The intersection of games and crypto feels rich with possibility. Vitalik was famously inspired to create Ethereum after Blizzard nerfed his WoW class. Warcraft was not “critical infrastructure,” but we expect virtual worlds to emerge that are: housing trillions of assets and millions of jobs. It is difficult to imagine them existing under the thumb of centralized platforms.

Of course, decentralized applications sound good in theory. The most compelling in practice are those uniquely enabled by crypto: applications that could only

have emerged onchain. Despite a lot of narrative momentum, it turns out to be difficult to identify precisely what onchain games uniquely enable.

Why put games on a blockchain?

This post reflects the state of our thinking on the question.

Designing for Emergence

Some games achieve long enduring engagement by giving creative users the tools to generate new content (“UGC”) themselves. Two of the richest sources of UGC - mods and open economies - are the attack vectors where we see a plausible case for onchain games.

Mods

Mods allow third-party developers to implement content beyond what was envisioned by a game’s original developer. Several genre-defining games (e.g. DoTA, LoL, PUBG) have their roots in modded versions of other games. Others, like

Roblox, have transitioned from being games into being mod developer platforms. While game studios usually focus on production value, engaged modder communities bring diversity and novelty: Netflix vs. YouTube.

Minecraft is an excellent concrete example. Simple game mechanics are conducive to tinkering. Mods that extend those mechanics can be recombined into functionally new experiences. Many popular Minecraft servers feel nothing like the original (prison escapes, battle royales, etc.).

However, even Minecraft has a limitation: players can't contribute new mods to existing servers. They have to start a new server to introduce changes. As a result, the Minecraft "universe" is fractured across many parallel, mostly non-interacting private servers.

There are good reasons that most modern games implement mods like Minecraft, through instancing (of new servers) rather than scripting (of existing ones). It is hard to ensure that player-contributed code is compatible with native rulesets (exploits are particularly challenging). Ruleset updates may break the mods built on top of them. Limited compute resources need to be rationed intelligently.

Still, instancing leads to fragmentation. Every mod that spawns a new server is in competition with all the other servers for players' attention. Modders can't just ask what additions to a world are interesting, they have to ask whether a change is worth starting a new server around.

Consider that many potential mods may only make sense in context

- adding to a world that already exists. For example, say you run a restaurant in some Minecraft server and want to add a new item to the menu. It doesn't make sense to start a new server to do so, because you'd need to convince all your customers to move to it too, and they probably won't because they have their own customers and commitments in the existing server.

Game worlds that fragment lose the ability to incrementally expand.

Open Economies

In-game economies are another dimension with almost limitless potential for player creativity. We'll use EVE, the first game to hire full-time economists on staff, as an instructive example.

Across an informal combination of in-game systems and external infrastructure, EVE players manufacture and trade goods; claim, lease and contest territory; and organize everything from industrial collectives to warmongering pirate gangs. Tasks as simple as hauling resources have full player-run corporations dedicated to their fulfillment - complete with customer service, SLAs and employee benefits.

Players have come to EVE for 2+ decades not because of new content from the developer, but because of the rich emergent social and economic world driven by the other players.

However, even EVE's economy has some significant limitations:

1. Limited In-Game Primitives.

Any transactions that fall outside the set of primitives enshrined by the developers (e.g. lending agreements) have to rely on informal, unenforceable trust networks. That trust limits the complexity and scale of emergent economic structures.

1. Regulatory Constraints.

Due to the compliance headache, the vast majority of games (including EVE) simply block players from off-ramping any assets or exchanging in-game goods or services for fiat. Those that do allow it only do so on onerous terms and have to maintain large compliance departments.

Onchain Games

There are many different potential forms of crypto-native games. Our focus is the most crypto-native end of the spectrum: fully onchain

games, whose state and logic live entirely on open smart contract platforms.

It's also important that onchain game mods can be permissionlessly deployed as their own contracts alongside the base game logic. And that users opt-in to mods by choosing which their clients will interpret (rather than an admin deciding for them).

So, why put games on a blockchain?

We think the strongest case rests on two points:

1. Composable modding.

Players can add mods to onchain games without asking for permission or fragmenting their state. Onchain infrastructure and smart contract developers are conditioned for the challenges of allowing players to permissionlessly upload code: security audits, access control, resource metering, etc. Traditional games are not adapted to this environment and are unlikely to restructure themselves around supporting composable modding.

1. Permissionless open economies.

Rather than being limited to the set of in-game primitives defined by a game's developer or having to rely on informal and unenforceable agreements, players can use smart contracts to create a game's economy. Moreover, sovereign player custody of game assets obviates compliance overhead.

Rather than being "uniquely enabled" by onchain games, composable modding could be a path-dependent innovation. While traditional games could theoretically support composable modding, they currently don't and have no incentive to change that. The model will only get explored by necessity (i.e., in crypto).

The combination of composable modding and permissionless economies is what could yield large emergent onchain game worlds. Modders will take simple ruleset substrates and expand them with new mod content. They'll have access to real money, proximity to DeFi markets, and freedom to experiment. The resulting economies could be highly complex and reflexively incentivize the creation of accumulating content. Once it's legible that there is money to be made, activity could explode, in the same speculation-experimentation cycle that birthed other crypto application ecosystems.

Most onchain games discourse zooms into that optimistic future at higher resolutions. We are more interested in getting specific about what stands in the way: the open problems that need to be addressed before it's plausible for large-scale game worlds to emerge.

Open Problems

Technical constraints limit game design.

The standard reasoning for why no onchain game has broken out yet is that the technical infrastructure isn't ready, and thus most games get stuck at the proof-of-concept stage: simple gameplay, buggy clients, limited engagement from players and mod developers.

Existing infrastructure and developer tooling is limiting. In particular, the EVM is slow and clunky, existing Solidity data models are not conducive to complex game development, and no mainnet chain is a viable deployment target for games (given high costs and low scale).

Fortunately, we have line of sight on solving most of these problems. Rollup scalability and cost reduction progress is already legible to most of the crypto community. There are also many teams working on games-specific infrastructure. For example, Lattice is developing a Solidity framework and compatible tooling (indexing, state sync, etc.) that could simplify EVM game development. Others, like Dojo, Argus and Curio are also working on infrastructure platforms.

Other problems feel more fundamental to the nature of onchain games. In particular, certain properties of permissionless blockchains prevent support for mainstay game design mechanics:

- Incomplete Information:

a crucial mechanic in many games. Existing solutions have unacceptable drawbacks (e.g. DarkForest's cryptographic fog of war devolves into a hardware mining race).

- Automation & Collusion:

fundamentally cannot be prevented. Bots can't be distinguished from real players, and players can't be guaranteed unique. Developers must build games that aren't broken by bot metas or by Sybil collusion.

- Ticking:

blockchains are driven by asynchronous transactions. Most traditional games are built around tick-based game loops independent of player interaction.

It's possible these constraints breed creativity and game types we've never seen before, similar to how MakerDAO and Uniswap emerged from DeFi without skeuomorphic TradFi analogs. However, traditional games are less technically and legally constrained than TradFi - they've already been able to explore more of the map - so it feels less likely that a novel onchain game will emerge from uncharted territory. We think progress likely needs to be made on the limitations to give onchain games a fighting chance at breakout success.

Research Directions

1. TEEs.

Although very heavy-handed for the task, Trusted Execution Environments (TEEs) are the only practical option for permissionless private computing on public blockchains.

1. MACI.

A mechanism originally designed by Vitalik Buterin to improve the collusion resistance of on-chain voting systems, MACI, could be adapted to onchain games and potentially improved upon through tight integration into relevant game systems.

1. Custom Rollups.

It seems feasible to get some form of traditional ticking game loops onchain by modifying rollups to include global ticks as part of their state transition function (with no gas cost). Other games-specific modifications might be interesting as well.

Using ZKPs to enable private state is another existing research direction. However, we're skeptical that the non-programmable privacy they enable is sufficient to unlock meaningful game mechanics. The current difficulty of writing circuits also limits their usefulness.

Composability is inherently financializing.

In a system that is open for the whole world to interact with, incentives are not just a suggestion. An incentive is more like a physical law such as gravity or entropy. If there is some aspect of the system that is not incentive-compatible, it is only a matter of time until it is exploited.

— Nikolai Mushegian, bank.dev/principles

Smart contract blockchains are highly adversarial, financializing environments. This is not a path-dependent artifact of degen culture: it is a mechanistic consequence of permissionless composability. As applications primarily premised

on composability, onchain games will be exposed to these incentives at a primitive level.

In a vacuum, before considering the impact of mods, onchain game developers will need to contend with the inescapability of real-money markets, MEV (frontrunning incentives), and economic exploits. The bar to design an incentive-compatible onchain game is likely quite high; potentially equivalent to designing a secure DeFi product.

The second-order problem is even trickier. Onchain games are designed to be modded, and mods will carry their own emergent incentives. Even a developer that deftly manages core game incentives doesn't know what will get built on top of it - or what incentives will be introduced. (Enabling such unpredictable emergence is in fact their goal.)

To draw another analogy to DeFi, consider an oracle. The oracle might be economically secure (unprofitable to manipulate) in a vacuum. However, the oracle can't predict what applications will integrate or compose with it. If a lending protocol uses the oracle to trigger liquidations, the oracle inherits the manipulation incentive - frequently fatally. Similarly, when a Minecraft mod introduces a MEV incentive to mine a block first, it will affect the gameplay of all players, even those whose clients don't interpret the mod.

This is a tough problem to fix. Attempting to permission or otherwise constrain who can develop mods for an onchain game is directly at odds with maximizing emergence (the reason to build onchain in the first place).

We suspect that incentive compatibility will be a defining challenge of onchain game design. Some traditional games avoid real-world money markets because of the compliance headache; many more simply think they aren't fun

. Onchain games will need to figure out how to leverage financialization pressures without being consumed by them.

Research Directions

1. Antifragile Design.

Core game mechanics can influence, but not dictate, what mods emerge on top of them. The degree to which onchain games can encourage pro social mods is an open question, as well as what types of game designs are least susceptible to corruption by N-th order incentives.

1. Permissioning.

A direct attack on financialization is simply to control who can play an onchain game, and who can deploy new code to it. There is an obvious tradeoff with emergence, but it may be necessary to at least experiment with games in a walled garden before exposing them to harsh permissionlessness. And we can get clever about how we permission (beyond simple whitelists).

1. Orderflow Auctions.

Instead of trying to prevent emergent incentives, we could try to harness them. For example, by forcing all game transactions through an orderflow auction that remits its proceeds to the game's economic faucets. Any value created by mods would be pumped back into the game's economy (e.g., by repurchasing scarce commodities). The downside is that the underlying behaviors could still hurt gameplay (i.e. players are mining coal to fund solar).

Metagames tend towards stagnation.

Onchain games will necessarily have longer release cycles than traditional games. They want to maximize emergence, and frequent breaking updates disincentivize creators from investing in worlds. Updates will also require new audits. And many onchain games builders view permissionless "autonomy" - no admin keys, no updates, infinite persistence - as a goal in and of itself.

So, for technical and philosophical reasons, onchain games will sit somewhere on the spectrum of autonomy between "never updated" and "infrequently updated."

The moonshot case for a maximally autonomous onchain game is that the right ruleset substrate can spark an engaged modder community and endless new emergent experiences. Even experiences that may only be possible with decades to percolate untouched.

However, most games are managed to prevent metagame stagnation. Players are already very good at figuring out optimal strategies for traditional games; now MEV will provide an additional, explicit incentive. These strategies tend to be static and uninteresting. A truly autonomous world loses the ability to control the metagame at any level - Vitalik was perhaps worried about the wrong issue with his Warlock.

Rather than an innate design goal, we suspect the crucial question will be: to what degree can successful onchain games be autonomous?

Research Directions

1. Seasonality.

Many traditional games deploy upgrades on a cadence of months to years (like WoW expansions). The main trade-off is disincentivizing players from building complex mods which could become broken in future seasons. We think this is one of the more promising approaches to iterative experimentation.

1. Automated Feedback.

Just as Bitcoin automatically adjusts its difficulty in response to hashrate, onchain games could build anti-stagnation retargeting into the core game mechanics. This is not specific to onchain games - centralized games' ability to do it is strictly more powerful - but they could innovate by necessity.

1. Novel Governance Mechanisms.

Although we are generally governance minimalists, there could be an interesting space of non-token-based systems to explore. The ability to create new rules could even be a part of core game loops (e.g., Mao). Some early attempts already exist; for instance, Topology tightly integrated a bespoke governance system into their onchain game Isaac

Should Games be Fully Onchain?

There might be accessible onchain game designs that leverage permissionless composability elegantly. These worlds could flourish as open economic incentives continuously drive new content, and persist indefinitely on censorship-resistant and credibly neutral blockchains.

However, it's also possible that there isn't enough unique enablement to justify threading the needle through the open problems (which are not trivial). Again, in contrast to traditional finance, games have always been highly experimental. So, the standard onchain games must meet to prove their existence worthy feels ironically higher than DeFi - which addressed a previously closed market.

If fully onchain games are not a viable approach, the reasons to be excited about them could get expressed in less "onchain" ways. The games that work may use smart contracts minimally, or not at all. Offchain game infrastructure with NFT assets and interoperability with DeFi could be the right practical setpoint. Especially if elements of an offchain game are controlled by onchain assets, smart contract-based coordination around just the assets could still be powerful.

Finally, whether or not games end up fully onchain, the patterns they explore - especially composable modding - could drive innovations in traditional game design. Traditional studios may see the potential and be willing to dedicate significant resources to redesigning offchain engines around supporting composable mods. Potentially coexisting with, outcompeting, or spiritually succeeding onchain games.

Conclusion

We see a lot of hard problems, but still have an intuitive sense that onchain games could leverage blockchains to create weird, novel outcomes.

We're excited to explore all the frontiers of crypto-native games with other builders. We're also more interested in building games than infrastructure - games that we would play ourselves.

If this sounds exciting to you, reach out: {charlie,doug}@paradigm.xyz

Acknowledgments

Thanks to our colleague t11s for many hours of input on this piece, as well as Matt Huang, Dan Robinson, Dave White, and Frankie for discussion and review.

Thanks also to Will Robinson, GuiltyGyzoa, Rafael Morado, Scott Sunarto, Robert Miller, and 0xhank for their feedback and Ludens, Phil Daian, John Guibas, Arthur Roeing Bear, and Hilmar Veigar Pétursson for discussion.