Tomo Sheep Fight

Version 1.0, published 20190926.
Author: Do Trung Kien

Feature

- P2P game
- Build by Unity, with Photon network & Netheum core.
- Blockchain tech with Tomochain.

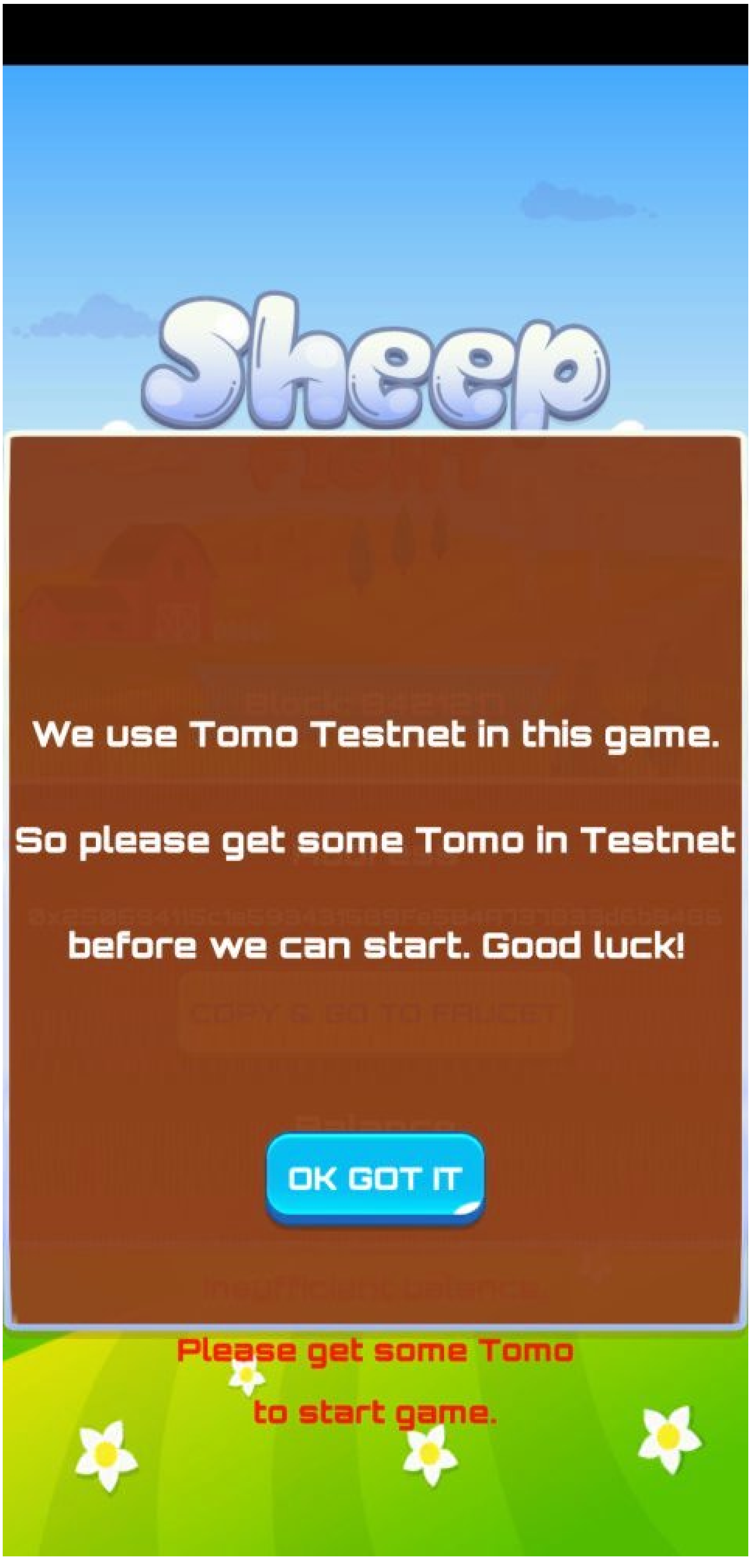
For end user: Play game

Playing this game is so easy.

Firstly, just grab the apk file, install it into your android phone.

Because of PoC purpose only, we decided to use Tomochain testnet instead of mainnet.

Open the app, the first thing you see will be like this:



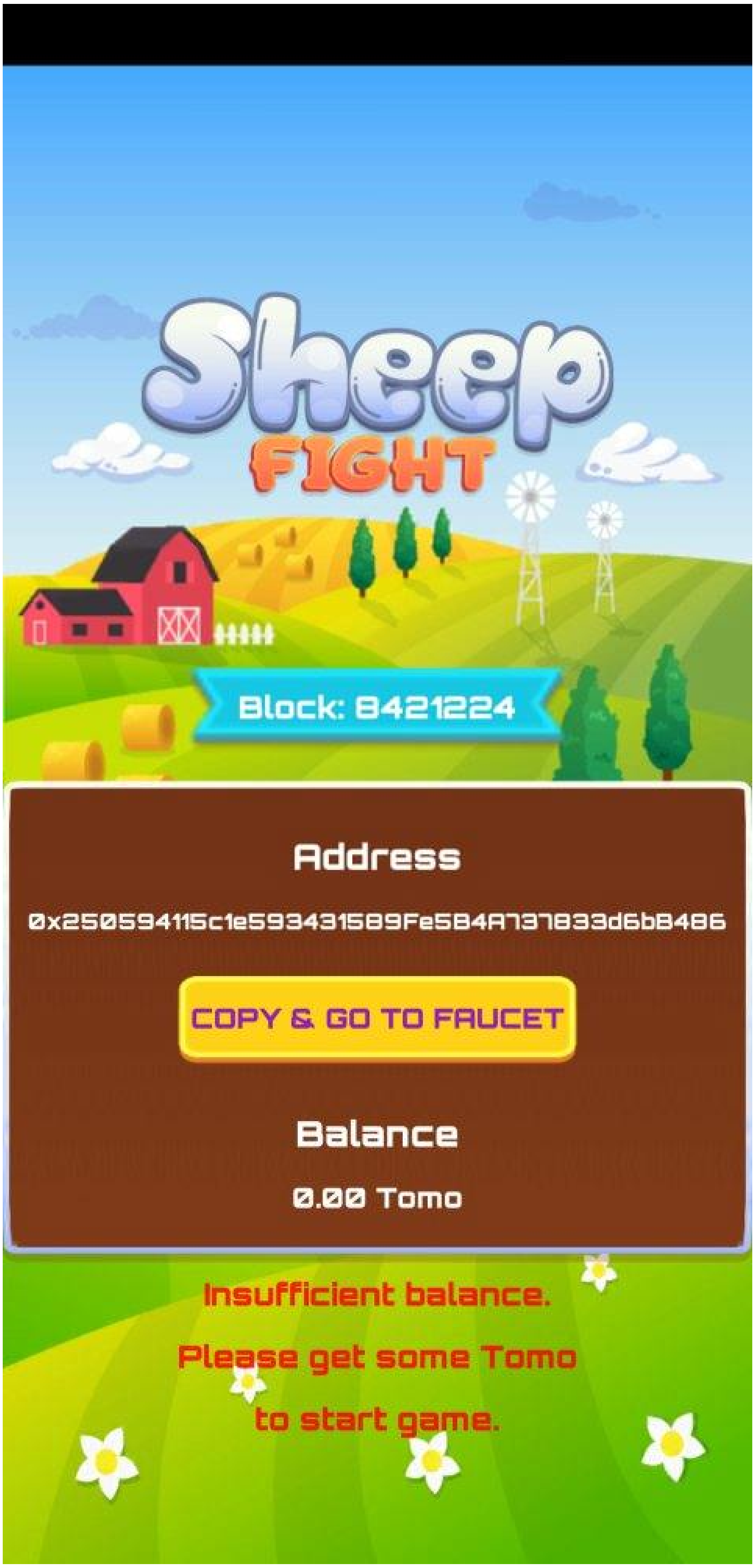
We use Tomo Testnet in this game.

So please get some Tomo in Testnet

before we can start. Good luck!

Please get some Tomo

to start game.



Block: 8421224

Address

0x250594115c1e593431589Fe584A737833d6b8486

Balance

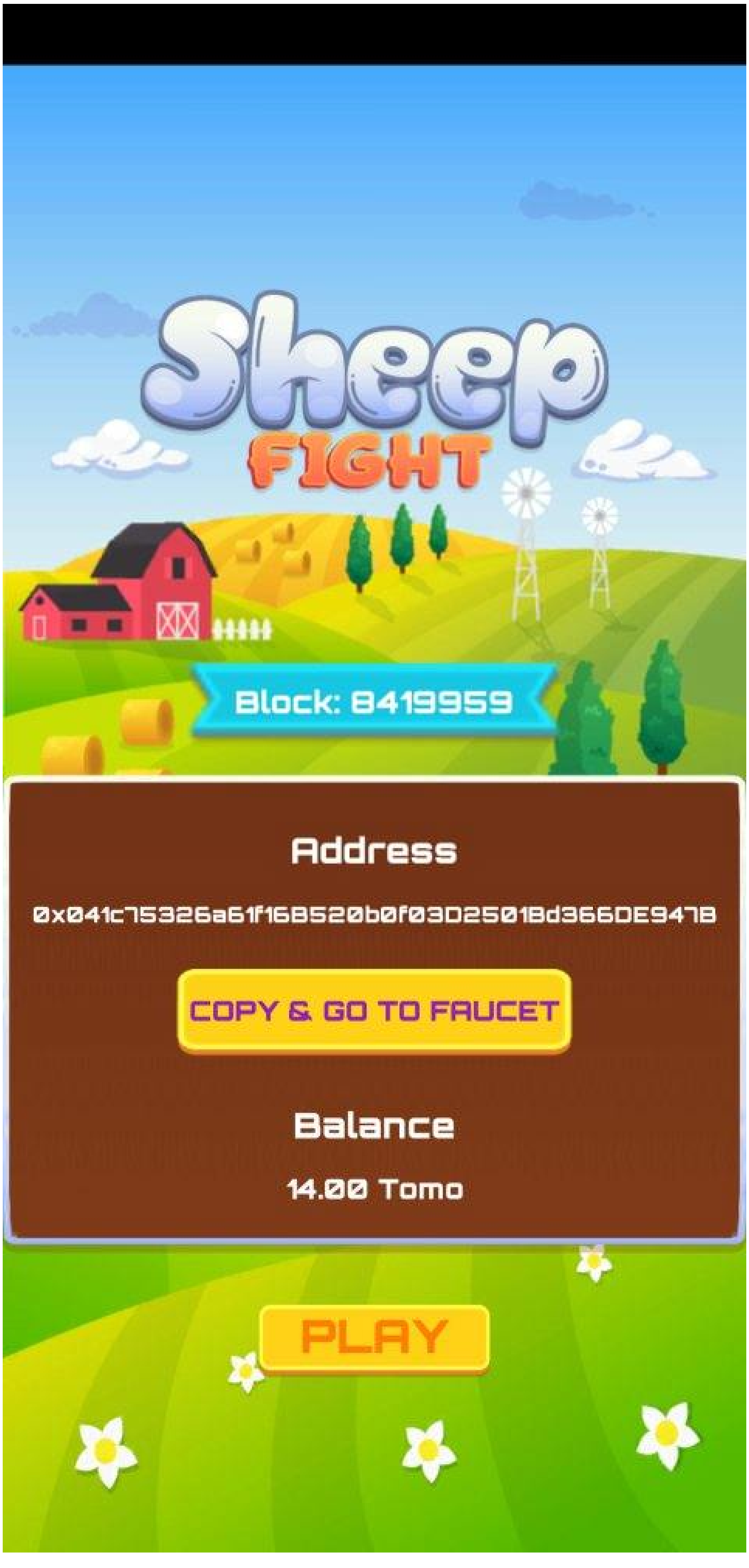
0.00 Tomo

Insufficient balance.
Please get some Tomo
to start game.

At first, we create an account for your, so you don’t need to worry about the private key or any key at all.

By clicking `COPY & GO TO FAUCET` we will be redirected to Tomo Faucet. Get 15 Tomo just enough to get started. Of course you can get as much as you want.

Now you have some Tomo in your account and are eligible to play this game. Hit `PLAY` and enjoy new game if there is a ready player in that, or wait for another one join your game. The bet to join game is 1 Tomo, the winner will get all!



Block: 8419959

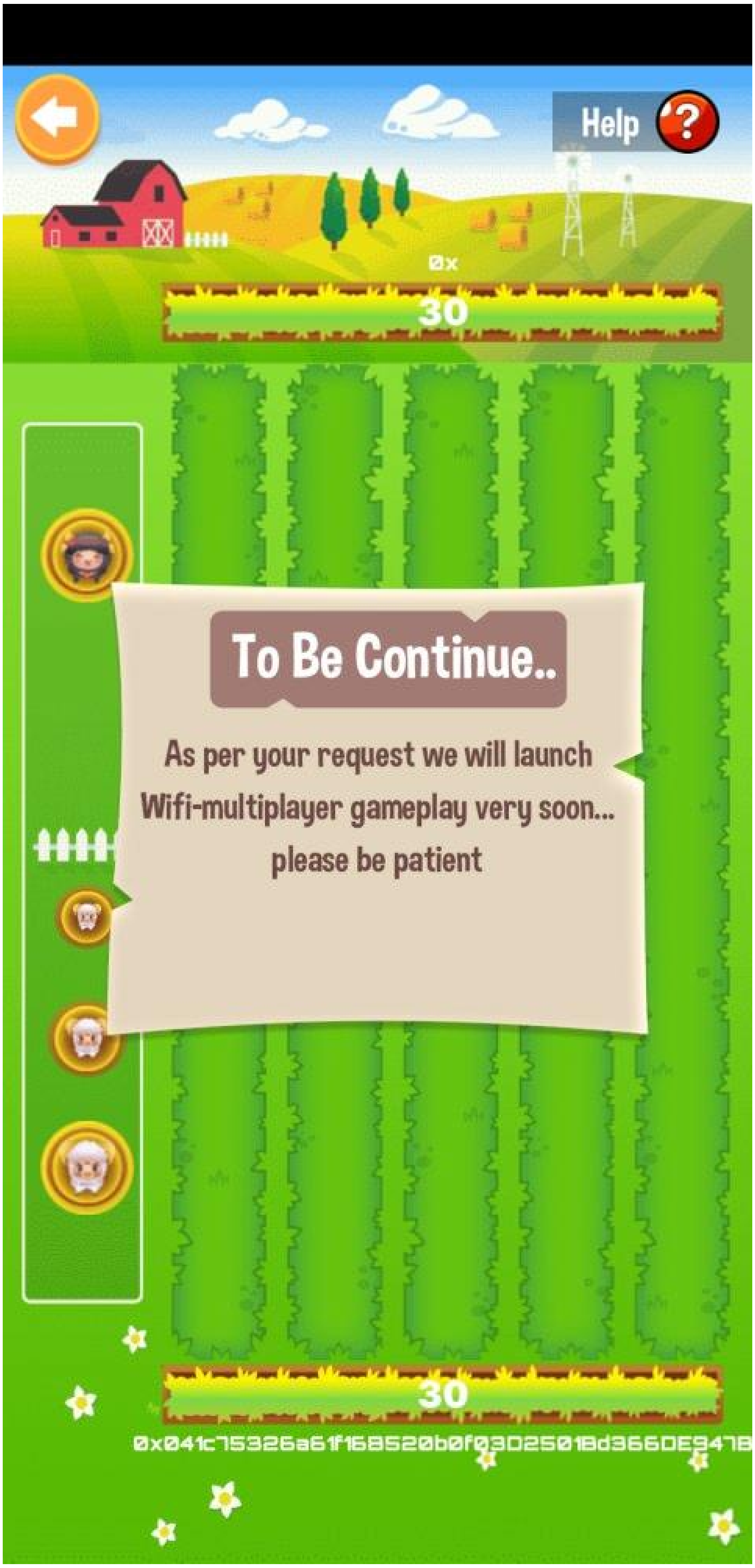
Address

0x041c75326a61f16B520b0f03D25018d366DE947B

Balance

14.00 Tomo

Wait for other play join:



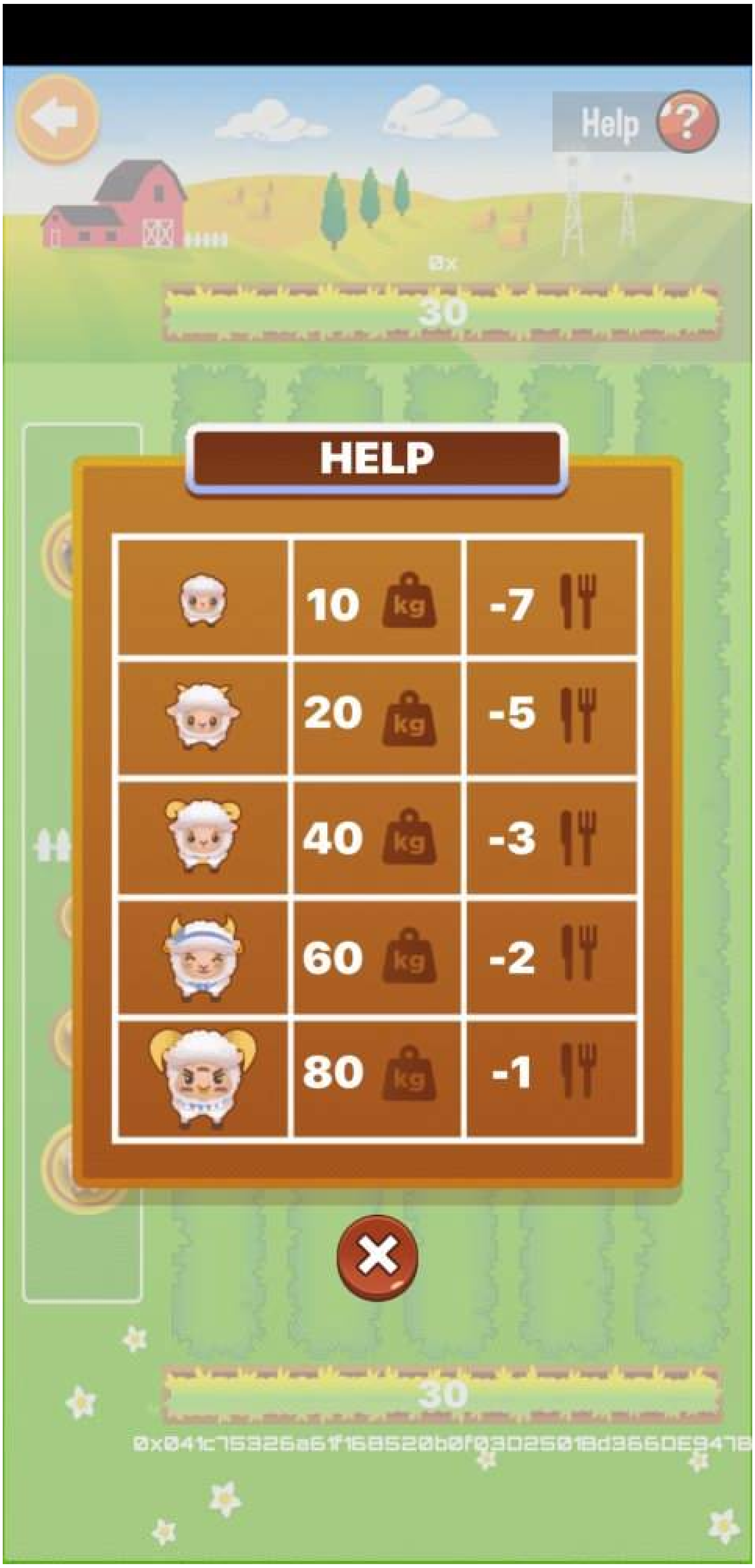
To Be Continue..

As per your request we will launch Wifi-multiplayer gameplay very soon... please be patient

In the game, each sheep has its own weight and point if it can break the enemy barrier. The heavier, stronger, but also the less point gain.

So you should design you strategy very carefully to get the highest score in the fastest time.

Press `HELP` if you need more detail:

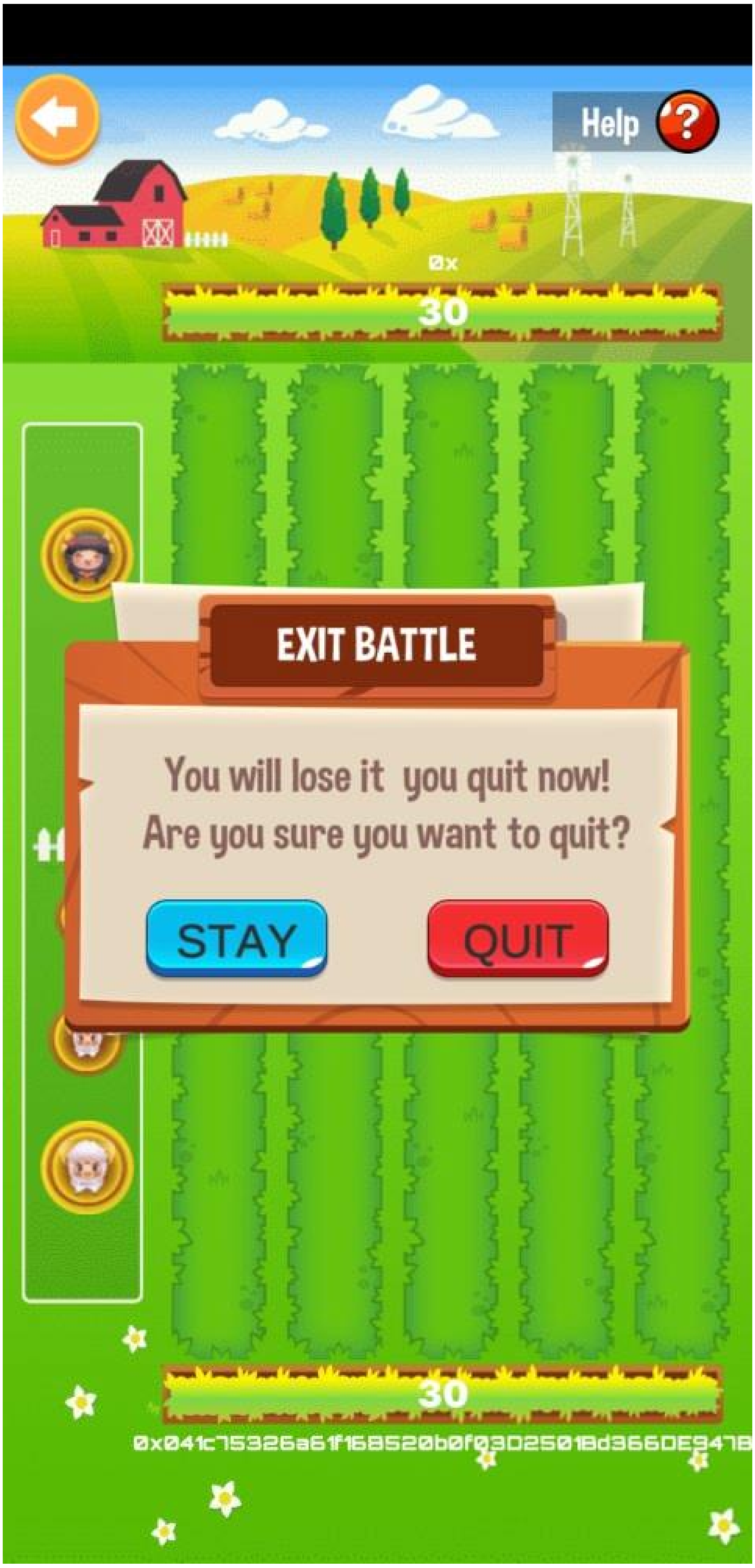


HELP

	10 kg	-7
	20 kg	-5
	40 kg	-3
	60 kg	-2
	80 kg	-1

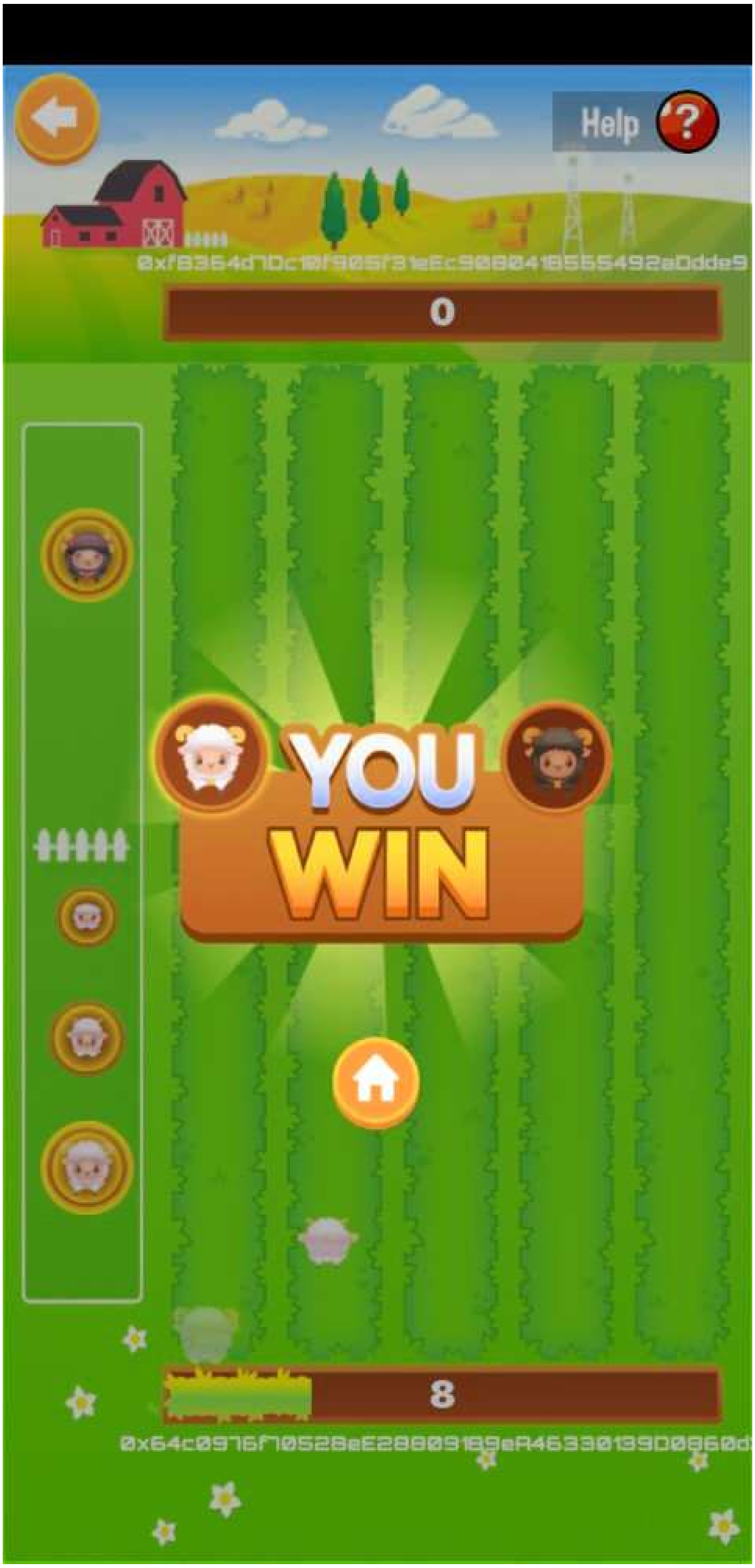
0x041c75326a61f168520b0f03025018d3660E9478

You can leave the game anytime you want. But you should not, because your deposit in game will be transfer to the opponent. You will lose 1 Tomo!!

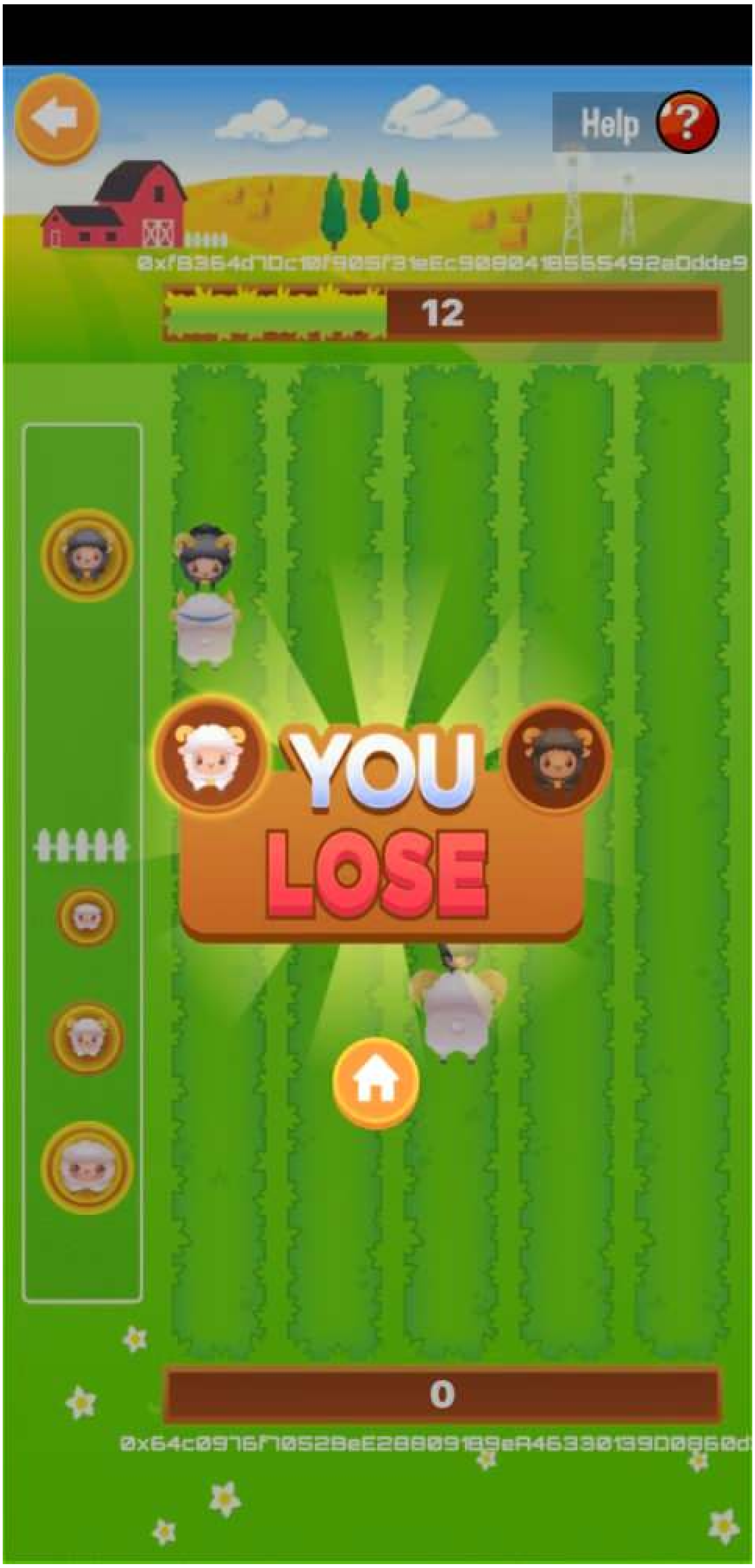


You will lose it you quit now!
Are you sure you want to quit?

At end game, winner will receive all Tomo bet in game.


Help
0xf8364d70c10f905f31eEc90804185555492a0dde9
0
YOU
WIN
8
0x64c0976f70528eE28809189eA46330139D0860d

and loser lost all


Help
?
0xf8364d70c18f905f31eEc9090418555492a0dde9
12
YOU
LOSE
0
0x64c0976f70528eE28809189eA463301390085 0d

For developer

There are several main components in this project:

- Contracts - created by Truffle
- Game - created by Unity
- Realtime multiplayer server - with Photon network
- Tomochain testnet connector - with Nethereum core

Contract

We created contract with `Truffle 5` and Solidity `0.5.0`. Currently, for fast prototype, we skip all cheat verifycation steps, and focus on play logic only.

```
pragma solidity 0.5.0;

contract SheepFight {
    struct Game {
        string id;
        address payable leftPlayer;
        address payable rightPlayer;
        uint wonID; // 0 NA, 1 left win, 2 rightwin
    }

    Game[] public games;

    mapping (address => bool) public isPlaying;
    mapping (address => uint) public playerToGame;

    uint public betValue = 1 ether;

    constructor () public {
        games.push(Game("123456", address(0), address(0), 0));
    }

    function searchGame(string memory gameID)
        internal
        view
        returns (uint)
    {
        for (uint i = 0; i < games.length; i++)
        {
            if (compareStringsbyBytes(games[i].id, gameID)) return i;
        }
        return 0;
    }

    function compareStringsbyBytes(string memory s1, string memory  s2)
        public
        pure
        returns(bool)
    {
        return keccak256(abi.encodePacked(s1)) ==
keccak256(abi.encodePacked(s2));
    }

    function play(string calldata gameID)
        external
        payable
    {
        require(msg.value >= betValue, "must bet 1 value");
        require(!isPlaying[msg.sender], "player must not be in game");

        isPlaying[msg.sender] = true;
        uint gameIdx = searchGame(gameID);
        if (gameIdx == 0) {
            createGame(gameID);
        } else {
            joinGame(gameIdx);
        }
        if (msg.value > betValue) msg.sender.transfer(msg.value - betValue);
    }

    function createGame(string memory gameID)
        internal
        returns (uint)
    {
        Game memory newGame = Game(gameID, msg.sender, address(0), 0);
        uint latestGame = games.push(newGame) - 1;
        playerToGame[msg.sender] = latestGame;
        return latestGame;
    }

    function joinGame(uint gameIdx)
        internal
    {
        Game storage game = games[gameIdx];
        game.rightPlayer = msg.sender;
        playerToGame[msg.sender] = gameIdx;
    }


    function winGame()
        external
    {
        require(isPlaying[msg.sender], "player must be in game");
        uint gameIdx = playerToGame[msg.sender];
        require(gameIdx != 0, "not exist game");
        Game storage game = games[gameIdx];
        require(game.wonID == 0, "game was ended");
        game.wonID = (msg.sender == game.leftPlayer) ? 1 : 2;
        reward(msg.sender);
        resetPlayer();
    }

    function loseGame()
        external
    {
        if (isPlaying[msg.sender]) {
            isPlaying[msg.sender] = false;
        }
        uint gameIdx = playerToGame[msg.sender];
        if (gameIdx != 0) {
            playerToGame[msg.sender] = 0;
        }
    }

    function forceEndGame()
        external
    {
        uint gameIdx = playerToGame[msg.sender];
        if (gameIdx == 0) return;
        Game storage game = games[gameIdx];
        if (game.leftPlayer == msg.sender && game.rightPlayer != address(0))
reward(game.rightPlayer);
        if (game.rightPlayer == msg.sender && game.leftPlayer != address(0))
reward(game.leftPlayer);
        resetPlayer();
    }

    function reward(address payable to)
        public
        payable
    {
        require(address(this).balance >= 2*betValue, "insufficient balance");
        to.transfer(2*betValue);
    }

    function resetPlayer()
        internal
    {
        uint gameIdx = playerToGame[msg.sender];
        if (gameIdx == 0) return;
        isPlaying[msg.sender] = false;
        playerToGame[msg.sender] = 0;
        Game storage game = games[gameIdx];
        address leftPlayer = game.leftPlayer;
        if (leftPlayer != address(0)) {
            isPlaying[leftPlayer] = false;
            playerToGame[leftPlayer] = 0;
        }
        address rightPlayer = game.rightPlayer;
        if (rightPlayer != address(0)) {
            isPlaying[rightPlayer] = false;
            playerToGame[rightPlayer] = 0;
        }
    }

    function () external payable {}
}
```

In next step, we will record every step that user player sent to both the contract and photon server. Thus, we can verify the match result and prevent player from cheating.

Unity game

There 2 scene in this game:

1. Lobby

At lobby we init the Photon network, Sheep Fight smart contract, set up player and match making.

Please check the Unity project `Scenes/Lobby` Scene and `Scripts/Lobby/` for more detail.

2. Game

In game, there a `Game Controller` to control the sheep spawn, for both local player (base on player click) and remote player (base on Photon RPC call).

Please check the Unity project `Scenes/Game` and `Scripts/Game/` for more detail.

Through all, we keep two singleton in this scene and in all game, `GameManager` to keep all game information and `SheepContract` to interact with the smart contract.

In next step, sound & music also gonna be added.

Photon network

- We use `PUN RPC` to communicate betweeen clients, and `MonobehaviorPUNCallbacks` for handle every network event.

Smart contract interaction

- We use Nethereum to implement Web3 & contract instance in game. All transaction will be done asynchronously.

Known issues

Due to short duration of development in this hackathon, we've faced many troubles, and some even still exist in the latest build.

As the consequence of many asynchronous actions between Game, Photon network and Blockchain, we still can not control those 100% and it lead to some unwanted delay effects in this game. We are trying to solve those.

Next Plan

In next versions, we are going to fix all bugs and publish the game not only Android version but also iOS and other platforms version, too. And of course, support multichain like Tomochain mainnet, ETH mainnet, Ropsten, Loom, Rinkeby...

Enjoy gaming.

Do Trung Kien
trungkien.keio@gmail.com