FOMO3D Smart Contract Documentation

Description

The FOMO3D Smart Contract is a decentralized game of financial incentives, where participants buy keys that add time to a countdown clock. The game ends when the clock reaches zero, and the last key buyer wins a pot of funds.

Contract Variables

The contract maintains several key state variables:

- `lastBuyer`: The address of the last participant who bought a key.
- `lastBuyTimestamp`: The timestamp of the last key purchase. This is also used as the countdown clock.
- `pot`: The current pot of funds to be distributed to the winner.
- `totalKeysSold`: The total number of keys sold in the current round.
- `roundCount`: The number of rounds completed.

It also maintains several mappings and arrays to track the distribution of rewards:

- `keyHolders`: A mapping from an address to the number of keys it holds.
- `keyHolderAddresses`: An array of addresses of all key holders.
- `accumulatedPlatformShare`, `accumulatedHolderPrizeShare`, `accumulatedInviterShares`, `accumulatedNewPlayerShares`: Variables and mappings that track the accumulated shares of various stakeholders.

Functions

buyKeys

The `buyKeys` function allows a participant to buy a certain number of keys. The participant specifies an inviter address (can be address 0), and sends a payment equal to the total price of the keys. The function calculates the shares of the payment to various stakeholders, updates the game state, and emits a `KeyPurchased` event.

_distributePrize

The `_distributePrize` function distributes the prizes when the game ends. It is called internally when the countdown clock reaches zero. The function transfers the pot to the last buyer, distributes accumulated shares to various stakeholders, resets the game state, and emits a `RoundEnded` event.

Events

KeyPurchased

The `KeyPurchased` event is emitted when a participant buys keys. It includes the buyer's address, the payment amount, the number of keys bought, and the inviter's address.

RoundEnded

The `RoundEnded` event is emitted when a round of the game ends. It includes the round number.

Security Measures

The contract inherits from OpenZeppelin's `ReentrancyGuard` and `Pausable` contracts to protect against reentrancy attacks and allow the contract owner to pause and unpause the contract. The `buyKeys` function uses the `nonReentrant` and `whenNotPaused` modifiers to enforce these protections.

Important Notes

The contract uses a logarithmic pricing model for keys, which means the price increases with each key purchase. The price calculation involves a user-defined `log10` function which approximates the base-10 logarithm. The contract also uses a random lottery mechanism to select a winner from all key holders every time keys are purchased.

This contract should be thoroughly audited before use in production, as it involves financial transactions and potential rewards for participants.

FOMO3D 智能合约文档

描述

FOMO3D 智能合约是一种去中心化的金融激励游戏，参与者购买的钥匙可以增加倒计时时钟的时间。当时钟归零时，游戏结束，最后一个购买钥匙的参与者将赢得一笔奖金。

合约变量

合约维护了几个关键的状态变量：

- `lastBuyer`：最后一个购买钥匙的参与者的地址。
- `lastBuyTimestamp`：最后一次购买钥匙的时间戳。这也被用作倒计时时钟。
- `pot`：当前要分配给赢家的奖金。
- `totalKeysSold`：当前轮次中售出的钥匙总数。
- `roundCount`：完成的轮次数。

它还维护了几个映射和数组以跟踪奖励的分配：

- `keyHolders`：从地址到其持有的钥匙数量的映射。
- `keyHolderAddresses`：所有钥匙持有者的地址数组。
- `accumulatedPlatformShare`，`accumulatedHolderPrizeShare`，`accumulatedInviterShares`，`accumulatedNewPlayerShares`：跟踪各种利益相关者累积份额的变量和映射。

功能

buyKeys

buyKeys 函数允许参与者购买一定数量的钥匙。参与者指定一个邀请人地址（可以是地址 0），并发送等于钥匙总价的付款。该函数计算

付款的各种利益相关者的份额，更新游戏状态，并发出一个 KeyPurchased 事件。

_distributePrize

_distributePrize 函数在游戏结束时分发奖金。当倒计时时钟到达零时，内部调用该函数。该函数将奖金池转给最后的购买者，将累积的份额分配给各种利益相关者，重置游戏状态，并发出一个 RoundEnded 事件。

事件

KeyPurchased

当参与者购买钥匙时，会发出 KeyPurchased 事件。它包括买家的地址、付款金额、购买的钥匙数量和邀请人的地址。

RoundEnded

当游戏的一轮结束时，会发出 RoundEnded 事件。它包括轮次数。

安全措施

该合约继承自 OpenZeppelin 的 ReentrancyGuard 和 Pausable 合约，以防止重入攻击，并允许合约所有者暂停和解除暂停合约。buyKeys 函数使用 `whenNotPaused 和 nonReentrant 修饰符来确保合约没有被暂停并防止重入。

计算和随机性

合约使用 log2 和 log10 函数来计算钥匙的价格，这两个函数实现了在链上的对数运算。它还使用区块链的 timestamp 和 difficulty 作为随机种子来选择抽奖的赢家。

重要提示

请注意，尽管这个合约包含了许多安全措施，但是还是有可能存在安全风险，比如以太坊的矿工可以稍微操纵区块的时间戳，这可能会影

响到抽奖的随机性。在正式使用之前，建议对这个合约进行彻底的安全审计。

此外，合约中的金额和比例都以最小单位（wei）进行处理和计算，以避免浮点数的精度问题。在与合约交互时，也需要将所有的金额转换为最小单位。

最后，buyKeys 函数的处理方式是，如果在购买钥匙的过程中游戏结束，那么首先会分发奖金，然后才会处理钥匙的购买。这意味着购买者可能会在一个新的游戏轮次中购买钥匙，而不是他们原来打算购买的那一轮。为了避免这种情况，可以考虑在调用 buyKeys 之前先检查游戏是否结束，或者在 buyKeys 函数中添加更多的逻辑来处理这种情况。