H20 Decentralised Liquidity Engine

Problem

Do you have a liquidity issue?

The problem with token liquidity

- **You can’t get what you want** either solutions are proprietary, custodial and heavily regulated (for both you and the custodians), or cookie cutter, zero context generic onchain algorithms.
- **CEX sucks** unless its binance but they list 500 projects only and then you still pay for market making and everything else, listing fees are huge and ongoing liquidity fees are high
    - Large hacks are common despite the premium price tag
    - Custodial solutions are opaque and attract regulators
    - Centralized solutions have proven to fall to the temptation to play games with client/customer tokens behind the scenes
    - Many CEX can only service one or limited regions, not global
    - Many CEX only allow deposits and withdrawals on limited chains/networks
- **DEX sucks** cos it costs you a fortune to provide double sided liquidity, you lose out with impermanent loss and traders destroy the pool so you have to top it up again and again
    - AMMs calculate slippage based on total deposited capital so need **more** capital to service **small** trades
    - AMMs have zero context, goals or rules about when/how they will trade so they always lose money for LPs every time the market moves
    - AMMs don't understand regulations or incentives/tokenomics
    - Onchain order books incur extreme gas costs to micromanage orders, whether manually or with a bot
    - Inflationary rewards given to LPs always lead to the reward token being dumped, which converts impermanent loss to permanent loss, and a vicious cycle where ever more inflation is required to attract the same liquidity

How it should work

- Anyone wanting to provide liquidity (LP) for a token decides for themselves what **their** rules are
    - Exactly what price is offered at all times, slippage is always 0, DEX fees are always 0, the price is whatever the rules say it is even if that leads to the token "selling out"
    - Who is eligible to be a counterparty, or even give discounts to preferred counterparties

    - How much capital is made available overall, per counterparty, per time period, per strategy, per trade etc.
    - What tokens they offer and accept at a given price, not restricted to a single pair
    - Front running, MEV, and other manipulations of the mempool **never** cause the price to change for bots or traders (unless the rules say it should)
- The LP pays gas **once** to deploy a strategy and traders/bots pay gas for all trades against the strategy
    - Simple strategies should be low gas to to take a trade and the gas for complex strategies paid by the market taker
- The LPs make all risky decisions themselves and have appropriate risk mitigation tools
    - Non-custodial onchain end to end, LPs deploy their own rules, tokens, everything themselves with their own private keys
    - Capital should be sandboxed in dedicated vaults so that strategies can only touch preallocated funds, funds can be dripped into vaults over time to mitigate contract risk
    - Which immutable contract to use for the full lifecycle of the strategy, no admin keys, upgradeable or pausable contracts
    - The LPs should be able to write their rules in a programming language no more complex than existing accounting/financial tools such as excel or pine script (Trading View)
- The revenue/inventory from sales/purchase from each strategy automatically feed into other strategies, either as a simple closed loop, or a complex network of vaults with inputs and outputs

Structural issue

There are middle men between the project, project token and community

- DEX, CEX, Market makers are all middle men, you pay them
- If things go down, right now it’s the middle man’s fault
- Projects don’t have a direct connection to their community with secondary market tokens sales, it’s all via a market maker
- Price / liquidity becomes expensive and inefficient because people have to profit from it
- Running a private bot might help your token with its price (temporarily) and volume but if your goal is to build liquidity then you need to convince other people that the trade is going to be there when they need it

Proposition

- Adwords for token liquidity

    - A/B test strategies
    - Double down on what works
    - Discard what fails
    - Spectrum of strategies including primary sales, algorithmic sales, treasury management, market making, buyback curves
    - End to end liquidity for a project

Benefits

End to end, aware and interactive based liquidity powered by a new technology stack:

1. Deploy strategies quickly using Interpreter, Rain and Rain studio (read Rain core concepts and visit Rain studio)
2. Express strategies using Rainlang (Learn Rainlang)
3. Create varied and innovative expressions using the full spectrum of tools in the studio e.g. simulation, context, rainlang, words and/or create your own words (visit Rain studio)
4. Write various buy and sell curves that operate according to the rules you set, and that are aware of and interact with each other to create cheaper liquidity
5. Deposit into order book vaults to commence strategy, and withdraw to complete (Use Order Book on Rain studio)

Which offers you the following benefits

- Custody over your assets
- Efficient and inexpensive liquidity through self-aware curves
- Bots you can run for a profit and flash loans to connect into third party liquidity
- Design and control over the strategies you run
    - Write yourself or
    - Use others to write
    - Use our team to write
- Turn on and off strategies at will
- Start / stop engagement at any time
- No vendor lock in

Deliver solid risk management together

- Risk management, no need to expose all capital to the DEX at one time, can drip into strategy as needed
- No admin keys
- Reduced hacking risk
- Reduced honey pots
- Spread over different strategies

Technology

Technology design

- N:N token pairings
- (future/roadmap) vault locks for trustless tokenomics-as-an-order
- Arbitrary daisy chaining of orders, not just a closed loop
- Conditional orders based on onchain data
- Base protocol is extensible at both interpreter and wordpack level
- Stateful strategies, e.g. track how much has been paid over time or average price etc. over time
- Chain agnostic, all evm+sg networks supported
- No IL, orders always clear at exact prices you specify or not at all
- Orders cannot be front run or MEVd because calculations happen onchain
- Gas efficient, one strategy deployment costs comparable to 1 or several uniswap swaps, runs indefinitely, arb bots and market takers pay gas for all storage, calculations, etc. each trade
- Full transparency in market making, order strategies are onchain and publicly independently verifiable using rainlang sdk/gui
- Run multiple strategies in parallel, A/B test performance in real time, reallocate capital to best performing
- KYCd strategies, only offer tokens to counterparties that meet regulatory requirements
- Stateful strategies per-counterparty, offer discounts or even free/dividends to specific wallets according to allocations or other logic
- AMM-like behaviors possible by writing price curves in terms of current vault balances, introspection possible through OB context
- Fully whitelabellable, directly call takeOrders from your own website/gui and present price forecasts using the offchain JS simulator
- Replace your own orders with new orders atomically to upgrade your own tokenomics as orders, even pointing to new interpreters with new logic available
- Nobody has any admin keys to OB, unpausable fully trustless permissionless DEX contract
- 0% fee flash loan support for all tokens deposited in OB vaults
- 0 fees, no DEX token

Offer

This is our current perspective on our offer. Goals, properties and processes. The trial phase of 2 months is designed to test these goals, properties and processes and they will be revisited at the end of the period.

Goals

- We are not a CEX
- We are not a DEX
- We don’t run bots
- We are a market maker

Solution

- Deploy sophisticated algos on a ‘DEX’ exist.https://github.com/rainprotocol/rain-protocol/blob/develop/contracts/orderbook/OrderBook.sol
- The strategies are all based on arb or ppl taking orders and the user pays they can either setup their own strategies, which would be cleared by some arb bot when the price curves intersect
- You specify a vault id per token when you deposit, so you can have up to uint256 max different vaults per token so basically unlimited daisy chaining as in, the inputs of one strategy can be the outputs of another
- You can do N:N tokens, so you can say "i'm happy to accept any of USDT/USDC/DAI for this price" it doesn't have to be pairs, just lists of "equivalent" tokens that you set prices for
- Think of it more like a budget you could deposit way more than N tokens into the vault, then just have a stepwise auction
- There's no admin keys on OB, any changes will be a fresh deployment so you'll just have to withdraw any funds and re-add to the new setup

Properties

	Owner	**Support**
Strategy writing	Project	H20
Strategy testing	Project	H20
Strategy deployment	Project	H20
Clearing bot	Ecosystem	Project & H20

Analytics/reporting dashboard (read-only)	H20	-
Vault balances	Project	
Strategy progress & results	H20	-

Processes

Process	**Owner**	**Platform**
Creation of strategies in consultation with project	H20	Telegram/Studio
Legal consultation	H20	Telegram
Deploy order	Project	H20 or Studio
Deposit to vault	Project	H20 or Studio
Withdraw from vault	Project	H20 or Studio
Clear order	Ecosystem and/or project	Don’t know don’t care
View vault balances	Project	H20
Monitor performance	Ordernomics / H20 and/or project	H20

What you pay for

Paywalled access to H20

- Creation of strategies
- Deployment of strategies
- Generic legal for this approach
- Access to read-only dashboard
- Access to performance reports
- Future features e.g. simulations

Ecosystem support

- Maintenance of Rain and Rainlang tooling incl Studio
- Maintenance of the Orderbook contract
- Maintenance of the generic Orderbook front end

Incentives

- Clearing bot incentivised for market to take
- …

2 month pilot

We’ve been honing the underlying infrastructure so that it is capable of bespoke or personalised market making. The 2 month pilot gives you the unique opportunity to work with the Rain team to deploy strategies with the underlying infrastructure and to provide input on the product we build. Hopefully you love the pilot and become our first customers. Details:

- H20 decentralised market making pilot
- Starts 14 March
- Ends 14 May
- 10 projects
- $5,000 USD in total
- Strategies will be written in rainlang
- Strategies will be co-created
- Stragies will be deploying on order book via Rain Studio
- Unlimited strategy experiments
- <=5 strategies Rain team can assist in writing
- Unlimited strategy deployments
- Project pays gas on strategy deployments

Who we are looking for

- Projects that are annoyed with the current market making scenario
- Projects that are prepared to explain and try writing the strategies they want to do

14 May our aim is to have a working product a project could then use. Those who have been pilot of the pilot will receive $5,000 USD discount on their next invoice.

More details

How does it work?

- With an AMM you have to put up TKN / ETH and then TKN rewards for liquidity providers. This is an alternative where you don’t need to risk your assets providing liquidity to create liquidity; all liquidity is put upfront, so we had situations where novice traders would buy/sell large chunks in 1 go without realising what they were doing. One side of the pool would empty without catalysing flows.
- Ideally, would like to put in our own algo for liquidity - one that is more responsive, gradually increasing or decreasing based on flows and ideally, place very small amounts of liquidity at any given time. And keep adding liquidity incrementally, in response to trades.
- The most basic version of this would be to have:
    - 1 TKN : 1 USDT at t=0
    - Someone comes in and buys the TKN at t=1. We then have 2 TKN : 0 USDT
    - The algo adds 0.9 TKN: 1 USDT (new price established) at t=5 (time interval can be tweaked).
    - It's like time based but also volume based - you wait to sell out at price X then add some more liq at price X + Y after duration t
- This is the advantage of Centralised exchanges currently. Market makers have decently sophisticated algos, which let us choose these strategies. But on DEXes, it just doesn't exist.
- Basically allow people to build or sell positions, but when we do it in this way, it facilitates flows.

Why does onchain matter?

If you are trying to market make on an existing decentralised exchange, then why do you need an onchain order book?

- What if the conditions to your rules change between when the bot submits the order to the mempool and the txn is in a block?
- What if you want to make the rules trustlessly visible and provable to your community buying your token?
- What if gas is 1000+ during a market meltdown and your bot has to cover this?
- Does your bot maintain its own single sig hot wallet of token inventory to place these trades? How much capital does it need to have access to?

- Does your bot get an exact price you asked for or merely whatever the AMM is offering according to its own algorithms and slippage?
- Is there any potential scope at all in the direct bot model to add more participants to the list of entities that can take the trades? or are you locked into the single bot model forever?
- Are you making careful backups and properly handling the state of your trading strategy? Is the bot carefully coded to also read the chain and modify its own storage to match onchain storage changes so you can accurately track outgoings and incomings inline with the blockchain? Are you handling uncle blocks also?
- Where does the bot run? Is it in aws? Are you _sure_ you won't get shut you down overnight due to regulatory or T&C changes? If it happened to the biggest data centre in germany, why couldn't it happen to amazon?
- Does the bot provide a way for you to talk about liquidity with your community and get other people to do similar things or even copytrade with their own orders?
- Can a DAO propose and vote on the logic a bot is running and independently confirm it is operating as intended with 100% uptime?

Some of this is why we want vault locks soonish. We are thinking about if we wanted to NEAP private equity, there's a big difference when talking about valuations of the equity between "yeah we run a private bot on aws and it hasn't placed any trades so the token must be worth a lot" and "we have this trustless immutable public auction running 24/7 backed by revenue and so that's the lower bound on our equity valuation".

Questions

- Every time we add/remove liquidity to the pool, there's gas fees on the DEX. On Polygon it's not a big deal, but Ethereum, it would be crazy expensive no? No you set the strategy up and deposit tokens once so you put in vault, then the strategy you write controls flows in/out of your vaults (i'd guess a strategy like this is a bit under 200k gas to deploy and deposit/withdraw token is less than 100k gas it's all based on arb or ppl taking orders and the user pays)
- Aren't we tried into the AMM algo for price? I guess in the v3 versions of Uniswap and Sushiswap we can add liquidity in narrow price bands - but is that how we would achieve it? It's an orderbook model, so no it's not an AMM. It's an orderbook where the amount/price is recalculated in realtime within the transaction, according to the rules. So you can have your users buy direct with the takeOrders function on the contract, or let arb bots hit external liq on a dex it has a flash loan feature built in, so arb

bots can loan the vault balances to clear against external liq sources whenever the market moves
- Regulatory issues. Don't want to have the regulatory burden of being an exchange ourselves. By using a DEX we insulate ourselves? it is this smart contract onchain with no admin keys, you're not the DEX, you're putting an order on the DEX
- So putting orders on a DEX with these price limits? yeah basically think of it like a programmable limit order. limit orders are possible on a DEX, they have been around a long time, the issue is that historically the logic to manage a limit order as a single price/amount requires a bot and a lot of gas and potential to be front run in this case the algo itself is the limit order, and runs onchain, so no bots, no extra gas, no front running
- Now that I realised it's an OB and not a liquidity pool order. Potentially, we could deploy on a dex with small amounts, and let Bots arbitrage with our main DEX and CEX, see how it goes.? yup, you can manage risk by just depositing tokens gradually into the vaults, on top of what the order does. it's an OB and also there's a takeOrders function where the caller says a specific order they want to take and it does the trade direct from your vault to their wallet so that's handy for building a dedicated GUI around your liq fyi the vault balances are available to rainlang so you can express e.g. x*y=k if you wanted, and even have a virtual balance as one side of the pool, so the "order" would look just like an LP pool on an AMM fyi the vault balances are available to rainlang so you can express e.g. x*y=k if you wanted, and even have a virtual balance as one side of the pool, so the "order" would look just like an LP pool on an AMM

What might an order look like?
- Each time we hit a new batch delay sales for 24 hours
- Price is exponential, 10% increase each batch
- To avoid dust/griefing issues we always offer a minimum number of tokens even if that technically exceeds the batch limit
- Can't move to next batch until current batch is cleared
- so like 0-1000 tokens => index 0 => 1.1^0 price
- 1000-2000 tokens => index 1 => 1.1^1 price
- 2000-3000 tokens => index 2 => 1.1^2 price
- and the dust guard offers the min number of tokens at the lower price, but still counts the total tokens sent towards the overall lot progression
- you pass this to the rainlang JS function that gives you the binary data to put into the DEX as the order
- on the roadmap is to have "locks" on orders, so you could actually lock in an order and prove to your community that the tokens in the vault will disperse according to your rules

- currently it's just an offer, you're free to cancel the order at any time, it will stay active until removed

How does this differ from something like hummingbot?

- https://hummingbot.org/
- HummingBot is an open source market maker
- H20 allows the project to create any liquidity strategy including primary sales, algorithmic sales, treasury management, market making, buyback curves - it’s a liquidity engine that includes ability to market make
- H20 is all onchain with the benefits outlined here

What might an expression look like?

```
/* Calculate IO */
/* 24 hour delay on new batches */
seed: call<0 1 3>(),
: ensure(gt(now() add(get(hash(seed 1)) mul(60 60 24)))),


/* get info about current batch */
batch-index batch-remaining: call<1 2 2>(0),


/* always offer at least 10 tokens even if that's beyond the current batch */
amount: max(batch-remaining scale-18<0>(10)),


/* exponential growth */
/* (1.1)^batch-index */
/* Power function x^y with y simple integer */
/* https://github.com/PaulRBerg/prb-math */
io-ratio: fp-powu(scale-18<1>(11) batch-index);


/*
 * Handle IO
 * Actual order size available here after cross referencing counterparty order
 */
seed: call<0 1 3>(),
seen-batch-index seen-batch-time: get(hash(seed 0)) get(hash(seed 1)),
total-amount-sent batch-index batch-remaining: call<1 3 2>(scale-18-dynamic(out-token-amount
out-token-decimals)),
/* if we're in a new batch record current time as batch time */
:set(hash(seed 1) if(gt(batch-index seen-batch-index) now() seen-batch-time)),
:set(seed total-amount-sent);


/* Calculate batch */
amount-this-batch:,
seed: call<0 1 3>(),
amount-per-batch: scale-18<0>(1000),
total-amount-sent: add(get(seed) amount-this-batch),
batch-index: div(total-amount-sent amount-per-batch),
batch-remaining: mod(total-amount-sent amount-per-batch);


/* shared constants */
seed: 0x897055f9a919d8fca1a0eb0bfeed3ba67d50c4c5667c072fa52bd5612787a690;
```