Introduction

1. Purpose:

The purpose of this document is to outline the design of a simple snake 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 30 minutes based on their scores.

1. Scope:

This document outlines the design of the unity frontend, simple 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 simple backend that will call the awardize function in the smart contract every 30 minutes, 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 2D snake game with the following specifications:
      1. A snake that grows as it eats food
      2. Food that respawns in random locations
      3. A game over state when the snake collides with the wall or itself
   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. Awardize: This function will be called by the back-end every 39 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 a simple server that will be responsible for calling the awardize function in the smart contract every 30 minutes.
   2. 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 simple 2D snake 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 snake, food, walls, 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. Snake Movement:
   1. The snake's movement will be controlled by keyboard with the arrow keys.
   2. The snake will move in the direction indicated by the player and its length will increase as it eats food.
   3. The snake’s speed will increase as the time goes
2. Food Spawning:
   1. The food will respawn in random locations on the game screen.
   2. The snake's length will increase every time it eats food.
3. Game Over:
   1. The game will be over when the snake collides with the walls or itself.
   2. The player's score will be calculated based on the length of the snake 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 simple snake game that will interact with the Qitmeer EVM testnet. The game will consist of a Unity front-end, simple 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.