# PROJECT LITHIUM

"Playmat".

Design intent.



https://noetic.garden/

PROJECT LITHIUM is an application that allows people to play and design tabletop games, with a powerful focus on card games.

# It should:

- … allow users to pick up and play a game with ease, with a user experience focus on ‘getting out of the way’. Each task along the way of a normal tabletop experience should:

    - feel natural,

    - obey platform conventions,

    - be recognizable as the user switches contexts (say, from a desktop computer to a tablet or headset), and:

    - where the experience of play cannot match physical components, to introduce as little friction as possible rather than trying to be skeumorphic.

- … have a built-in, ‘on-top’ design experience that moves from importing game components into designing them. It should:

    - be progressively disclosed without feeling intrusive.

    - recognize that these games have acts of authorship, such as deckbuilding, and ensure that the UI for these moments in the game allows for expression both as a player and as a game author.

    - integrate into existing workflows without requiring onerous reupload or re-pairing steps.

- … strive for powerful immersion.

    - in particular, it should target headsets as first-class experiences.

# But:

- It should avoid physical accuracy except as a useful affordance.

    - there are gestures and component placements that are literal, and gestures and component placements that are approximations of symbolic arrangements or intentions. This application should not confuse the former for the latter, and should focus on interpreting and producing clarity around the symbolic arrangements as much as possible.

    - This means games of dexterity or precision may not be representable easily with this app. This is okay. This is a trade-off we are making to focus on the majority of card game experiences.

- It should not be a graphics design application.

    - it should integrate with existing tools, including tools hosted on other machines, rather than replicate them.

    - it should strive to understand and maintain design intent without friction, for example with powerful defaults that produce predictable component results from the output of a graphic export.

- It should avoid onerous pregame rituals:

    - once a game has been prepared, it should not require installation for a remote user. Joining a game and obtaining the game setup should be synonymous.

    - it should clearly provide for both whole-game authorship, and acts of player authorship (e.g. bringing your deck) for that game, without conflating them or making players go through more onerous design interfaces than strictly needed.

# Technical forecast:

This application will be developed at least initially in Swift for Apple operating systems and platforms As a cost- and effort-containment measure, we are targeting only the following, by leveraging SwiftUI & RealityKit:

| Primary (Expected) | Secondary (Possibly facing the losing side of trade-offs:) |
|---|---|
| macOS, | iOS |
| visionOS, | |
| iPadOS. | |

That said, part of the design effort will at least attempt to remove a hard dependency between behavior-driving code and specifically SwiftUI. This may already be required if we find out that the optimal solution for visionOS support is to run that particular set of interactions entirely in RealityKit, but is a matter of policy anyway to not tie the code down specifically to a UI environment.

Tertiary platforms are not part of the initial target. The hope is to eventually, once done with acceptance on the above, be able to either build decoupled code with Swift toolchains for Windows and Android, or port the core behaviors to a different language (e.g., Rust or C++) that can be deployed to these platforms.

| Tertiary (Explicitly not sought on first pass:) |
|---|
| Meta Quest |
| PC VR (Win32) |