Monet
Finance Detailed Version

What is DeFi?

DeFi is a set of smart contracts that operate independently on the blockchain. The smart contracts may or may not interact with other smart contracts or other blockchains.
The goal of DeFi is to increase the profitability of DeFi investors by automatically executing contracts in a way that seeks to maximize returns on our investment funds. DeFi is the beginning of finance, rapid innovations and concepts are created daily and the opportunities are enormous.

What is NFT?

NFT is the abbreviation of Non-Fungible Token. It usually refers to a token issued by developers according to the ERC1155 standard / protocol on the Ethereum platform. Its characteristics are that it's indivisible, irreplaceable and unique. In short, the token issued by the ERC1155 standard / protocol is called NFT.

What is Monet Finance?

Recently, we have seen extreme excitement about DeFi and people are jumping in with the mindset of FOMO. With the constant monetary incentives, cryptocurrency investors have become numb to buying, selling and mining, and have lost touch with understanding and exploring decentralization and the series of subtle designs within the blockchain. Price appreciation is the reason that you are here, but we hope you will have been a long term holder and appreciate the true value of blockchain.

Don't keep your eyes on price and ignore the value of DEFI and NFT.
That's why we're teaming up with multiple artists to link with NFT's design for a blockchain-perfect artistic extravaganza.Are you guys ready? Start to appreciate the aesthetically pure land of an impetuous market.

Monet uses a combination of DEFI and NFT. The unique mining model, combined with art, has created a variety of gamification methods and incentives systems.
We will be working with multiple artists to produce exquisite rare artwork and pay

homage to the classical artist Monet, who was an inspiration to many in the art industry.

The artwork may be from a particular classical artist or a book. All of which can be considered as collector's items. We hope that you will enjoy the aesthetics of the artwork, all of which are not only on the chain, but are also unique. After combining different styles and colors, new products will come out. When you have gathered enough collections, you will be promoted and rewarded.
We hope that the treasure you will get in the process is not just money, but amazing artworks for you to keep and explore on your own.
Monet Token MNT

Introduction

The MNT token is the governance and original token of the Monet ecosystem. Users holding MNT can participate in community proposals and decisions making, participate in LP mining, obtain artworks and continuously receive dividends.

Token Specifications

Name: Monet Finance Token
Token abbreviation: MNT
Network: Ethereum
Specification: ERC20
Accuracy: 18

Token Distribution

Total supply (100%): 21,000 MNT
Public offering (40%):   8,400 MNT
Uniswap liquidity pool (20%):   4,200 MNT
NFT bonus pool fund (30%):   6,300 MNT
Technology development and marketing expenses (5%): 1,050 MNT
Monet DAO (5%): 1,050 MNT

Uniswap initial liquidity

840,000 USDT and 4,200 MNT
Uniswap listing price: 1MNT = 200 USDT

How to get MNT?

Referral Awards/Community Events
Public offering/Uniswap purchase
Liquidity Pool Incentives

Introduction to L-MNT

L-MNT is an integral token on Monet Finance, which can be generated by LP mining and is the only fuel for extracting rare artworks.

In this game, the number of L-MNT mined every day is set to be 1 million, and the probability of synthesizing artwork 10 by L-MNT is unified to 0.1%. However, in order to encourage early ecosystem construction and community users, the probability of synthesizing artwork 10 will be increased to 0.8%, 0.4%, and 0.2% in the first, second, and third months.

Monet NFT Artwork

Introduction

Monet NFT is an exploration of GameFi. It's an innovative combination of NFT + DeFi, combining some of the most exciting innovations in the DeFi and crypto collections. New things in the form of games tend to be more appealing. The world of DeFi deserves more innovation to express the project's homage to Monet, and Monet NFT will be presented in a new and exciting way.

NFT Artwork Details

Ten artwork levels: 1 to 10 (different numbers represent different artwork)

Artwork/suit	Heart❤	Diamond♦	Club♠	Spade♣
Artwork 1	1	1	1	1
Artwork 2	4	4	4	4
Artwork 3	16	16	16	16
Artwork 4	64	64	64	64
Artwork 5	256	256	256	256
Artwork 6	1024	1024	1024	1024
Artwork 7	4096	4096	4096	4096
Artwork 8	16384	16384	16384	16384
Artwork 9	65536	65536	65536	65536
Artwork 10	262144	262144	262144	262144

Four suits: Heart, diamond, club, spade.
Face value: The value of Monet mining L-MNT casting NFT

Compositions and bonus: If you gather 4 pieces of Artwork 10 of different suits it can be combined into 1 piece of Artwork 9 (random suits), 4 pieces of Artwork 9 can be combined into 1 piece of Artwork 8, and so on... Each NFT Artwork 1 will earn you a quarter of the prize pool (i.e. 1575 MNT).

Acquisition methods for the Monet NFT artwork
Generated by mining for L-MNT exchange
NFT trading between players (can be traded on the official website or NFT trading platform)
Airdrops/incentives

Introduction to gameplay

1. Monet mining of $UNIV2-LP tokens
Add the same amount of USDT and MNT to the Uniswap pool

2. Enter the LP pool on the official website
-Unlock your wallet
-Approval of UNIV2-LP
-Enter the quantity of UNIV2-LP

The pool will earn you L-MNT every day according to the number of LP you bet. When you have earned enough L-MNT, you can redeem the L-MNT directly on the website for the required Monet Artwork 10.

When you have collected all 4 different decks of Artwork 10, you can move up, and so on.

When you successfully upgrade to NFT Artwork 1, the prize pool will distribute to you the Mega Millions jackpot.

Technical aspects

Liquidity mining

For the liquidity mining in Monet, we have used the SDK of Uniswap for mining, which can ensure the security of the liquidity pledged funds.

SDK used

Token

Token instance, which represents the ERC-20 token on a specific chain. The token instance information of ERC-20 can be obtained through the instance API.
Code example:

```
import { ChainId, Token } from '@uniswap/sdk' const token = new
Token(ChainId.MAINNET, '0xc0FFee0000000000000000000000000000000000', 18,
'HOT', 'Caffeine')
```

Pair

Pair represents the transaction pair instance in Uniswap. Through which the pair can be used to get a code example of the corresponding pair in Uniswap: the information includes the balance and address of the transaction pair.

```
import { ChainId, Token, TokenAmount, Pair } from '@uniswap/sdk' const HOT =
new Token(ChainId.MAINNET, '0xc0FFee0000000000000000000000000000000000',
18, 'HOT', 'Caffeine') const NOT = new Token(ChainId.MAINNET,
'0xDeCAf00000000000000000000000000000000000', 18, 'NOT', 'Caffeine') const
pair = new Pair(new TokenAmount(HOT, '2000000000000000000'), new
TokenAmount(NOT, '1000000000000000000'))
```

Route

The routing entity represents one or more ordered Uniswap transaction pairs with a complete path from the input token to the input token.
Code example:

```
import { ChainId, Token, TokenAmount, Pair, Route } from '@uniswap/sdk' onst HOT
= new Token(ChainId.MAINNET,
'0xc0FFee0000000000000000000000000000000000', 18, 'HOT', 'Caffeine') onst NOT
```

= new Token(ChainId.MAINNET, '0xDeCAf0000000000000000000000000000000000', 18, 'NOT', 'Caffeine') onst HOT_NOT = new Pair(new TokenAmount(HOT, '2000000000000000000'), new TokenAmount(NOT, '1000000000000000000')) onst route = new Route([HOT_NOT], NOT)

Fetcher

An example method for getting data on a chain.
Code example for getting DAI token information:
import { ChainId, Token, Fetcher } from '@uniswap/sdk' const chainId = ChainId.MAINNETconst tokenAddress = '0x6B175474E89094C44Da98b954EedeAC495271d0F' const DAI: Token = await Fetcher.fetchTokenData(chainId, tokenAddress)

Trade

The SDK does not execute or send transactions on your behalf. It provides utility classes and functions that make it easy to calculate the data needed for secure interaction with Uniswap. Almost everything you need to do for a secure transaction with Uniswap is provided by the trade entity. Use Trade to send transactions to the route, assuming we want to swap as many DAIs as possible with 1 WETH.

Code example:

import { ChainId, Token, WETH, Fetcher, Trade, Route, TokenAmount, TradeType } from '@uniswap/sdk' const DAI = new Token(ChainId.MAINNET, '0x6B175474E89094C44Da98b954EedeAC495271d0F', 18) // note that you may want/need to handle this async code differently, // for example if top-level await is not an option const pair = await Fetcher.fetchPairData(DAI, WETH[DAI.chainId]) const route = new Route([pair], WETH[DAI.chainId]) const amountIn = '1000000000000000000' // 1 WETH const trade = new Trade(route, new TokenAmount(WETH[DAI.chainId], amountIn), TradeType.EXACT_INPUT)

Contract Structure

Artwork Contract

The contract that we use is ERC1155 protocol. This protocol is the most powerful token standard ever created in the history of Ethereum, and the protocol is already in its final version and does not require any modifications to it.

ERC1155 has the following characteristics:

One smart contract can manage an unlimited number of tokens.
While the ERC-20 and ERC-721 tokens require the deployment of a new smart contract for each new token "class", the core concept behind the ERC-1155 is a smart contract that can manage an unlimited number of tokens.
Think of it as a vending machine with a variety of sodas, juices, and even snacks. Customers use a secure interface to interact with the machine (insert coins, press the button), and the machine will distribute the goods of their choice. In the same way, an ERC-1155 game contract can contain a variety of items, from weapons and armor to health potions, magic scrolls, etc.
Each item can be "fungible" with multiple copies available. Fungible tokens are used for divisible currencies (most erc-20 tokens), and they are also useful for stackable items that do not need to be distinguished.
One kind of token is called the Non-Fungible Token (NFT). This structure allows each token to have its own unique parameters. A pet dragon in video games can be an NFT, and has its own unique name, power level and full game history.

Batch transmission

With ERC-1155, multiple tokens can be sent in a single transaction - this significantly saves on GAS costs and avoids the need to wait for each block in a single transmission.
Projects using ERC-1155 can also build atomic exchanges with the same basic design, allowing users to exchange another token with an absolutely secure token without involving any middlemen.
Ronan Sandford, a Sandbox developer, was able to generate more than 1,500 tokens in a block while retaining real ownership through their ERC-1155 implementation.

Phillippe Castonguay, a developer at Horizon Games, uses a technique called Balance Packing (storing 16 lower resolution tokens in a single ID), saving 80 to 90% of the gas cost compared with regular transfers.
Phillippe was also able to achieve over 155 asset transfer speeds per second via ERC-1155 tokens!

Economic data

The main feature of the ERC-1155 multi token standard is to instantiate multiple tokens in a smart contract. This means that "creating" a new token type can be as simple as calling a function that adds a new ID to the available token pool.
In contrast, creating a new token type using the ERC-20 and 721 standards means editing the code and deploying a brand-new contract to an address on the Ethereum blockchain. Deploying a contract requires a significant amount of gas (ETH), because the cost of storing data on a globally shared blockchain is naturally very high. Most ERC-20 contracts are based exactly on the same code, thus there is no need for modification. This means that in every node of Ethereum, more than 25,000 nodes will be messed up with redundant code that will stay there forever!
Each time a new token is started, the operational cost of each new ERC-20 or ERC-721 contract address needs to be notified. 100 new tokens means 100 new contracts to look at - because every wallet and every piece of software must know that the token exists.

So, while the first two criteria apply to narrow use cases - whether it's a single currency or a single category of NFT – the vast majority of creators will benefit from ERC-1155, because it will be able to dynamically create new tokens and represent more than one "type" without repeating the same contract.
Strict rules will make our tokens more reliable.

One of the incredible features of the ERC-1155 token standard is the strict set of rules that will need to be followed to achieve this.

The ERC-1155 token is the first token type that can perform deterministic smart contract functions by simply sending a token to an address.

By sending a token to a DEX exchange address, the exchange can immediately return another token to the sender's address. Similarly, a blockchain game can perform game functions immediately upon receiving an ERC-1155 token from a user.

The tokens can be packaged, converted, produced, or hosted without accessing ABI or interacting directly with a smart contract.

This feature was originally proposed in ERC-223, but the standard did not catch on. ERC-721 also implements " safeTransferFrom ", but there is a fatal problem, that is, all transfers are not strictly required, which means that users may lose their tokens permanently, and contract authors cannot rely on guaranteed execution when sending tokens.
The decision to strictly implement this function in ERC-1155 means that a reliable network of smart contracts and tokens can now begin to build on this powerful capability.

Dao Token Contract

With the consideration of the characteristics of our governance token in mind, the ERC20 protocol was chosen to meet Monet's demand for a homogeneous token.

L-MNT Token Contract

The homogeneous ERC20 protocol token L-MNT, which is mainly responsible for extracting 10 artworks of various suits.

Exchange Contract

This contract provides access to the ERC1155 protocol and the ERC20 protocol assets for swapping, using as escrow sales, and can be invoked by the user to complete the asset realization of ERC1155.

Mining Pool Contract

This contract provides liquidity mining for MNT / USDT trading pairs. Mining is conducted according to the user's weight ratio in the trading pair, and L-MNT output is conducted according to the production block speed of ETH, which is about once in 10 seconds.

Art Pledge Mining Contract

Users can pledge 6-2 artworks for L-MNT mining and output according to mortgage weight.

Governance Contract

The contract has the function of proposing and voting. All users who hold MNT are able to decide the future developments of the protocol. When there is more than 60% of support of the proposal, the proposal will be executed, and the contract will be created.

Routing Contract

This contract is responsible for the main logic of Monet, including lottery, composition, exchange, etc. The contract works together with other contracts, and can be upgraded separately without affecting other contracts, so as to ensure the compatibility of the project in the later stage.

Ethereum random number generator

Currently, what's in the market uses the target contract random algorithm to collide with the winning probability in the cheating contract, and the hit will call the target contract to obtain the reward.

Solution

Built-in random seeds.
Restrict contract calls involving random interfaces in the game

```
library SafeRandom {
     function revise(uint256 x) internal pure returns (uint256 z) {
        require((z = x + 1) >= x, 'ds-random-revise-overflow'); }

    function rand(uint256 x, uint256 y) internal view returns (uint256) {
        return uint256(keccak256(abi.encodePacked(block.difficulty, now, x))) % y; }

     function seedRand(uint256 x, uint256 y, uint256 z) internal view returns (uint256)
{  return uint256(keccak256(abi.encodePacked(block.difficulty, now, x, y))) % z;}

contract Seed {
    using SafeRandom for uint256;
    uint256 internal seed = 0;
    modifier reviseSeed() {
        seed = seed.revise();  }

    modifier validateSender(address account){
        uint256 size;
        assembly { size := extcodesize(account) }
        require( size == 0,"account is contract");}

contract MonetGame is Seed {

    function fun() external validateSender(msg.sender) reviseSeed {

        uint256[] memory nums = new uint256[](10);
        nums[0] = seed.rand(4);
        for (uint256 i = 1;i < 10;i++){
       nums[i] = seed.seedRand(4,1);     }

        emit random(msg.sender,nums);  }

    event random(address indexed owner,uint256[]  nums);
```