WEWESWAP Whitepaper V1

WEWE love the yields

Strategy

Will be deployed on **weweswap.com**.
Base.Univ3 Positions will be vampired to WEWESwap using CHAOS points farming tokens.
First pool will be WEWE-WETH, then expand to USDT, and other memecoins.

Overview

WEWESWAP needs to achieve:

1) Simple experience for users to enter/leave
2) Passive experience, with no active management required
3) Consistent yield paid in "cash"

WEWESWAP changes the perception that a "Liquidity Position" has a set of assets, which are exposed to Impermanent Loss (IL). Users become married to their principle instead of treating it as working capital.

The solution is that users should treat their Liquidity Positions as "working assets that earn yield". These working assets may appreciate in value, or they may depreciate in value (IL), very similar to real-world assets (businesses etc).

The yield received must only be in USD stablecoins - the holy grail of yield. Thus users get to realise their yield and in an asset they perceive as "cash" and stable.

Problems

UniV1/UniV2 Pools design (Constant Product, CP) allowed passive liquidity with exposure to IL, but was not capital-efficient. Yield was accrued in the pool itself in two different assets.

UniV3/UniV4 Pools (Concentrated Liquidity) moved to capital-efficient liquidity, but increased the experience complexity, by adding price ranges, NFT Positions and more. One good move was to accrue the yield outside the pool, so the user

could always claim fees (in both tokens) which was “saved from IL”. However, CL makes liquidity “active” so retail-users were chased out, and they find themselves going back to CP pools (eg most memecoin pools are CP).

CL is much more capital efficient, but has more experience problems:

1) Users have to provide multiple assets to enter (experience) and at different amounts
2) Users have to understand and choose a “price range”
3) Users have to understand that their liquidity can “go out of range” and re-balance
4) Users accrue yield to two tokens
5) Users have too many choices for “pools”

Is it possible to build an liquidity protocol that serves the basis of liquidity and yield for 100m “retail” users?

Solution

To achieve this

1) It must be easy to enter a pool, with just one asset (ZAP)
2) There must be a simple selection of pools, so users don’t get decision paralysis
3) The yield must always accrued outside the pools so it is not lost to IL
4) The yield must always be paid in USDC, as “cash back”
5) The user must be auto-rebalanced to always be in range (so it is passive)

Pools are only non-USDC options to the user, so they perceive that they are purchasing a “working asset” with a potential “yield”. All the pools are paired with USDC, to allow ZAPing of the rewards back to the yield asset - USDC.

The only permutation between pool types are price ranges and this is performed by the Zapper Aggregator, which chooses price-ranges on behalf of the user, depending on the ASSET type:

“STABLE”: eg {USDT}-USDC // +/- 1% range; 5bps
“BLUE-CHIP”: eg {WETH}-USDC // +/- 10% range; 30bps
“SMALL-CAP”: eg {WEWE}-USDC // +/- 50% range; 100bps

Liquidity Ownership in Pool

Users add their working asset into a pool. The liquidity manager contract distributes into only three chosen bands of liquidity (not seen to the user). The three bands are Narrow, Mid, Wide; in order to achieve effective CL.

The user is given ownership units of the total liquidity for the pool. This tracks their total ownership of the principle AND of the accrued yield. The Liquidity Manager contract owns the liquidity positions, not the user.

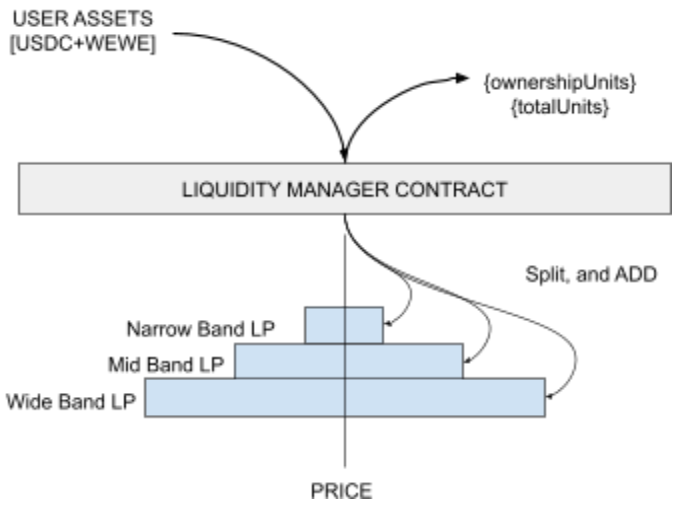
The user adds their assets (single or dual) via the LM Contract. The assets are "zapped" by an aggregator to be ideally in the correct ratio prior to deposit. This splits it into three bands and gives the user an Ownership. Any left-overs are refunded by LM back to user.

Auto-rebalancer Bot

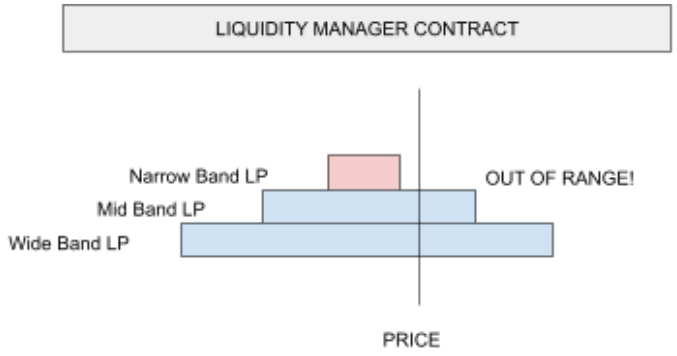
The Liquidity Manager contract has the simple re-balancing rules, that any bot can call:

1) The band must be out of range.
2) The band must only be moved perfectly back in range using Zap.
3) The bot can charge up to 10bps to move.

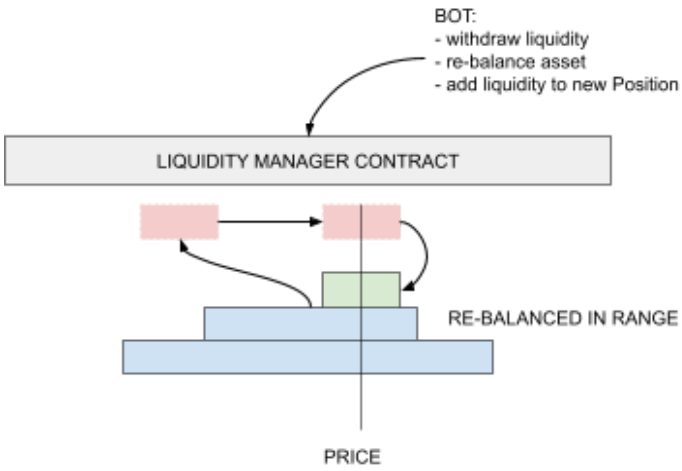
Any leftovers, are left in the LM contract - can be collected, the next rebalance.



The price moves and one of the bands becomes out of range for some time (eg 1 hour).



Any bot can call into the LM Contract to (burn NFT) remove the Band. The bot re-balances the assets correctly using Zap and charges a small fee. Then bot fetches the latest price range for the band, and adds the liquidity back.



Liquidity Bands

There are 3 bands in each pool. The target price range is as per the Pool Type. The LM contract stores the desired band targets (but allows the owner to update the strategy if required). When the re-balancer moves the band, it will fetch the latest desired band strategy from LM Owner and re-deploy liquidity as per the params. The bands are kept away from each other (or being a modulo of each other, to ensure they are never re-balanced at the same time).

Pool Type	Target Price Range	Fees
Stable	+/- 1%	5BPS

Blue-Chip	+/- 10%	30BPS
Small-Cap	+/- 50%	100BPS

StablePool - 1%

Band	Range	Lower	Upper
Narrow	40% Target	0.996	1.004
Mid	100% Target	0.99	1.01
Wide	170% Target	0.983	1.017

Blue-Chip - 10%

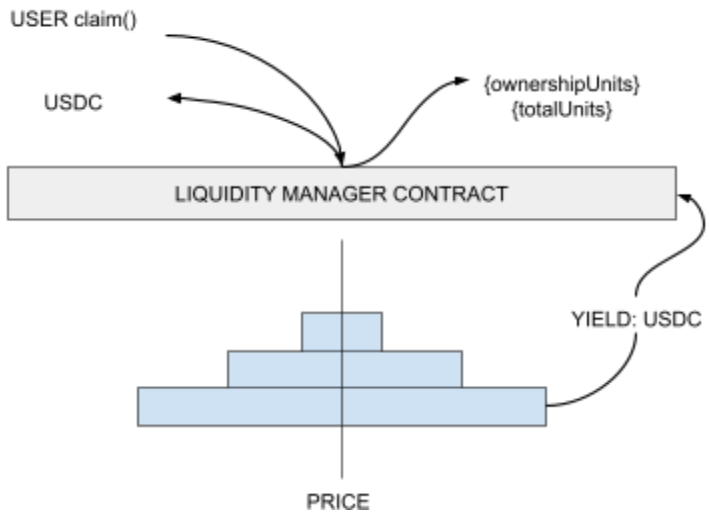
Band	Range	Lower	Upper
Narrow	40% Target	0.96	1.04
Mid	100% Target	0.9	1.1
Wide	170% Target	0.83	1.17

Small-Cap - 50%

Band	Range	Lower	Upper
Narrow	40% Target	0.8	1.2
Mid	100% Target	0.5	1.5
Wide	170% Target	0.3	1.7

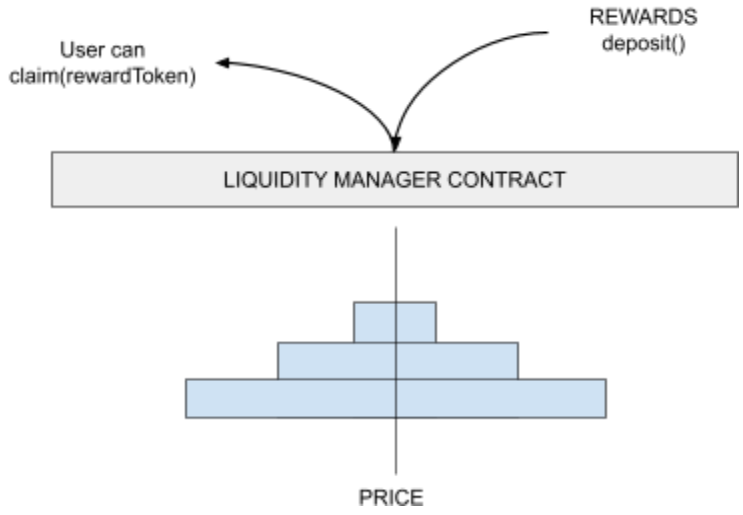
Yield

All liquidity fees in the pool are paid back to the Liquidity Manager Contract, but swapped to USDC only. Thus the user accrues USDC yield only, and at any stage, they can "claim" their USDC yield back. The user computes their share of the fees by assessing their share against total value of the vault.



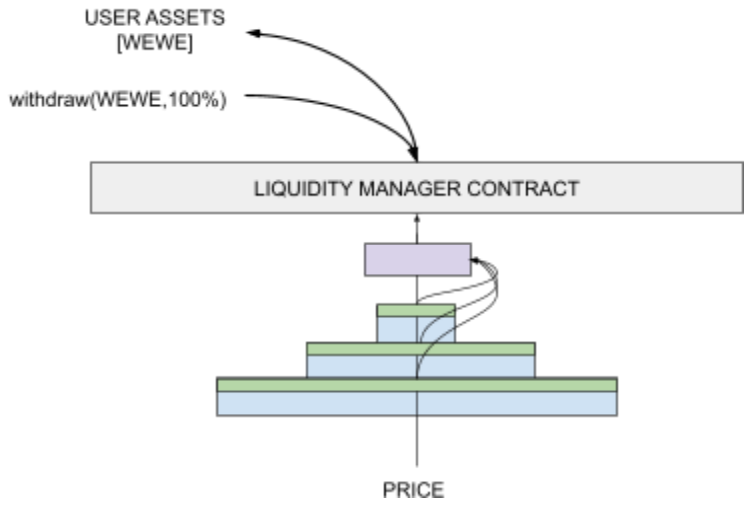
Subsidies

Initial users will earn a points token (CHAOS), which will be redeemed via the Liquidity Manager Contract. The subsidies are needed to bootstrap liquidity, until fees (USDC) take over. Anyone thus can “add” a subsidy token, and users can “claim” exactly like the USDC mechanism. CHAOS will be “streamed” in via a decay function.



Withdrawing

User can withdraw their position any time by assessing their ownership units. It is withdrawn equally from all positions, swapped to one side (if requested) and then withdrawn.



Tokenised Vault

Tokenised Contracts are good for

1) DeFi composability (lending)
2) Compatibility with farming contracts (MasterChef)
3) The User should not need to know they hold the token or move it
4) LM Contract should have infinite approval to move when minted out to user

Implementation

LM Factory: Master Factory that creates out the LM contracts and holds master params.
LM Contract: The LM contract that wraps UniV3 NonFungiblePositionManager, 1 per pool.
Subsidy Farming Contract: The Contract that holds the subsidy tokens for farming. 1 Per pool.

Infra

1) Kyber Zap Aggregator to ZAP liquidity into the WEWESWAP LM Contracts, best-effort
2) LM Contract can handle unequal tokens, it will simply refund the left-overs
3) Krystal ALM Bot to re-balance liquidity, using internal LM contract logic

4) Uniswap V3 wrapped with fees swapped to USDC and deposited into LM Contract (by any bot)
5) LM Contracts that give ownership:
    a) Factory Contract (one child per pool)
    b) Tokenised
    c) Owner can pause ADD-REBALANCE
6) Farming contract
    a) One farming contract per subsidy coin
7) Frontend Interface
    a) Migrate
    b) Pool
    c) Swap
    d) Earn

Conclusion

WEWESwap is presented. It is a super-simple passive liquidity protocol that combines Concentrated Liquidity, Auto-rebalancing and a Liquidity Manager Contract that manages user positions. Users can "zap" in liquidity easily from any token. It accrues all the yield in USDC and lets the user claim it.