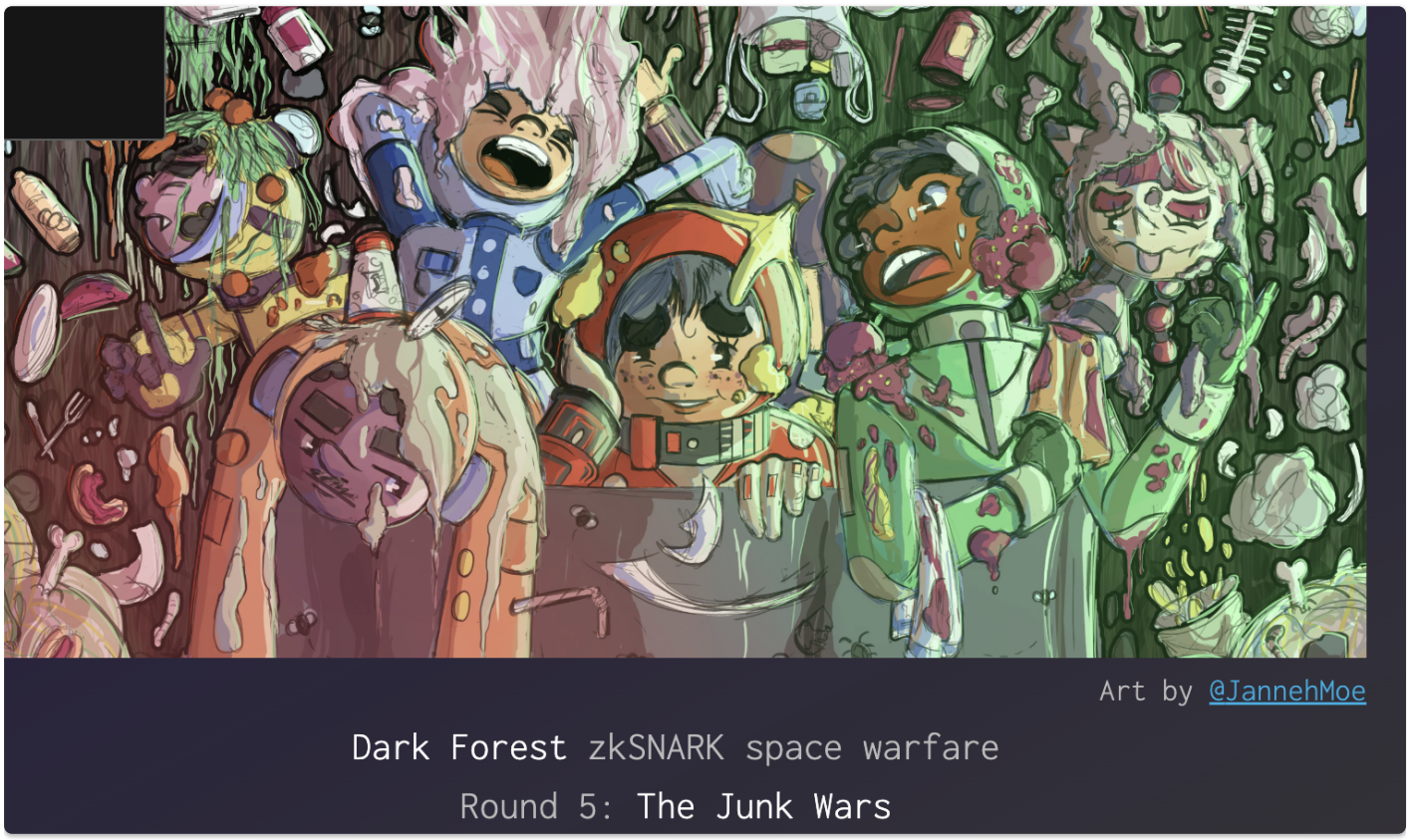
Lesson 8


always has been
wait zkml was just general purpose zk computation ?
imgflip.com

zk Games Introduction

Dark Forest



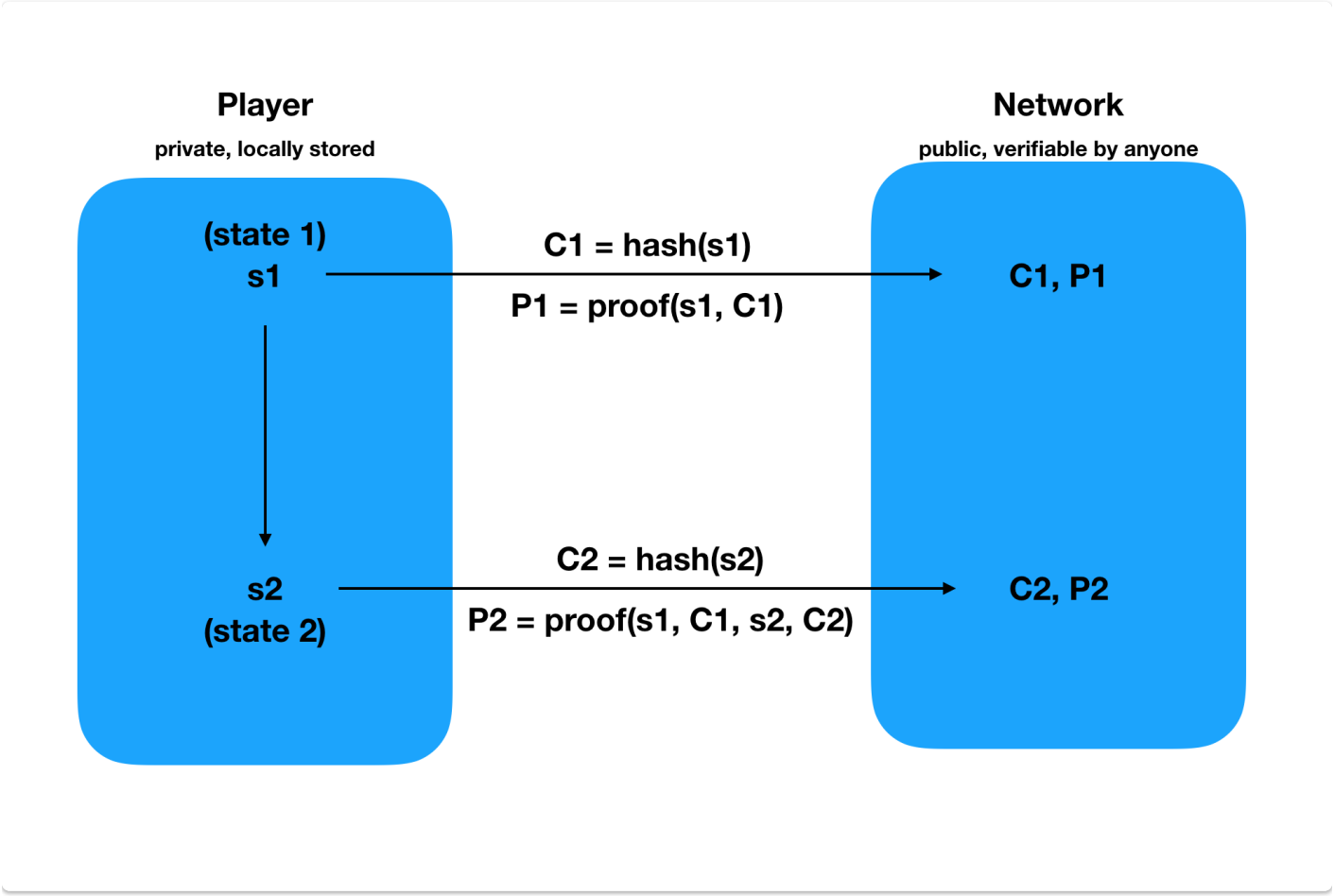
Dark Forest is an MMO space-conquest game where players discover and conquer planets in an infinite, procedurally-generated, cryptographically-specified universe.

Complete information games are games where all players know the full state of the game universe. For example, checkers and chess are complete information games, since all players always know where all pieces are on the board.

Incomplete information games (also known as "hidden information games") are games where

players may not know the full state of the world. For example, poker is an incomplete information game, since you don't know the cards your opponent has in their hand. Strategy games like StarCraft and EVE Online also fall into this category.

With zkSNARKs, players can keep private state while publicly submitting verifiably-valid actions. This allows us to build games like Dark Forest, which relies on a "cryptographic fog of war" secured with zkSNARKs.



In Dark Forest, players don't submit the coordinates of planets they conquer to the core

smart contract - rather, they submit commitments to their planet locations (by hashing the planet coordinates), along with zero-knowledge proofs that the hashes are valid. This keeps planet locations secret.

Cryptographic Fog of War

See article

Ingonyama claim that Dark Forest illustrated a new game mechanic, with the following requirements

- PvP: No dealer/trusted party
- Hidden information (fog of war)
- Deterministic state

Oblivious Transfer

In OT, a sender has n secret messages, and a receiver wants to learn k out of them (this is known as k-out-of-n OT) without learning the others, and without the sender knowing which messages were learned.
This is further developed through the use of KZG commitments to ensure verifiability of the messages.
For example

In its most general form, our Oblivious KZG interface between Bob (player B/Sender/Prover) and Alice (Player A/Receiver/Verifier) now looks like this:

Commit:

- Bob commits to a polynomial
- Alice commits to a polynomial

Probe: Alice sends a message to Bob containing a hidden set of indices she wishes to learn and a proof for valid probing

Prove:

- Bob verifies probing
- Bob uses Alice's message to sends a message containing the hidden result of the probing

Verify-decode: Alice decodes the values to be used in the game and verifies their correctness against Bob's commitment.

zkHunt

See video
From article
ZK Hunt started out as a simple idea for a new

way to do private movement in an onchain game.
Movement is performed by selecting a destination and confirming the path. Each move is submitted as a separate transaction, with a new move is submitted as soon as the previous move has been confirmed.
Units have a piece of state (their position) which can change from being public to private, and back again based on in-game actions. When a unit's position becomes private, rather than going from no ambiguity to complete ambiguity, it instead gains a limited amount of ambiguity that can be increased over time, due to the constrained nature of the tile-based movement. This allows other players to act with some degree of confidence over the unit's location, and greater confidence the sooner they act.

Example circom code showing a commitment

```
template Example() {
        signal input state; // Private
        signal input commitment; // Public
```

```
        // Calculates the poseidon hash
commitment from 'state'
        signal result <== Poseidon(1)
([state]);
        result === commitment;

        // The rest of the circuit can now
trust the validity of 'state'
        ...
}

component main {public [commitment]} =
Example();
```

Autonomous Worlds and MUD

Autonomous Worlds

Autonomous worlds are digital ecosystems governed by their own set of rules and entities. These worlds,operate under a unique “diegesis” or the set of conditions that define what exists within their boundaries.
They are based on 3 components :

A Blockchain state root.
The commitment and summary of all the diegetic entities, which compose the world’s state.

A Blockchain state transition function.
The definition and execution of the rules, which govern how we introduce and change diegetic entities.

A Blockchain consensus mechanism.
The agreement between a world's stakeholders that a state transition is valid. This makes everyone a participant and a decision maker.

Autonomous Worlds workshop video

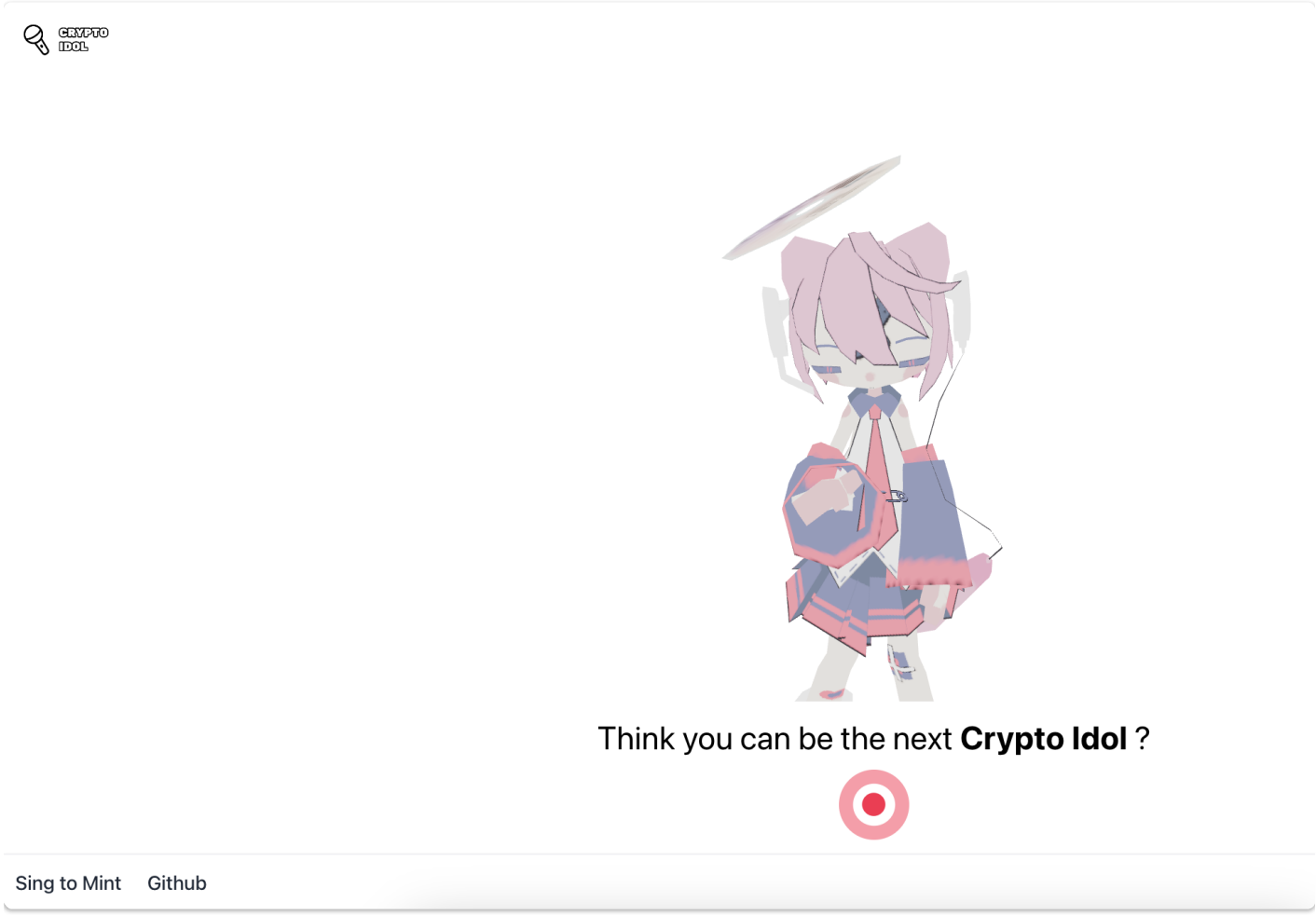
zkML Games

From article by Lattice

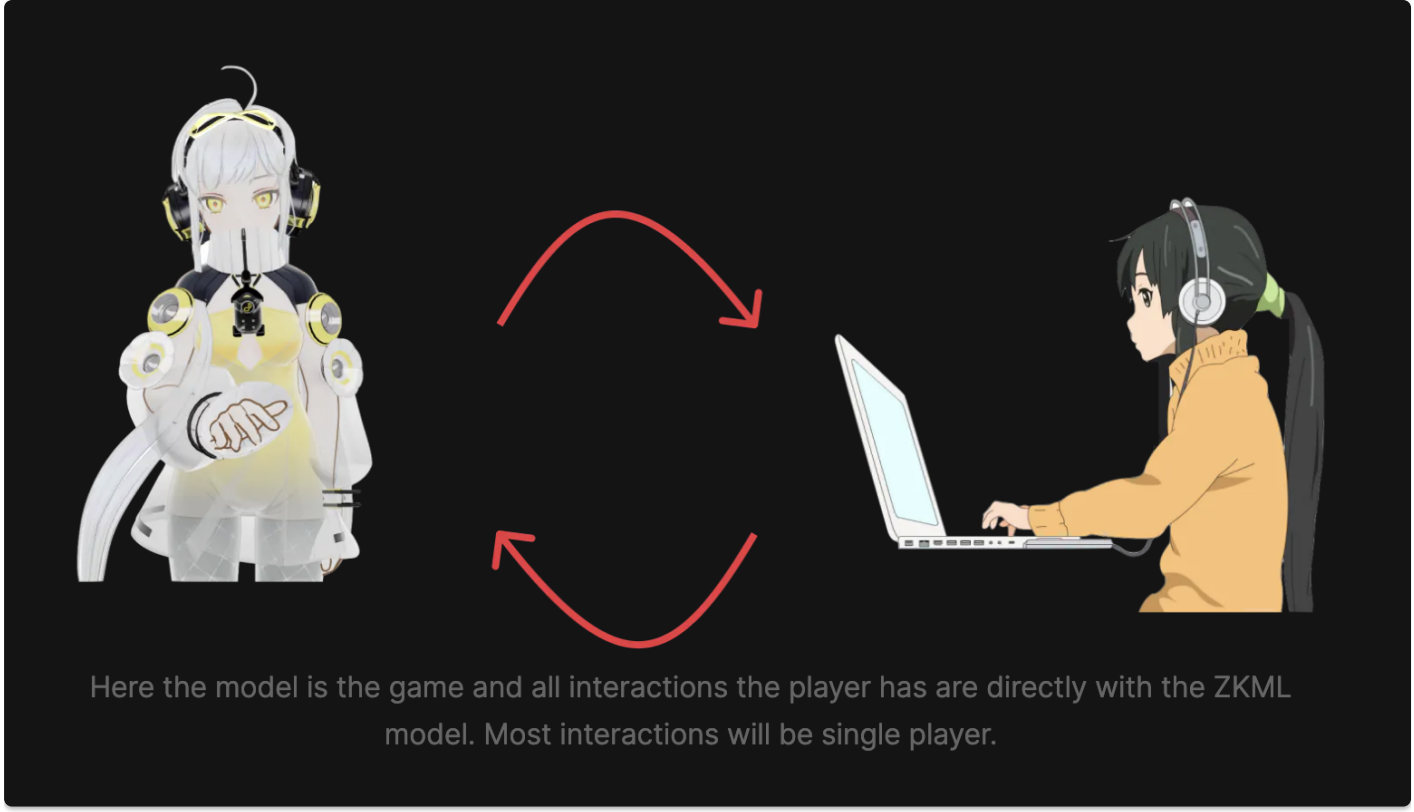
3 scenarios for the use of zkML

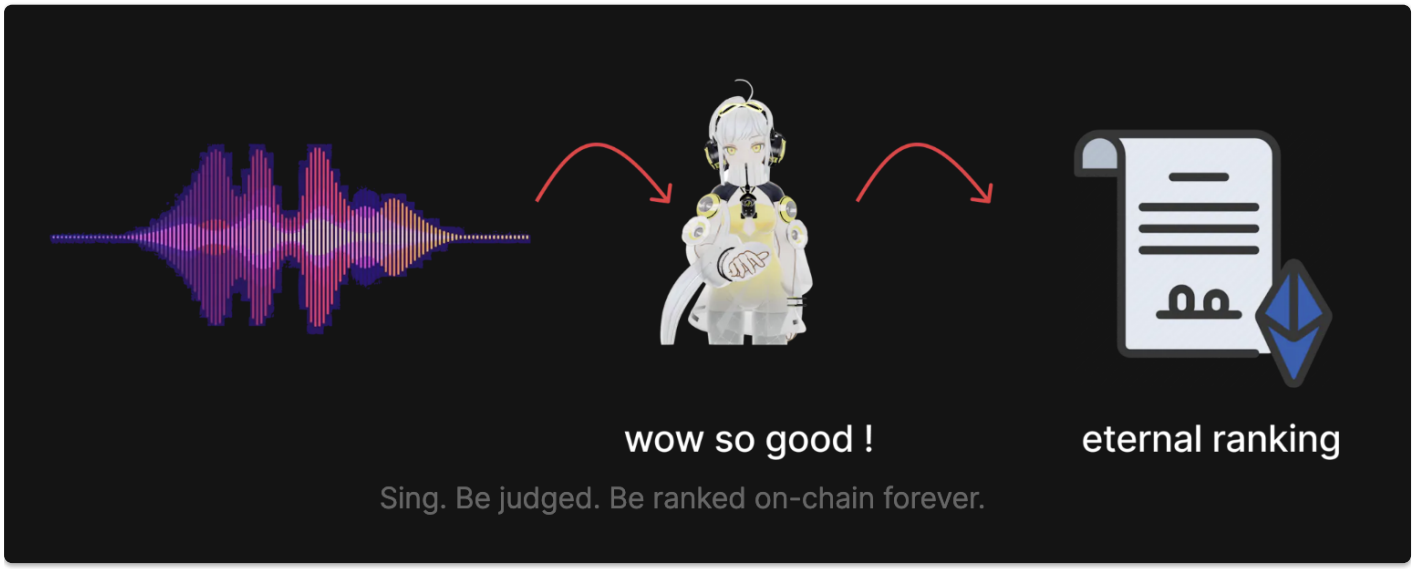
(1) The model is the game,
(2) zkML as digital physics, and
(3) zkML for narrative and lore.

The model is the game



Here, players interact directly with the ZKML model, and this interaction constitutes the entirety of the game dynamics.

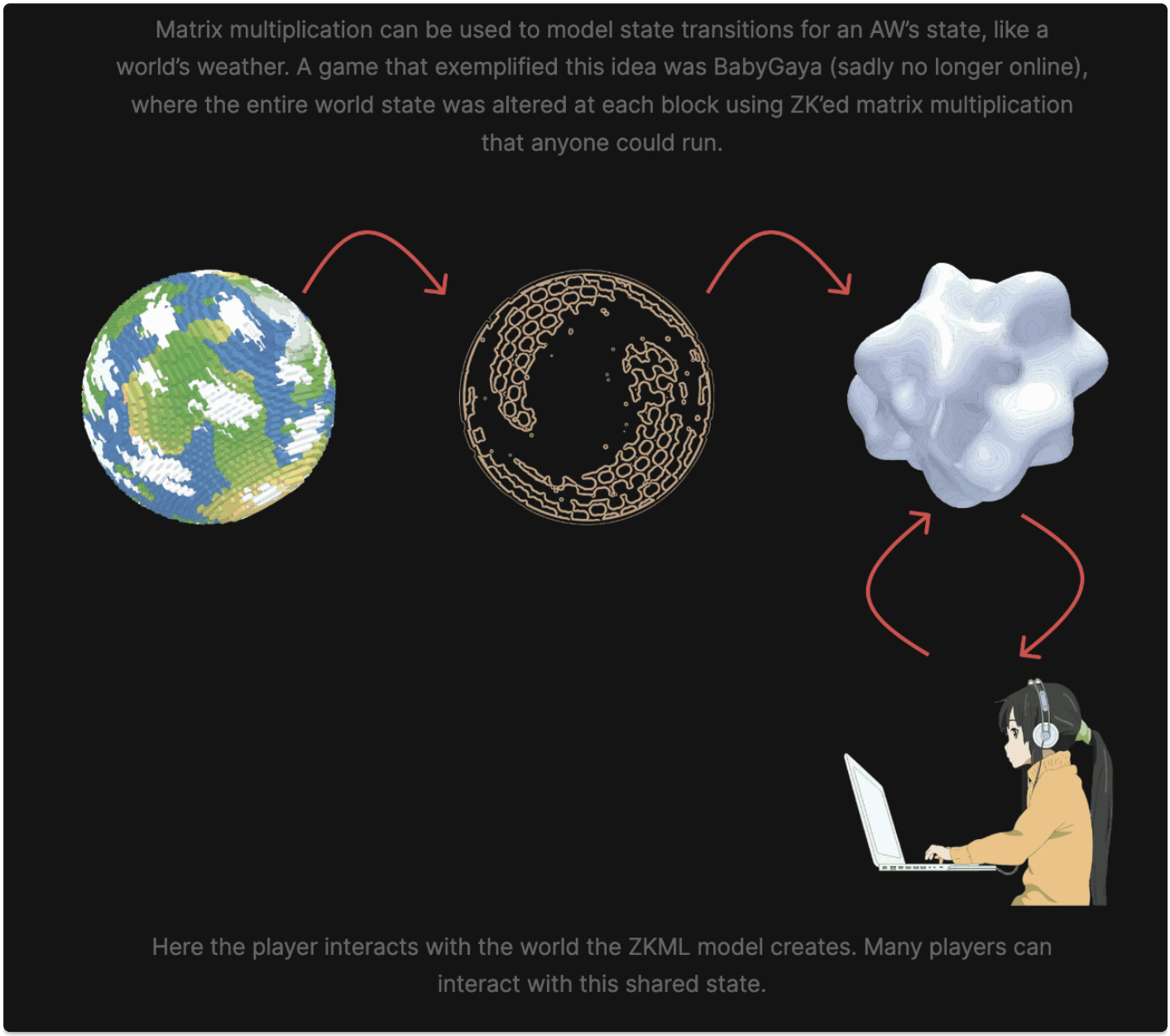




zkML as Digital Physics

Lattice designed a simple onchain-tictactoe library,
where a neural network is trained on tictactoe telemetry data to recognise valid or invalid game patterns, such that games can be played off-chain and then submitted and settled on-

chain in their final state.
See this tutorial and this one



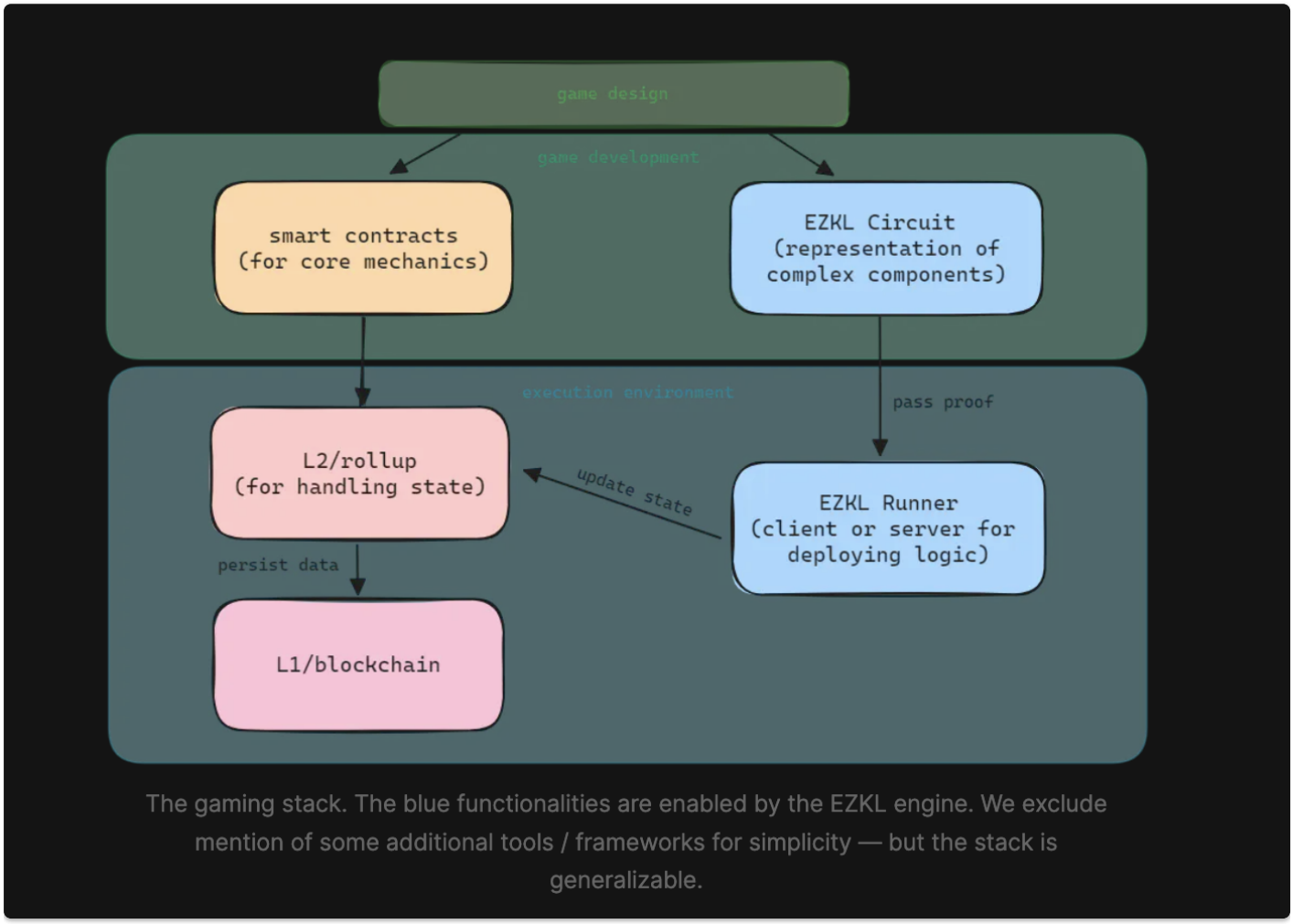
zkML for narrative and lore

Here zkML is leveraged to build a world's lore and narrative.
ML models can be used to create sophisticated agents / NPCs and concurrent storylines that are nuanced elements of a larger whole.
This could for instance be an a language model writing a world's narrative as the game

progresses, or writing storylines for individual NPCs.
An example of this is a generative model, described here, which models the voltage activations (given sensory inputs) of the C. Elegans nervous system.

zkML Game Stacks

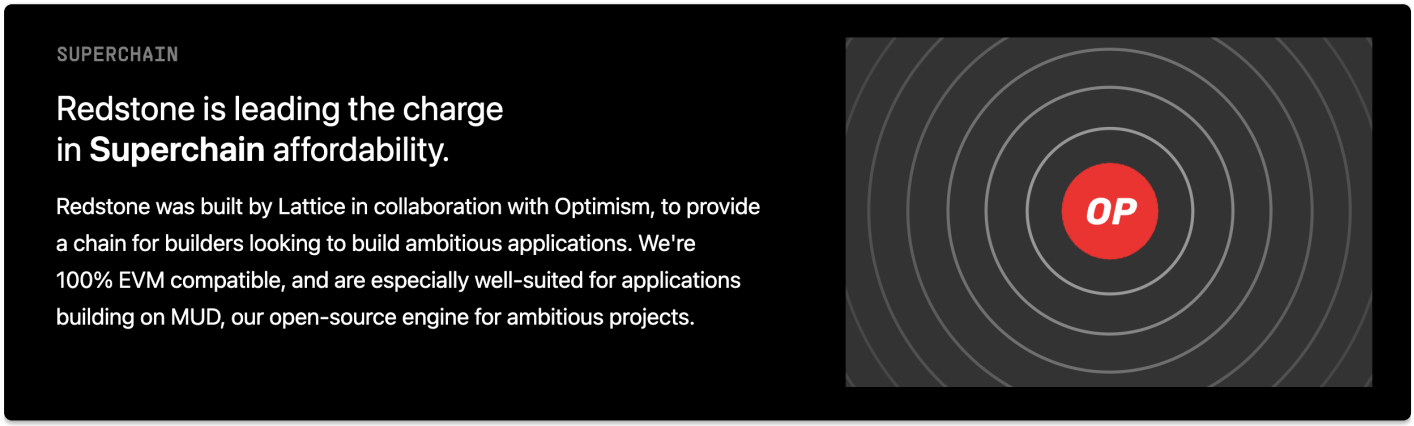
From article



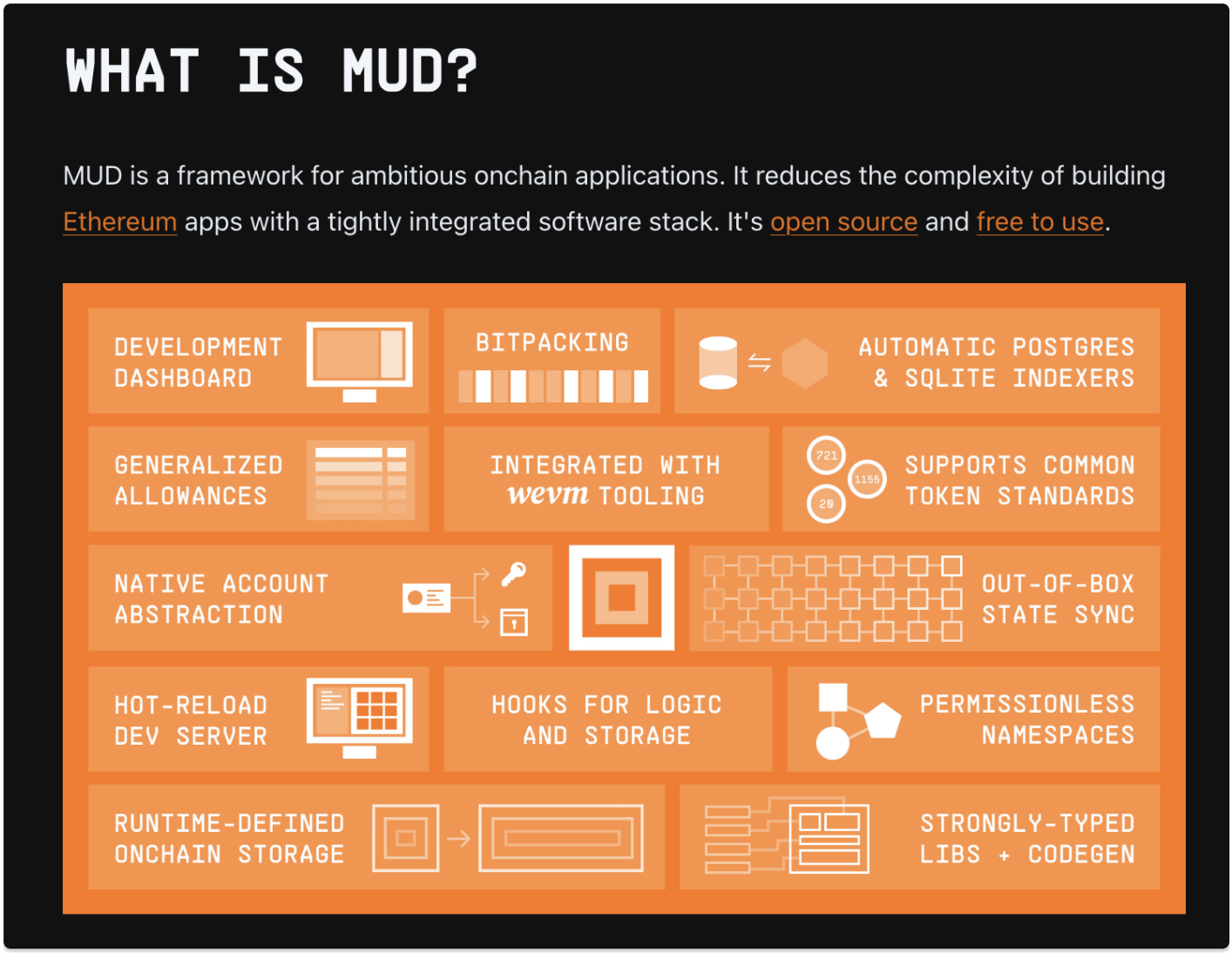
The gaming stack. The blue functionalities are enabled by the EZKL engine. We exclude mention of some additional tools / frameworks for simplicity — but the stack is generalizable.

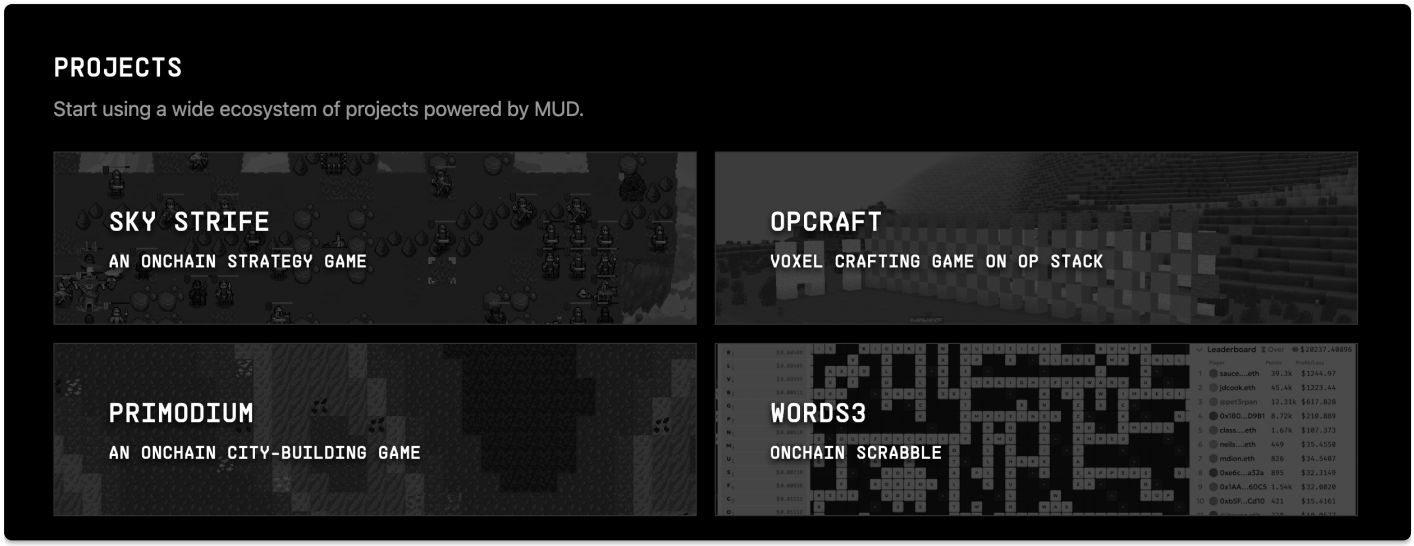
Redstone

See site



MUD





See Repo

See Docs

MUD apps are autonomous worlds, infinitely extendable by default. They come with access control, upgradability, hooks, plugins, and a suite of great developer tools.

MUD is maximally onchain: the entire application state lives in the EVM, and the only requirement for clients and frontends is an Ethereum Node.

OPCraft

OPCraft, a fully onchain voxel world, was built by Lattice in 1.5 months. While it didn't have the security and auditing requirement of financial applications, it handled more transaction throughput than most apps on Mainnet ever did in their lifetime: in 10 days, OPCraft players

made 3.5 million transactions, filling the MUD onchain database with billions of gas worth of storage; with the client seamlessly handling synchronizing that state back.