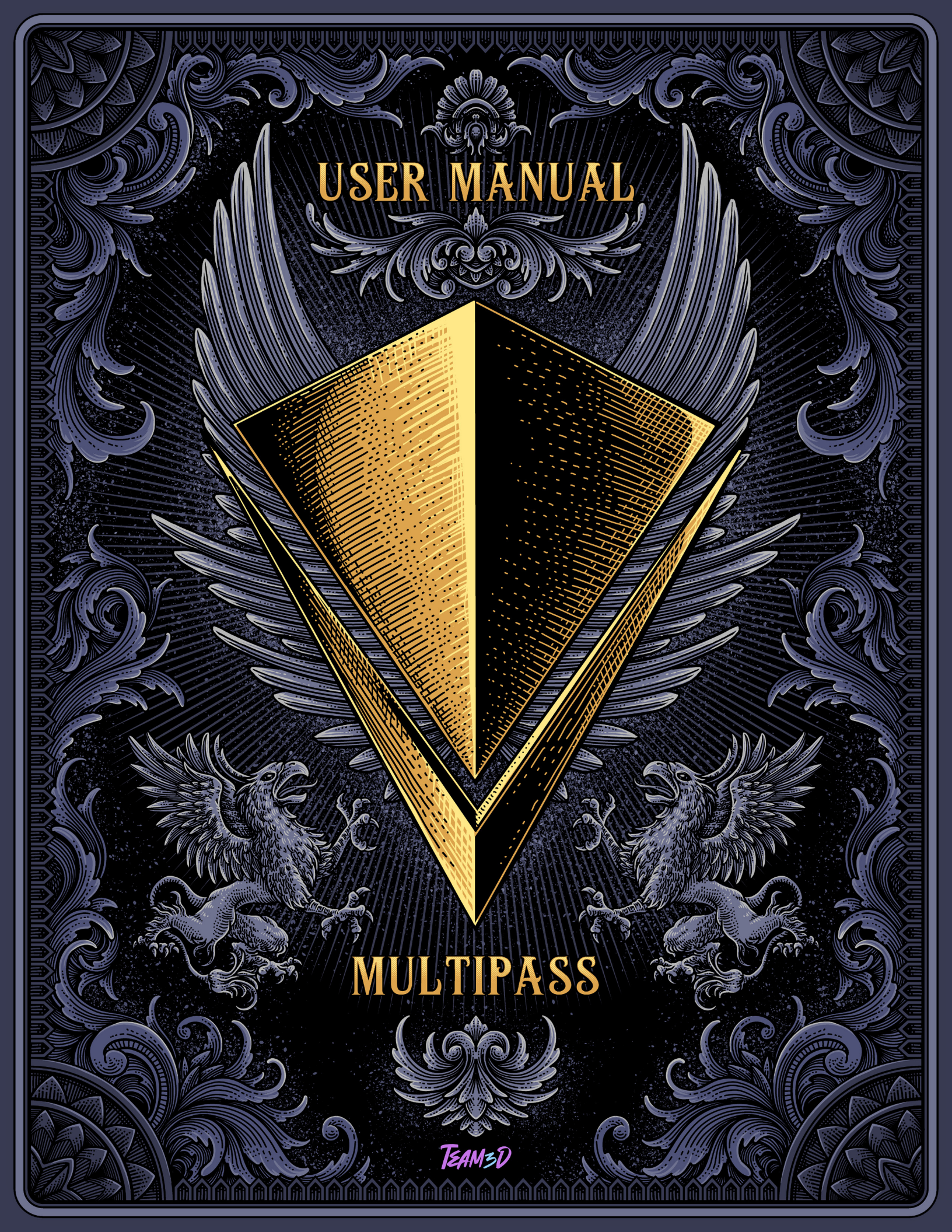
USER MANUAL



MULTIPASS

TEAM3D

TABLE OF CONTENTS

C:\team3d\multipass

OVERVIEW

> *IN THE WAKE OF THE ILLIQUID JPEG, THE LIQUID MP4 META CONQUERED*

The MultiPass is a liquid non-fungible token that functions as the Vidya ecosystem's native battle pass. It constitutes one of the first applications of a provably liquid blockchain-based membership pass with game theory baked into its software. MultiPass liquidity is functionally similar to any decentralized exchange, using Ethereum as a reserve currency and building upon reserves with every pass minted or merged and every level purchased.

MultiPass holders can redeem Reserved Ethereum in exchange for burning passes or levels through the MultiPass contract. It leverages economic incentives to limit the number of passes in circulation instead of a hardcoded and predefined maximum supply. With each MultiPass minted, the cost to mint increases, encouraging users to purchase passes from the open market when the cost of purchasing MultiPasses from it is lower than the cost of minting new passes into circulation through the contract. Though the mint price remains stable during the whitelist mint phase, the cost to mint is determined by the number of passes minted and increases ad infinitum. The cost of purchasing levels increases in the same fashion.

HODL Mode

Minting new passes becomes viable when the open market price meets or exceeds mint price, growing Ethereum reserves for existing holders who are incentivized either to hold their MultiPasses and continue collecting ETH over time or burn their passes to redeem collected ETH. This ensures that the MultiPass NFT is always liquid and reduces sell pressure on the open market.

Infinite Scarcity

As the cost to mint rises every time a new MultiPass is minted, eventually the only way to accumulate passes used for Vidya ecosystem access rights, accumulating ETH reserves, merging and leveling is to purchase them from existing holders on the open market until the floor price matches the mint price. This mechanic helps to support a growing Ethereum reserve and rising floor in tandem, depending on which provides a better acquisition price to users.

Progression

In contrast to the illiquid JPEG method of randomized rarity determination, the MultiPass utilizes a system of progression most often implemented in video games. Upon mint, every pass begins at level 1. As we'll outline later in this document, your rank is determined by your level against the highest level a MultiPass has achieved. Levels can be attained by being purchased or by merging multiple passes. As your level grows against the leader's level, you can achieve higher ranks; increasing the rarity of your pass, its access privileges, voting power and providing it with a more esteemed aesthetic. Should you wish to keep your MultiPass, you can burn access levels and redeem ETH instead of burning the entire pass.

TERMINOLOGY

Access Level

The level a MultiPass has been upgraded to. Exclusive ranks and incentives are granted to those who hold a higher level than other MultiPasses.

Burn

While not officially deleting the NFT token, burning the token will effectively destroy it. The typical method is to send the token to an address nobody possesses.

Count

The sum of all MultiPasses, minted after the whitelisted phase, plus the addition of 1 for every 5 levels a MultiPass has been upgraded. This variable impacts the price to mint MultiPasses and upgrade Access Levels.

Merge

The act of combining multiple MultiPasses into a single MultiPass. All but one MultiPass are burned. The new Access Level is the combination of all levels from the merged MultiPasses.

Mint

The process of converting a digital file into a crypto collectible. In this context, minting will send the ERC721 MultiPass token to your wallet.

MultiPass

A gamified ERC721 token granting special privileges in the Vidya Ecosystem.

Rank

A hierarchy of MultiPass Access Levels. Higher ranks are granted more exclusive privileges.

Redeem

The act of burning the MultiPass or Access Levels through the MultiPass contract in order to receive ETH.

Reserve

Ethereum collected in the reserve contract for redemption by MultiPass holders.

Voting Power

The weight at which your votes are counted in a DAO.

Whitelist

A list of wallets guaranteed access to an event or item before the public is given access.

PRICE

WHITELIST PRICE

0.02 ETH

WHITELIST PERIOD

Mint price of the MultiPass will remain constant during the whitelist phase.

STANDARD PERIOD

After the whitelist period has expired, mint price will increase with each MultiPass minted and for every 5 Access Levels a MultiPass is upgraded.

The ETH price for a given amount of MultiPasses minted is given by the following calculation:

Mint Price= Amount*(0.2ETH+(count*0.001ETH))+(Math.sum(Amount))*0.001ETH
count=sum of total MultiPasses minted+1 for every 5 levels a MultiPass has been upgraded

MATH.SUM RETURN TABLE

Amount	Math.sum Returns
1	0
2	1
3	3
4	6
5	10
6	15
7	21
8	28
9	36
10	45

EXAMPLE 1

Suppose there are 100 MultiPasses minted during the whitelist minting phase.

Count = 0.
After the whitelist mint phase ends, 1 additional MultiPass is minted.

Mint Price of MultiPass #101=1*0.2+0*0.001+Math.sum(1)*0.001=0.2 ETH

EXAMPLE 2

During the whitelist minting phase 100 are minted, after the whitelist minting phase 10 MultiPasses are minted and 3 MultiPasses have been upgraded to level 20.

Count = 22 = 10(MultiPasses) + 12(one for every 5 levels a MultiPass has been upgraded).
10 additional MultiPasses are minted by a user at once.

Mint Price of MultiPasses #111-#120=10*(0.2+(22*0.001))+(Math.sum(10)*0.001)
=
2.265ETH or 0.2265 ETH per m=MultiPass

ACCESS LEVELS

Each MultiPass starts at Access Level 1.

The MultiPass can be upgraded by purchasing additional levels. Similar to how there is no hardcoded mint cap on the MultiPass itself, there is no maximum level a MultiPass can reach. This is soft-capped by the increasing cost to purchase levels. Purchasing Access Levels can upgrade the MultiPass holder's rank and increase the amount of ETH they receive from burning the MultiPass in the redemption process. Your voting power in the VidyaDAO is determined by your Access Level.

Access Level Rank

Purchasing levels increases your rank in the Vidya ecosystem. Rank is given to each MultiPass holder based on the individual MultiPass Access Level compared to all other MultiPasses.

Tier 5

Plebeian

Tier 4

Gladiator

Tier 3

Centurion

Tier 2

Magister

Tier 1

Overlord

Buying Access Levels

The cost to purchase levels is equal to 5% of the current MultiPass mint price. This is summarized in the formula below.

$$\#\text{lvls bought} * \frac{\text{Mint Price(1)}}{20} = \text{Access Level Cost}$$

Purchased levels 1-4 do not impact the mint price calculation. Starting at level 5, and every 5 additional levels purchased, the mint cost of the MultiPass will increase. Consequently, the cost to purchase levels will also increase after the sale is processed.

Redemption and Burning

The owner of a MultiPass has two options to redeem the stored ETH value.

1. Users can choose to burn the MultiPass in its entirety to redeem its total stored ETH value.

2. Alternatively, the owner can burn access levels to redeem the stored ETH while keeping the MultiPass. If the owner chooses to burn the total level of the MultiPass, the pass is destroyed and the owner receives their ETH.

    a. If the owner burns levels but not the MultiPass, the MultiPass is downgraded to the appropriate Access Level. However, as this is only a downgrade and not total loss of the MultiPass, the MultiPass owner is charged a 2% fee at redemption. This partial redemption fee is left in the contract and increases the backing ETH for the remaining levels.

The calculation for Ethereum redeemed is:

$$\frac{\textbf{burnedLevels*balance}}{\textbf{totalLevels}} = \textbf{ETHtoReceive}$$

Merging MultiPasses

MultiPass owners can combine up to 10 MultiPasses in a process known as merging. As the name implies, merging MultiPasses is a means of fusing multiple access levels into a single MultiPass.

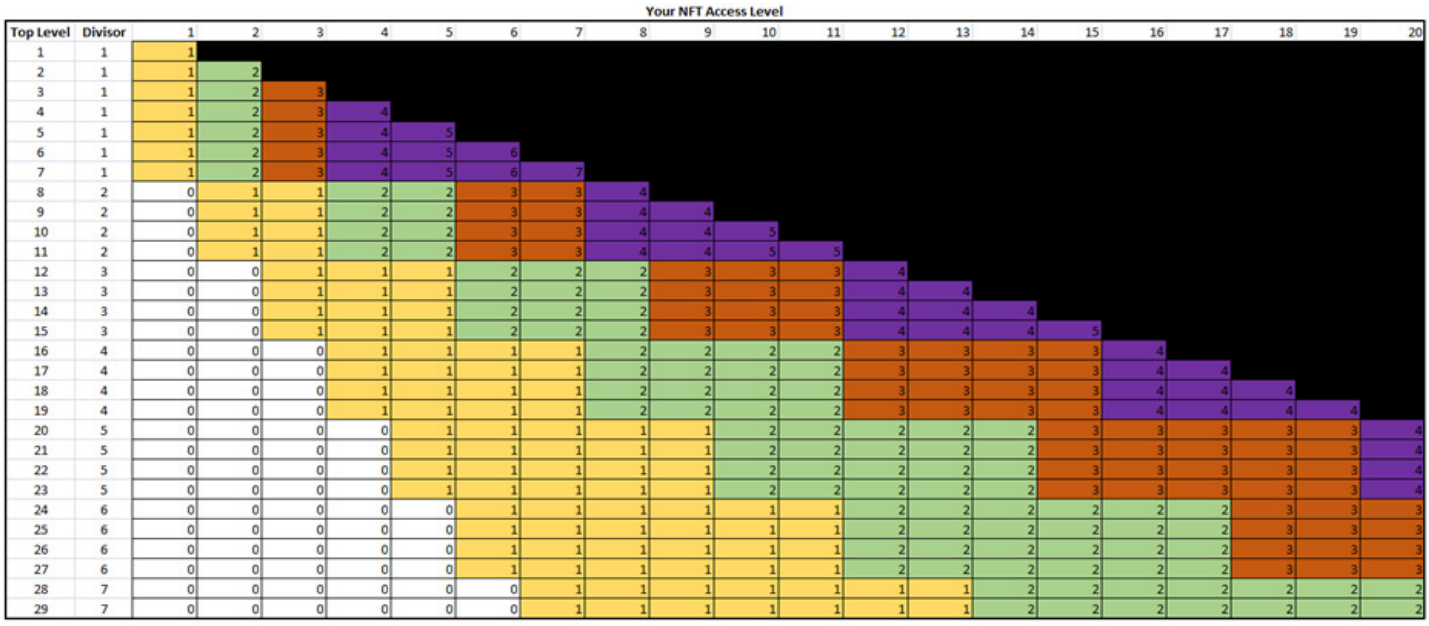
All levels across all merged MultiPasses are added into a single MultiPass from the pile.

The remaining MultiPasses are downgraded to level 0 and burned. As these MultiPasses were not used in the “Redeem Access Levels” process, the owner will not receive ETH for these burned cards. This should be obvious as those access levels were inherited into the single card received during the merging process. Thus nothing substantial was lost and the owner has consolidated their MultiPass position.

RANKS

Rank Determination

$$\frac{\text{floor(currentLevel)}}{\text{floor(TopLevel/4)}} = \text{Your Access Rank}$$



Top Level	Divisor	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
1	1	1																			
2	1	1	2																		
3	1	1	2	3																	
4	1	1	2	3	4																
5	1	1	2	3	4	5															
6	1	1	2	3	4	5	6														
7	1	1	2	3	4	5	6	7													
8	2	0	1	1	2	2	3	3	4												
9	2	0	1	1	2	2	3	3	4	4											
10	2	0	1	1	2	2	3	3	4	4	5										
11	2	0	1	1	2	2	3	3	4	4	5	5									
12	3	0	0	1	1	1	2	2	2	3	3	3	4								
13	3	0	0	1	1	1	2	2	2	3	3	3	4	4							
14	3	0	0	1	1	1	2	2	2	3	3	3	4	4	4						
15	3	0	0	1	1	1	2	2	2	3	3	3	4	4	4	5					
16	4	0	0	0	1	1	1	1	2	2	2	2	3	3	3	3	4				
17	4	0	0	0	1	1	1	1	2	2	2	2	3	3	3	3	4	4			
18	4	0	0	0	1	1	1	1	2	2	2	2	3	3	3	3	4	4	4		
19	4	0	0	0	1	1	1	1	2	2	2	2	3	3	3	3	4	4	4	4	
20	5	0	0	0	0	1	1	1	1	1	2	2	2	2	2	3	3	3	3	3	4
21	5	0	0	0	0	1	1	1	1	1	2	2	2	2	2	3	3	3	3	3	4
22	5	0	0	0	0	1	1	1	1	1	2	2	2	2	2	3	3	3	3	3	4
23	5	0	0	0	0	1	1	1	1	1	2	2	2	2	2	3	3	3	3	3	4
24	6	0	0	0	0	0	1	1	1	1	1	1	2	2	2	2	2	2	3	3	3
25	6	0	0	0	0	0	1	1	1	1	1	1	2	2	2	2	2	2	3	3	3
26	6	0	0	0	0	0	1	1	1	1	1	1	2	2	2	2	2	2	3	3	3
27	6	0	0	0	0	0	1	1	1	1	1	1	2	2	2	2	2	2	3	3	3
28	7	0	0	0	0	0	0	1	1	1	1	1	1	1	2	2	2	2	2	2	2
29	7	0	0	0	0	0	0	1	1	1	1	1	1	1	2	2	2	2	2	2	2

Plebeian | **Gladiator** | **Centurion** | **Magister** | **Overlord**

Your rank is not guaranteed by your level. The Top Level MultiPass sets the levels required for each rank.

Top Level	Your Level	Your Rank
16 (Overlord)	8	Centurion
20 (Overlord)	8	Gladiator
29 (Overlord)	8	Gladiator
29 (Overlord)	14	Centurion

Rank Privileges




	Plebeian	Gladiator	Centurion	Magister	Overlord
VidyaDAO Voting Rights	✓	✓	✓	✓	✓
Automated Whitelisting	✓	✓	✓	✓	✓
Exclusive NFT & Cosmetic Mints	✓	✓	✓	✓	✓
MultiPass Alpha Chat	✓	✓	✓	✓	✓
Token Reserve Access	✓	✓	✓	✓	✓
Token Launchpad Access		✓	✓	✓	✓
Closed BETA Game Access			✓	✓	✓
AimBots Comic Minting Rights				✓	✓
Token Reserve Access				✓	✓
Internal Application Test Access				✓	✓
Giveaways					✓

Plebeian
The general body of masses, "commoners". At least they're washed.

Gladiator
Gladiators are revered more highly than Plebeians, for what better honor is there than glory in battle?

Centurion
Respected for their leadership in the crucible of battle, Centurions rule over warriors and commoners. However, they often struggle with counting to one hundred.

Magister
Masters of both knowledge and battle, Magisters bow only to their overlords. It is believed they hold the answer to "wen".

Overlord
Highest among all Vidyans, the Overlords stand watch of their virtual kingdom. They command the respect of all subordinates; through violence, if necessary.

CONTRACT FUNCTIONS

uint256 accessLevel

- accessLevel – Access Level of MultiPass.

Constructor()

- Hardcoded values.

ETHmultiPurchase(uint8 _amount)

- _amount – Amount of MultiPasses the user has requested to mint.

topContender(uint256 _level)

- _level – A level check to see if this MultiPass is the top level to grant the rank of Overlord.

priceToken(uint256 _amount)

- _amount – Amount of MultiPasses the user has requested to mint.

rank(uint256 _tokenId)

- _tokenId – The ERC721 token ID of the MultiPass used.

spawn()

- Minting function

burnAccessLevels(uint256 tokenId, uint256 _burnLevels)

- tokenId – The ERC721 token ID of the MultiPass used.
- _burnLevels – The number of levels the user has requested to burn.

buyAccessLevels(uint256 _tokenId, uint256 _amount)

- tokenId – the ERC721 token ID of the MultiPass used.
- _amount – Amount of levels the user has requested to purchase.

mergePasses(uint256[] memory _tokenIDs)

- _tokenIDs – Array of tokenIDs the user has requested to merge.

ETHToReceive(uint256 levels)

- levels – Current level of MultiPass.

_beforeTokenTransfer(address from, address to, uint256 tokenId)

Overrides required by Solidity.

- from – Transfer from address.
- to – Transfer to address.
- tokenID – Token to be transferred.

---

supportsInterface(bytes4 interfaceId)

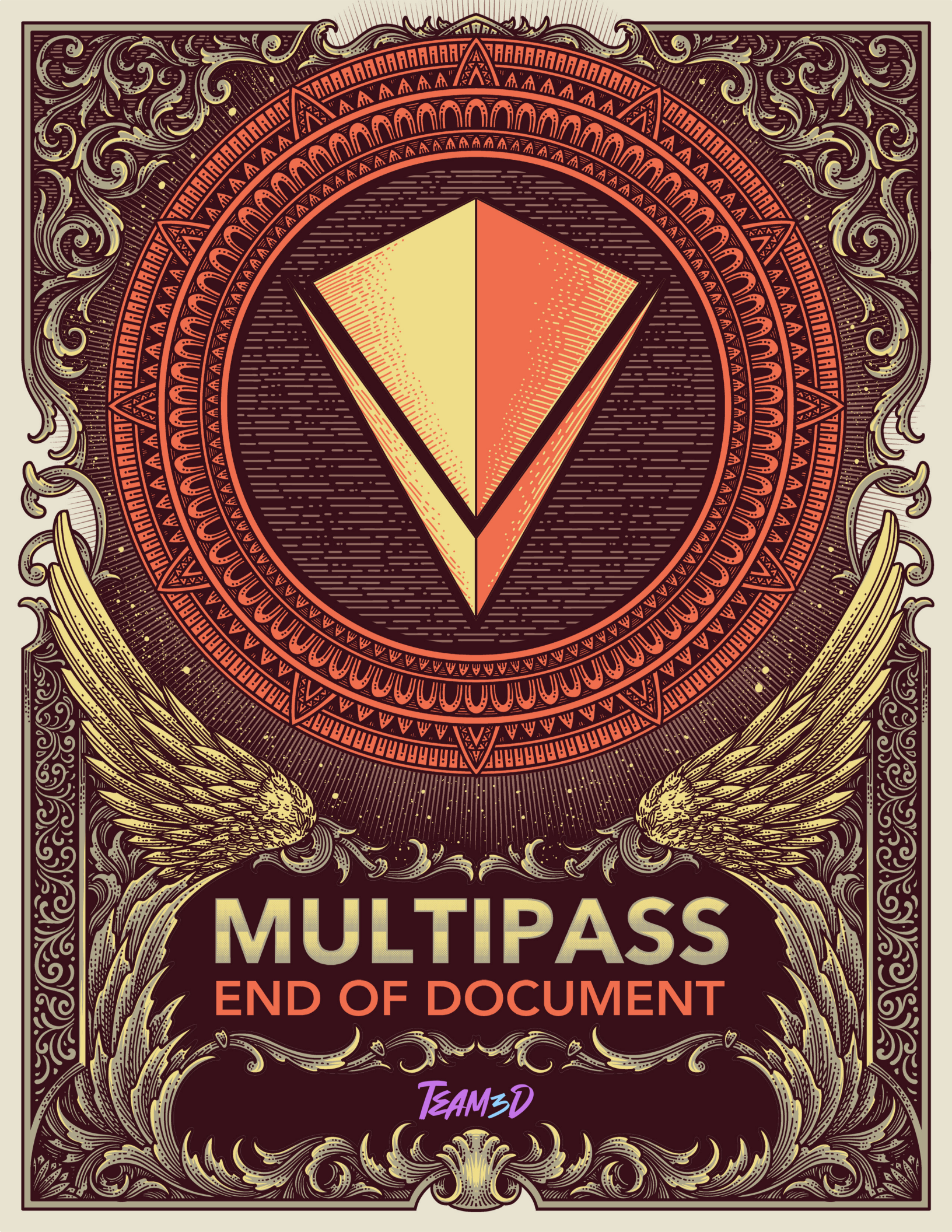
Overrides required by Solidity.

- interfaceId – Contract type.

---

addWhiteList(address[] memory _wl, uint8[] memory _amount)

- _wl – Array for MultiPass whitelist addresses.
- _amount – Array for maximum MultiPass token mint per address during the whitelist period.


MULTIPASS
END OF DOCUMENT
TEAM3D