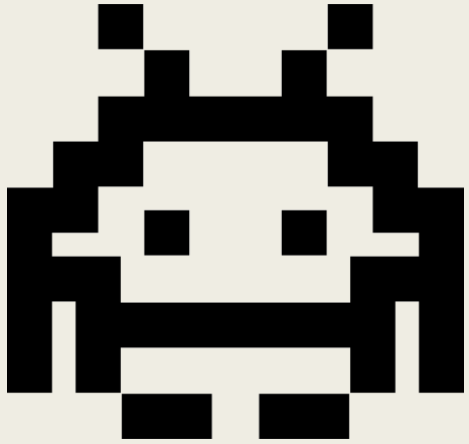
SPACE INVADERS

P2E GAME WITH SMART CONTRACTS



Project 3

EXECUTIVE SUMMARY

- Our group created the ability to play a retro arcade game, Space Invaders, using Solidity smart contracts and game tokens on the ethereum blockchain. We have two smart contracts, gametoken.sol and gameEscrow.sol, that contain all the instructions for our front end interface. The front end and game graphics were created using Javascript, HTML and CSS. We used Ganache/Metamask for our test network as well as Remix IDE and Visual Studio for our code writing.

TECHNOLOGIES

- Remix
- Solidity
- Visual Studio Code
- Live Server
- JavaScript
- HTML
- CSS
- MetaMask
- Ganache
- API Libraries
- openzepplin
- ERC20

P2E – Play to Earn gaming

- Play to earn is exactly what it sounds like, **video games where players have the ability to earn revenue while they play.** Unlike normal console or PC titles, P2E games give gamers the opportunity to earn revenue just by playing a video game.

GANACHE

- Quickly fire up a personal Ethereum blockchain which you can use to run tests, execute commands, and inspect state while controlling how the chain operates.

SOLIDITY

- Solidity is an object-oriented, high-level language for implementing smart contracts. Smart contracts are programs which govern the behavior of accounts within the Ethereum state.

gametoken Contract

- gametoken contract handles the token mints.
- Imports Ownable and ERC20 from openzeppelin
- Ownable allows the possibility to renounce ownership of the contract so nobody can own the token contract.
- Constructor runs once and mints 100 tokens and sends to devwallet (using metamask and linking ganache), as the ownable account and acts as admin and treasury reserve.

```
pragma solidity ^0.8.0;

import "@openzeppelin/contracts/utils/Context.sol";
import "@openzeppelin/contracts/access/Ownable.sol";
import "@openzeppelin/contracts/token/ERC20/ERC20.sol";

contract gametoken is Context, Ownable, ERC20 {
    /**
     * @dev Constructor that gives _msgSender() all of existing tokens.
     */
    constructor(string memory name, string memory symbol) ERC20(name, symbol) {
        // init circulation
        _mint(msg.sender, 100 * (10**uint256(decimals())));
    }


    // player must approve allowance for escrow/P2EGame contract to use (spender)
    function approve(address spender, uint256 amount)
        public
        virtual
        override
        returns (bool)
    {
        address owner = _msgSender();
        amount = 100 * (10**uint256(decimals())); // <-- 100 by default which is ma
        // amount = max possible to allow for better player UX (don't have to appro
        // in-game this means UX doesn't need to include call to approve each play,
        // TODO: player approves only amount needed each play
        _approve(owner, spender, amount);
        return true;
    }
}
```

Escrow Contract

- Escrow contract handles the flow of tokens.
- Acts as an escrow between players and admin/treasury.
- Imports Ownable from openzeppelin and gametoken.sol

```
pragma solidity ^0.8.0;

import "gametoken.sol";
import "@openzeppelin/contracts/access/Ownable.sol";

contract escrow is Ownable {

    address admin;
    uint256 constant _gameId = 1;

    uint256 public totalBalance;
    // this is the erc20 gametoken contract address
    address constant tokenAddress = 0xD4Aa127689f942Caf532fA1d0F6fBD15059F6901; // <-- IN

    // game data tracking
    struct Game {
        uint256 id;
        address treasury;
        uint256 amount;
        bool locked;
        bool spent;
    }
    // map game to balances
    mapping(address => mapping(uint256 => Game)) public balances;

    constructor() {
        admin = msg.sender;

    }

    // retrieve current state of game funds in escrow
    function startGame(address _treasury, uint256 _amount) external returns (uint256) {
        gametoken token = gametoken(tokenAddress);
        //approve contract to spend amount tokens
        require(token.approve(address(this), _amount), "Escrow: approval has failed");
        require(_amount >= 100000000000000000, "Escrow: must insert 1 whole token");
        token.transferFrom(msg.sender, address(this), _amount);

        totalBalance += _amount;
```

Front end game

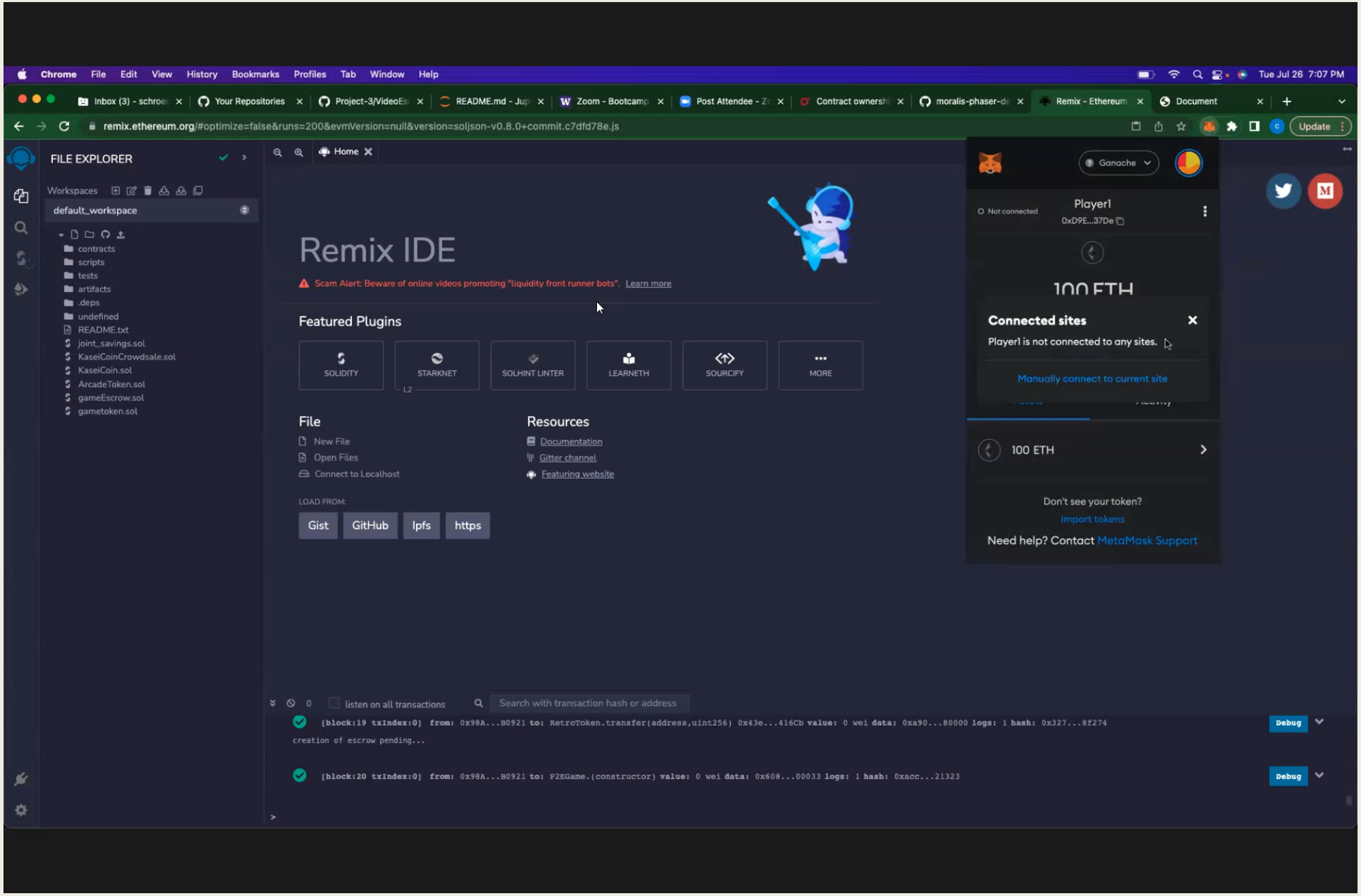
Player

gameEscrow.sol

gametoken.sol

Admin & Treasury Reserve

Demo

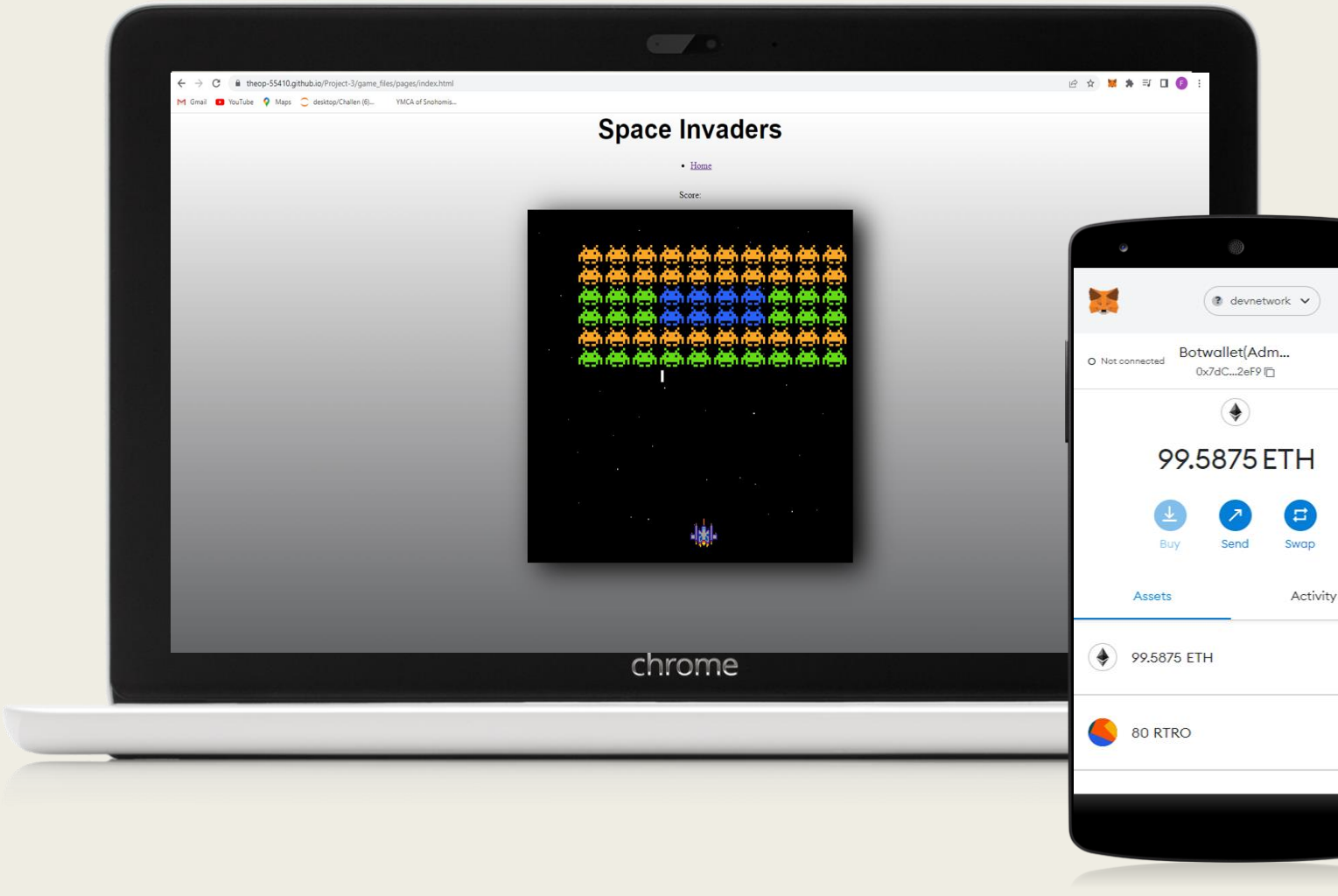

Chrome File Edit View History Bookmarks Profiles Tab Window Help
Tue Jul 26 7:07 PM
remix.ethereum.org/#optimize=false&runs=200&evmVersion=null&version=soljson-v0.8.0+commit.c7dfd78e.js
Update
FILE EXPLORER
Workspaces
default_workspace
contracts
scripts
tests
artifacts
.deps
undefined
README.txt
joint_savings.sol
KaseiCoinCrowdsale.sol
KaseiCoin.sol
ArcadeToken.sol
gameEscrow.sol
gametoken.sol
Home
Remix IDE
Scam Alert: Beware of online videos promoting "liquidity front runner bots". Learn more
Featured Plugins
SOLIDITY
STARKNET
L2
SOLHINT LINTER
LEARNETH
SOURCIFY
MORE
File
New File
Open Files
Connect to Localhost
Resources
Documentation
Gitter channel
Featuring website
LOAD FROM:
Gist
GitHub
Ipfs
https
Ganache
Not connected
Player1
0xD9E...37De
100 ETH
Connected sites
Player1 is not connected to any sites.
Manually connect to current site
100 ETH
Don't see your token?
Import tokens
Need help? Contact MetaMask Support
listen on all transactions
Search with transaction hash or address
[block:19 txIndex:0] from: 0x98A...B0921 to: RetroToken.transfer(address,uint256) 0x43e...416Cb value: 0 wei data: 0xa90...80000 logs: 1 hash: 0x327...8f274
Debug
creation of escrow pending...
[block:20 txIndex:0] from: 0x98A...B0921 to: P2EGame.(constructor) value: 0 wei data: 0x608...00033 logs: 1 hash: 0xacc...21323
Debug

POTENTIAL NEXT STEPS

- Adding to the contract, we would add different payouts, difficulty levels, and different games.
- Automating the bank so that the player can get tokens without admin. Adding functionality to the escrow contract so it’s automated instead of having an admin account.
- More efficient environment setup. Variables should be more automatic.

CONTRIBUTORS

- Aaron Bumgarner
- Aranda Furth
- Cody Schroeder
- Fadiya Ahmed
- Hilary Willis
- Theo Prentice



QUESTIONS?