Introduction

1. Purpose:

The purpose of this document is to outline the design of a classic gold miner game that will interact with the Qitmeer EVM testnet. The scores earned by players will be saved in a smart contract and awards will be given to the top 5 players every 60 minutes based on their scores.

1. Scope:

This document outlines the design of the unity frontend, backend, and smart contract of the game. It also covers the use of Metamask as the wallet for the players and the process of depositing Meers to play the game and use them for awards.

1. Overview:

The game will consist of a Unity-based frontend, a backend, and the smart contract end. Players will deposit a certain amount of Meers to play the game, and the scores they earn will be saved in the smart contract.

Technical Requirements

1. Unity front-end:
   1. The front-end will be developed using Unity game engine
   2. The game will feature a gold miner game with the following specifications:
      1. Miner and hook
      2. Underground objects spawns according to the random number from the smart contract
      3. Duration of level is 120 seconds
      4. Props like strength potion, dynamite, Stone collection
   3. The front-end will interact with the smart contract to store player scores on-chain.
2. Smart contract end
   1. The smart contract will be developed using the Solidity and will be deployed on the Qitmeer EVM testnet.
   2. The smart contract will have the following functions:
      1. startGame: This function will allow players to deposit a certain amount of Meers to play the game
      2. endGame: This function will be called at the end of the game to store the score.
      3. RNG: generate random number for creating level
      4. Awardize: This function will be called by the back-end every 60 minutes and will distribute awards to the top 5 players based on their scores earned in the period.
3. Back-end
   1. The back-end will be responsible for calling the awardize function in the smart contract every 60 minutes to finish one period and start new period.
   2. Back-end will broadcast events to clients.
   3. It will be developed using Node.js
4. Wallet:
   1. The game will use Metamask as the wallet for players.
   2. Players will be required to install Metamask and connect it to the Qitmeer EVM testnet to play the game.
   3. The game will interact with Metamask to deposit and withdraw Meers.

User Interface Design

1. The Unity front-end of the game will feature a classic gold miner game with a menu screen that displays the options to connect wallet, start the game, view scores, and exit the game.
2. The menu screen will also display the current balance of the player's Metamask wallet.
3. The game screen will display the miner game and the current score of the player.
4. The view scores screen will display history and current state of scores and awards saved in the smart contract.

Data Structures

The smart contract will store the following information for each player:

* 1. Wallet address
  2. Accumulated score
  3. Accumulated award
  4. Points earned in current period

It will also store the information of the awards history.

Algorithms

1. Miner and hook:
   1. The hook’s speed depends on the size of the object it is holding.
   2. Strength potions will increase speed of hook twice for a short while.
   3. Dynamite instantly removes the object attached to hook.
2. Underground objects Spawning:
   1. The underground objects will spawn according to the random number from the smart contract.
   2. There are several underground object types: large gold block, small gold nugget, Large stone, small stone, mole, mine.
3. Game Over:
   1. The game will be over when time is up.
   2. The player's score will be calculated based on the sum of values of objects the miner mined and will be saved in the smart contract.

Security Considerations

1. The smart contract will be thoroughly tested and audited before deployment to prevent any potential security vulnerabilities.
2. The back-end server will save owner’s private key or seed phrase to call awardize function in smart contract.
3. The game will use Metamask as the wallet, ensuring that player funds are secure.

Deployment

1. The Unity front-end will be compiled and packaged for deployment on two platforms: Windows desktop standalone and Webgl.
2. The back-end server will be deployed on a suitable platform, such as Vercel or Render.
3. The smart contract will be deployed on the Qitmeer EVM testnet.

Maintenance and Upgrades

The game will be regularly maintained and updated to fix bugs and improve performance.

The smart contract will be updated as required to reflect changes in the Qitmeer EVM testnet or to add new features.

The smart contract will be deployed on Qitmeer EVM mainnet after testnet trial.

Conclusion

This software design document outlines the design of a classic gold miner game that will interact with the Qitmeer EVM testnet. The game will consist of a Unity front-end, a back-end, and a smart contract end, and will use Metamask as the wallet for players. The scores earned by players will be saved in the smart contract, and awards will be given to the top 5 players every 30 minutes based on their scores.The game will be developed and deployed within 1 month and will be open-sourced under the MIT license.