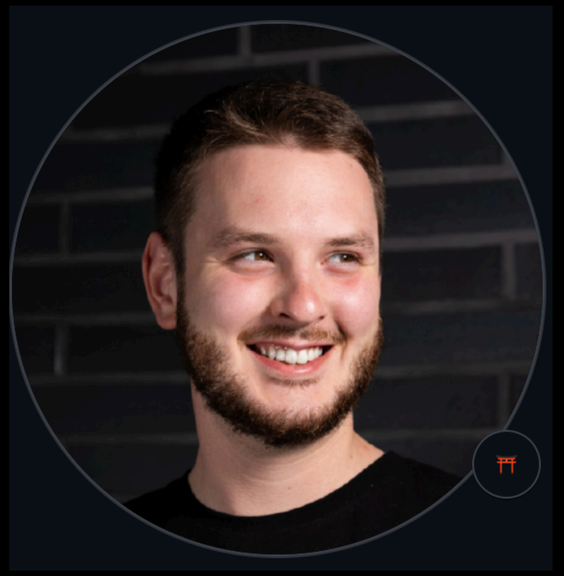
DAOs in gno.land

ETHBelgrade Meetup - March 5th, 2025

gno.land

Who am I?

- Leon Hudak
- DevRel Engineer @ gno.land
- 3.5 yrs in Web3, started with Ethereum
- https://github.com/leohhhn



Today’s agenda

- What is gno.land?
- Why Go?
- Why Gno?
- DAOs

What is  gno.land ?

- A new blockchain, running a custom virtual machine - the GnoVM
- Allows for writing smart contracts in Gno, an interpreted and fully deterministic version of Go
- Deployed code is completely transparent and viewable via gnoweb, gno.land’s universal frontend

Why Go?

- Go is a simple and straightforward language with a minimal learning curve
- Go has a large developer community, and lots of readily available resources, most of which can be used 1:1 for learning Gno
- Solid collection of performant, well-known standard libraries
- Gives lots of power to developers in spite of its simplicity

More on Gno…

- Modeled after Go 1.18
- Interpreted instead of compiled, allowing seamless composability on-chain
- Currently does not support all go features, such as generics & goroutines

Gno

```
package alice

var x int

func GetX() int {
        return x
}

func SetX(n int) {
        x = n
}
```

```
package bob

import "alice"

func IncrAlice() {
        x := alice.GetX()
        alice.SetX(x+1)
}
```

Why DAOs?

- Centralized governance structures may fail in an AI and internet-driven world.
- Decentralized systems provide resilience, transparency, and adaptability.
- While they may currently lack the efficiency level of centralized systems, they provide a framework to, over time, reach improved levels of efficiency and coordination. In addition, to placing more responsibility and oversight on individual collectives, rather than a corporation.

The Philosophy of Governance

- What does it mean to create a decentralized governance system for everyone?
- Governance should be:
  - Adaptable: Able to evolve as needs change.
  - Inclusive: Welcoming towards diverse voices and perspectives.
  - Sustainable: Built for long-term stability and transparency.
  - Customizable: Provide options for different governance needs: flat, hierarchical, tiered…

DAOs in gno.land

- GovDAO: Decentralizes responsibility and authority over time for the maintenance and evolution of the platform
- CommonDAO: A general-purpose DAO framework for dApps
- Gno.me DAO: A use-case specific DAO for content management and ecosystem development
- … and more to come!

GovDAO

- Governance baked into the chain
- Invite-based tiered system: T1, T2, T3
- T1 can invite T2, T2 can invite T3
- Reputation based - invited based on contributions to the project
- No tokens involved, except for renumeration down the line

WIP

Gno.me DAO

- An organizational DAO framework to manage different roles and expertise within a “company”
- An ecosystem development DAO for content publication, budget approvals, and community engagement.
- Tree-based DAO (expert subDAOs)
- SubDAOs allow for more structure and remove time overhead of voting on different topics by everyone, by delegating different types of proposals to DAOs with the right expertise

WIP

CommonDAO

- A modular governance framework for dApps.
- Allowing anyone to easily create DAOs, with a few lines of code.
- Used in gno.land’s boards app - reddit-like forum platform with subDAOs for each board.

WIP

Using CommonDAO

```
package socialmedia

import (
	"std"

	"gno.land/p/nt/commondao" // pure package with definitions
	cmndao "gno.land/r/nt/commondao" // DAO factory
)

// DAO exposes methods for handling DAO-related
// functinality, such as proposing to add new members,
// pushing a custom proposal to the DAO, etc.
var DAO = cmndao.New("My DAO")

func init() {
	DAO.Members().Add("g125em6arxsnj49vx35f0n0z34putv5ty3376fg5") // @leohhhn
}
```

WIP

Using CommonDAO

```
// RemovePostProp implements ProposalDefinition
type RemovePostProp struct {
	postToRemoveID string
	reason         string
}

func (p RemovePostProp) Execute() error {
	_, removed := posts.Remove(p.postToRemoveID)
	if !removed {
		panic("Removal failed")
	}

	return nil
}
```

```
func ProposePostRemoval(id, reason string) error {
	caller := std.PreviousRealm().Address()
	if _, err := DAO.Propose(caller, RemovePostProp{
		postToRemoveID: id,
		reason:         reason,
	}); err != nil {
		return err
	}

	return nil
}
...
```

WIP

Using CommonDAO

```
// CommonDAO defines a DAO.
type CommonDAO struct {
→       id           uint64
→       slug         string
→       name         string
→       parent       *CommonDAO
→       members      *addrset.Set
→       propIDs      seqid.ID
→       activeProps  *avl.Tree // string(proposal ID) -> *Proposal
→       finishedProps *avl.Tree // string(proposal ID) -> *Proposal
}
```

WIP

gno.land's Governance Experiment

- A continuous learning process:
  - Testing governance models in real-world scenarios.
  - Adapting based on observed behavior and community needs.
- Progressive decentralization through DAOs.
- Enabling adaptive and scalable governance.
- CommonDAO as a blueprint for future Gno-based governance models.

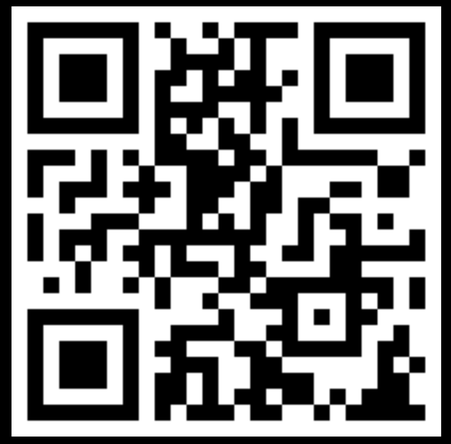
WIP

Final remarks

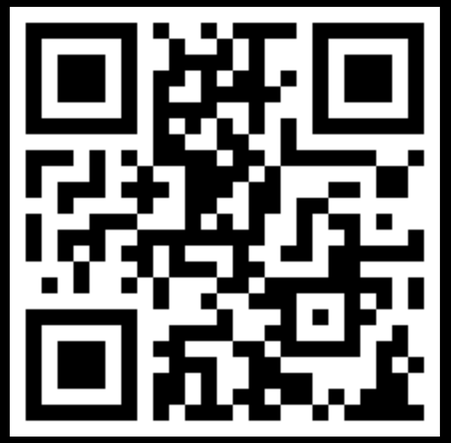
- …. DAOs are hard.
- Creating them involves organizational skill, technical expertise, knowledge of world history, and much more.
- DAOs strive to create a better governance in the world.
- Participate in your DAOs of interest. That’s the only way to keep them alive.

WIP

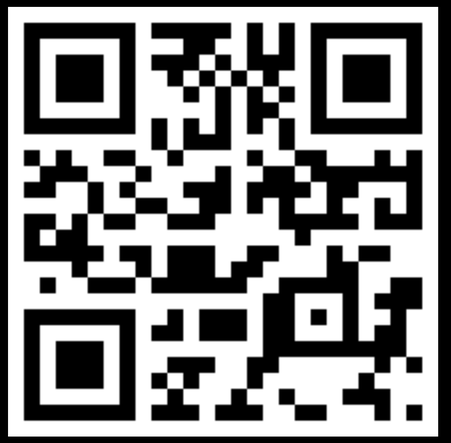
Thanks!



gno.land



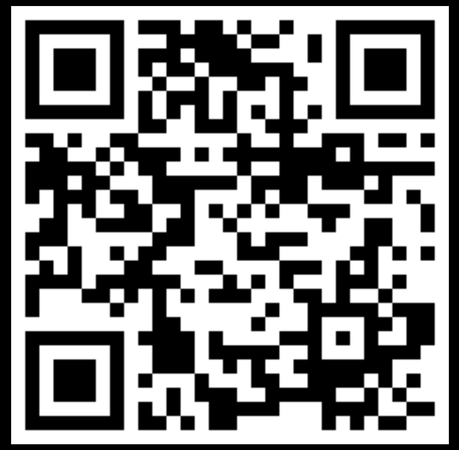
GitHub



Discord

Call for Contributions

- We are looking for early adopters to contribute to gno.land.
- Be among the first to build useful dApps, libraries and work on protocol level problems and make an impact on the future of gno.land.



Linktree (Grants, student program, etc.)