Mystery box

Project Overview

Catgirl Mystery Box provides users with the opportunity to buy Catgirl NFT with different characters and rarity of NFT by using Chain Link V2 to get the random factor. Support users payment by BNB and Catgirl token.

1.Functional Requirement

1.1 Roles

- **UPGRADER_ROLE:** Catgirl contracts are UUPSUpgradeable contracts, so dev has a role to upgrade the contract.
- **SETTER_ROLE:** able to change some settings of contracts.
- **FACTORY_ROLE**: not being used at the moment.

1.2 Features

Catgirl Mystery Box has the following features:

- Buy mystery boxes by BNB and Catgirl ERC20 Token.

- Swap BNB to Catgirl token through Pancake Router.
- Distribute the sales profit to the appropriate wallets.

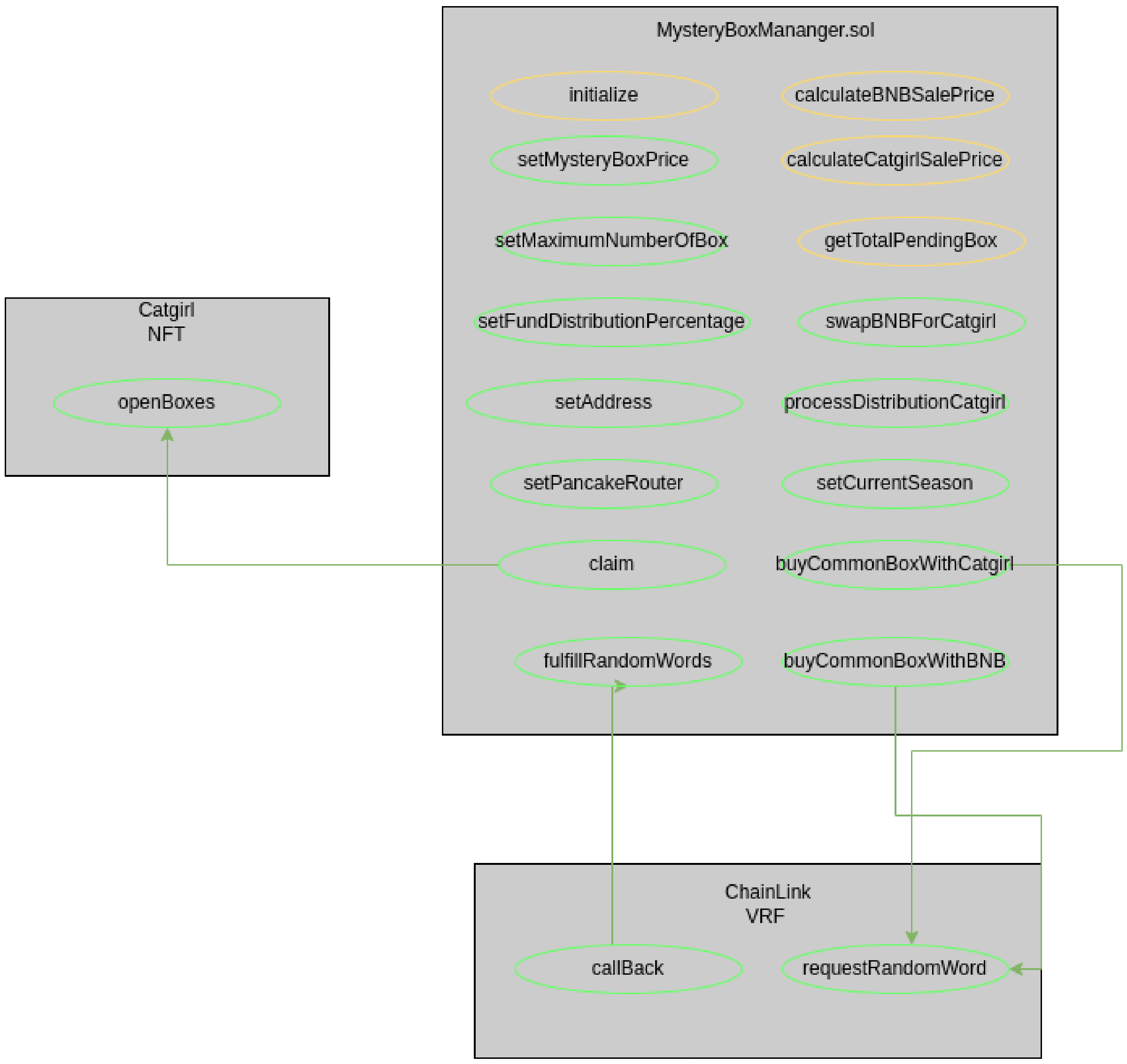
1.3 Use cases

- Buying mystery boxes and claiming Catgirl NFTs with random factors from Chain Link to mint random NFT. We use the technique of pending claims due to the nature of the fallback from chainlink VRF so we can allow the user to pay their minting gas fee directly. This due to the fact that the cost of minting our NFT is relatively low, and some users would want to open 100+ boxes at a time.
- Support swap BNB to Catgirl token through Pancake Router.
- Distribute the sales profit to the appropriate wallets. The structure is as follows:
    - 20% of the sales will be allocated to the platform fee
    - 75% of the sales will be allocated to the reserved farming pool. This farming pool wallet is controlled by the team, and further will be distributed to the actual NFT farming pool contract.
    - 5% will be burned

2. Technical Requirement

2.1 Architecture Overview

2.2 Contract information



2.2.1 MysteryBoxMananger.sol

2.2.1.1 Assets

MysteryBoxMananger contract contains 2 structs:

- **PendingBoxs** : This object contains information about the signature of the user
    - **uint8 numberOfBox:** number of pending boxes users has purchased.
    - **uint256 randomNumber:** random number get from Chain Link callback..
    - **uint64 tokenType:** define type of token user bought mystery box is BNB or Catgirl.(0 is BNB, 1 is Catgirl)

- **RequestBuyBoxDatas:** each time a user purchases mystery boxes, the contract generates a request to Chainlink to get the random factor and save the request to this object.
  - **address user**: address of user purchased mystery box.
  - **uint8 numberOfBox**: number of mystery boxes user bought.
  - **uint64 tokenType:** type of token user bought mystery box (0 is BNB, 1 is Catgirl)

Besides the mentioned structs, the following entities are present in the contract:

- **bytes32 public constant UPGRADER_ROLE:** hashed string name for Upgrade role
- **bytes32 public constant SETTER_ROLE:** hashed string name for Setter role
- **bytes32 public constant FACTORY_ROLE:** not being used at the moment
- **uint32 public currentSeason:** current season of NFT released by year, increasing each time a new season comes.
- **uint256 public mysteryBoxFiatPrice:** price of a mystery box by USD.
- **int128 developmentPercentage:** percentage of distribution token to the development team.
- **int128 burnPercentage:** percentage of distribution token needed to burn each time distribution.
- **address public farmingAddress:** address of farming contract where users could use Catgirl NFT to farm Catgirl ERC20 token.
- **address public developmentAddress:** address of development team receives token distribution.
- **IPancakeRouter02 pancakeRouter:** router of Pancake contract.
- **IPriceOracle priceOracle:** oracle contract which determines price of BNB and Catgirl ERC20 token.
- **address public constant deadAddress:** dead address where burned tokens go to.
- **uint256 previousSeed:** old random mechanism
- **IERC20Upgradeable uCatgirlToken:** Catgirl ERC20 token.
- **VRFCoordinatorV2Interface COORDINATOR:** Address of coordinator Chain Link v2 where Mystery Box contract sends the request to get a random factor.
- **uint64 sSubscriptionId:** subscription ID registered with Chain Link service.

- **bytes32 sKeyHash:** hashed key comes along with subscription provided by Chainlink.
- **uint32 callbackGasLimit:** gas limit when Chain Link callback to contract to send random word.
- **uint16 requestConfirmations:** number of request confirmation of Chain Link.
- **uint32 numWords:** number of random words requested to Chain Link.
- **address vrfCoordinator:** address of VRF coordinator.
- **mapping(address => PendingBox[]) public pendingBoxs:** a public mapping containing a list of pending boxes requested of an user.
- **mapping(uint256 => RequestBuyBoxData) public requestVrfs:** a mapping containing a list of requests to chainlink.
- **uint8 public maximumNumberOfBox:** maximum number of box users can pending before buying new boxes, that would limit number of box could open when claim in 1 transaction is: **maximumNumberOfBox + maximumNumberOfBoxAtATime - 1** with **maximumNumberOfBoxAtATime** configured in Catgirl NFT contract. The reason we have the **maximumNumberOfBox** variable is to prevent the user from spending an insane amount of gas fee at a time when claiming new boxes.
  We should check **maximumNumberOfBox** at both function **buyCommonBoxWithBNB** and **buyCommonBoxWithCatgirl** but there is a mistake at **buyCommonBoxWithBNB** that causes misunderstanding.

2.2.1.2 Modifier

Mystery Box contract does not have any modifier.

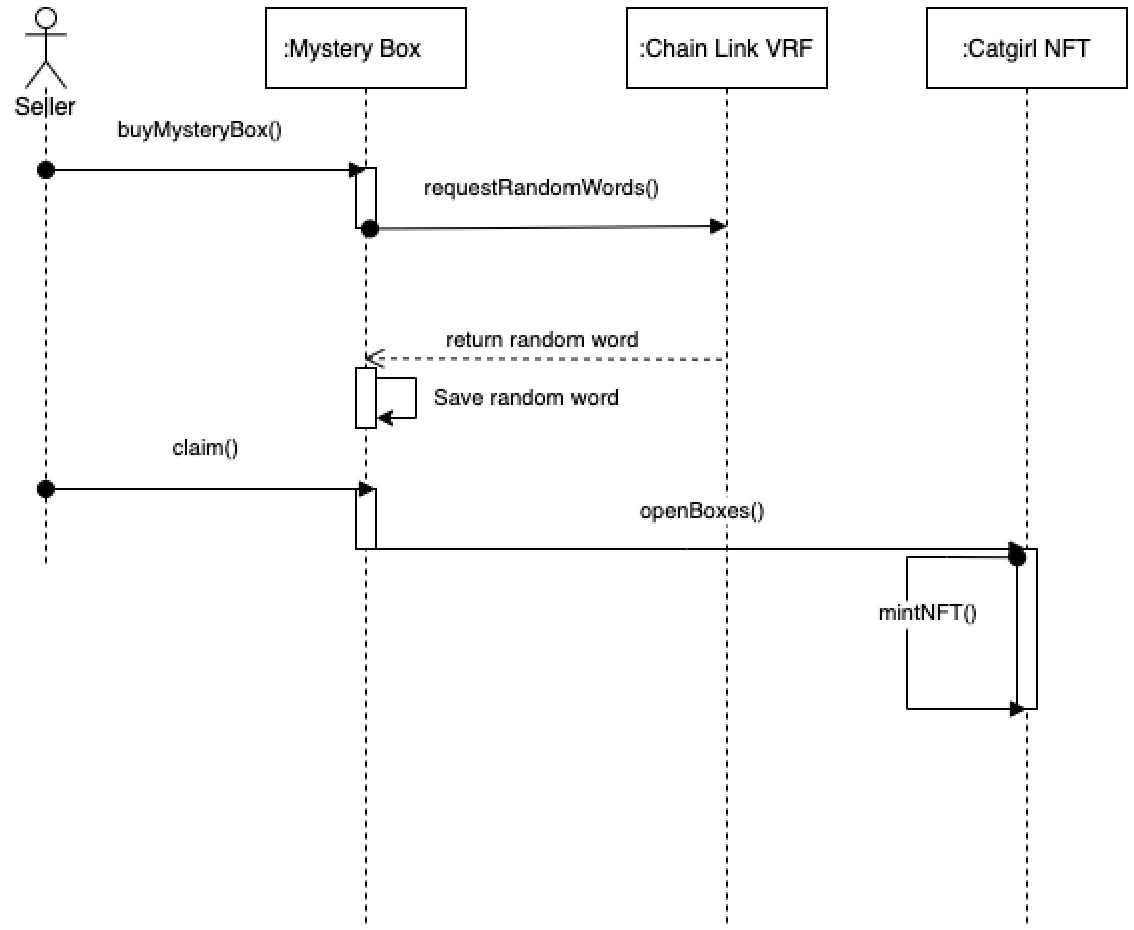
2.2.1.3 Functions

- **initialize(IERC20Upgradeable _catgirlAddress, ICatgirlNFT _catgirlNFT,IPriceOracle _priceOracle)**: initialised function of contract.
- **__MysteryBoxManager_init_unchained(IERC20Upgradeable _catgirlAddress,ICatgirlNFT _catgirlNFT,IPriceOracle _priceOracle)**: Initialise default value of some variable.
- **_authorizeUpgrade(address newImplementation)**: Update new implementation when contract needs to upgrade.
- **supportsInterface(bytes4 interfaceId)**: Returns true if this contract implements the interface defined by interfaceId.
- **setMysteryBoxPrice(uint256 _newPrice)**: Setting new value of **mysteryBoxFiatPrice.**

- **setMaximumNumberOfBox(uint8 _newMaxNumberOfBox)**: Setting new value of **maximumNumberOfBox.**
- **setFundDistributionPercentage(uint256 _development, uint256 _burn)**: setting new value for percentage of distribution.
- **setAddress(address _farmingAddress, address _developmentAddress)**: setting new value of farming address and development address.
- **setPancakeRouter(IPancakeRouter02 _pancake)**: setting new value of Pancake router.
- **setCatgirlTokenAddress(address _catgirlTokenAddress)**: setting new value of **catgirlTokenAddress**
- **setCatgirlNFTAddress(address _catgirlNFTAddress)**: setting new value of **catgirlNFTAddress**
- **setCurrentSeason(uint32 _season):** setting new value of season, should check if the new season is valid.
- **emergencyWithdraw():** support withdraw if something broken.(this function should be removed)
- **processDistributionCatgirl()**: distribute Catgirl token as the percentage defined.
- **processDistributionBNB()**: distribute BNB token as the percentage defined.
- **swapBNBForCatgirl(uint256 amount)**: Call to Pancake router to support user swap from BNB to Catgirl.
- **pause()**: temporarily stop the contract.
- **calculateBNBSalePrice()**: Calculate number of BNB needed to buy a Mystery box, get the BNB/USD rating from oracle contract.
- **calculateCatgirlSalePrice()**: Calculate number of Catgirl needed to buy a Mystery box, get the Catgirl/USD rating from oracle contract.
- **buyCommonBoxWithCatgirl(uint8 _numberOfBoxes)**: buy mystery boxes by Catgirl.Contract would call Chainlink to request a random factor to attach with these boxes.
- **buyCommonBoxWithBNB(uint8 _numberOfBoxes)**: buy mystery boxes by BNB.Contract would call Chainlink to request a random factor to attach with these boxes.
- **setConfigVfr(uint64 _ssubcriptionId, bytes32 _skeyhash,uint32 _callbackGasLimit, uint16 _requestConfirmations,uint32 _numWords, address _vrfCoordinator)**: update some Chain Link config if needed.
- **claim()**: claim all pending boxes.
- **requestRandomNumber(address _user, uint8 _numberOfBox, uint64 _tokenType):** call to VRF contract to request random word.
- **fulfillRandomWords(uint256 requestId_,uint256[] memory randomWords_):** callback function, should be called from VRF contract.

- **getTotalPendingBox(address user) public view returns (uint256)**: count total of pending box of an user.

Flow Diagram:



Seller
:Mystery Box
:Chain Link VRF
:Catgirl NFT
buyMysteryBox()
requestRandomWords()
return random word
Save random word
claim()
openBoxes()
mintNFT()