CryptoZombie Starter Code Package Report.

Team Members:

Joshna Suresh – 885146902 [joshnasuresh04@csu.fullerton.edu]

Lemmie Carvalho – 870802634 [lemmiecarvalho0910@csu.fullerton.edu]

Shubham Jakhete – 823597208 [sjakhete@csu.fulelrton.edu]

Susan Dsouza – 809394463 [susandsouza@csu.fullerton.edu]

We have successfully executed the starter code works with create zombie, level up and display zombie functionalities.

Here are the following Improvement in the package:

1. Built a nicer website:

Before:

A screenshot of a computer

AI-generated content may be incorrect.

After:

A screenshot of a video game

AI-generated content may be incorrect.

1. Added images to demo zombies in a better way: A screenshot of a video game

   AI-generated content may be incorrect.
2. Changed the starter code to not hard-coded:

Modifications:

Remove Hardcoded cooldownTime: We can make the cooldown time configurable by allowing the owner to change it.

Remove Hardcoded levelUpFee: This can be updated dynamically by the contract owner.

Allow configurable dnaDigits and dnaModulus: Instead of hardcoding the DNA digits, we can allow the owner to update them, making it more flexible for future changes.

Updated Contracts

ZombieFactory.sol (with configurable cooldownTime and dna values)

pragma solidity ^0.4.25;

import "./ownable.sol";

import "./safemath.sol";

contract ZombieFactory is Ownable {

using SafeMath for uint256;

using SafeMath32 for uint32;

using SafeMath16 for uint16;

event NewZombie(uint zombieId, string name, uint dna);

uint dnaDigits = 16; // Default, can be changed

uint dnaModulus = 10 \*\* dnaDigits; // Default, can be changed

uint cooldownTime = 1 days; // Default, can be changed

struct Zombie {

string name;

uint dna;

uint32 level;

uint32 readyTime;

uint16 winCount;

uint16 lossCount;

}

Zombie[] public zombies;

mapping (uint => address) public zombieToOwner;

mapping (address => uint) ownerZombieCount;

// New function to allow changing of cooldownTime, dnaDigits, and dnaModulus

function setCooldownTime(uint \_time) external onlyOwner {

cooldownTime = \_time;

}

function setDnaDigits(uint \_digits) external onlyOwner {

dnaDigits = \_digits;

dnaModulus = 10 \*\* dnaDigits; // Update modulus based on new digits

}

function \_createZombie(string \_name, uint \_dna) internal {

uint id = zombies.push(Zombie(\_name, \_dna, 1, uint32(now + cooldownTime), 0, 0)) - 1;

zombieToOwner[id] = msg.sender;

ownerZombieCount[msg.sender] = ownerZombieCount[msg.sender].add(1);

emit NewZombie(id, \_name, \_dna);

}

function \_generateRandomDna(string \_str) private view returns (uint) {

uint rand = uint(keccak256(abi.encodePacked(\_str)));

return rand % dnaModulus;

}

function createRandomZombie(string \_name) public {

require(ownerZombieCount[msg.sender] == 0);

uint randDna = \_generateRandomDna(\_name);

randDna = randDna - randDna % 100;

\_createZombie(\_name, randDna);

}

}

ZombieHelper.sol (with configurable levelUpFee)

pragma solidity ^0.4.25;

import "./zombiefeeding.sol";

contract ZombieHelper is ZombieFeeding {

uint levelUpFee = 0.001 ether; // Default fee, can be changed

modifier aboveLevel(uint \_level, uint \_zombieId) {

require(zombies[\_zombieId].level >= \_level);

\_;

}

function withdraw() external onlyOwner {

address \_owner = owner();

\_owner.transfer(address(this).balance);

}

// Function to allow owner to change the level up fee

function setLevelUpFee(uint \_fee) external onlyOwner {

levelUpFee = \_fee;

}

function levelUp(uint \_zombieId) external payable {

require(msg.value == levelUpFee);

zombies[\_zombieId].level = zombies[\_zombieId].level.add(1);

}

function changeName(uint \_zombieId, string \_newName) external aboveLevel(2, \_zombieId) onlyOwnerOf(\_zombieId) {

zombies[\_zombieId].name = \_newName;

}

function changeDna(uint \_zombieId, uint \_newDna) external aboveLevel(20, \_zombieId) onlyOwnerOf(\_zombieId) {

zombies[\_zombieId].dna = \_newDna;

}

function getZombiesByOwner(address \_owner) external view returns(uint[]) {

uint[] memory result = new uint[](ownerZombieCount[\_owner]);

uint counter = 0;

for (uint i = 0; i < zombies.length; i++) {

if (zombieToOwner[i] == \_owner) {

result[counter] = i;

counter++;

}

}

return result;

}

}

ZombieAttack.sol (no significant changes needed here, unless you want more configurability)

pragma solidity ^0.4.25;

import "./zombiehelper.sol";

contract ZombieAttack is ZombieHelper {

uint randNonce = 0;

uint attackVictoryProbability = 70;

function randMod(uint \_modulus) internal returns(uint) {

randNonce = randNonce.add(1);

return uint(keccak256(abi.encodePacked(now, msg.sender, randNonce))) % \_modulus;

}

function attack(uint \_zombieId, uint \_targetId) external onlyOwnerOf(\_zombieId) {

Zombie storage myZombie = zombies[\_zombieId];

Zombie storage enemyZombie = zombies[\_targetId];

uint rand = randMod(100);

if (rand <= attackVictoryProbability) {

myZombie.winCount = myZombie.winCount.add(1);

myZombie.level = myZombie.level.add(1);

enemyZombie.lossCount = enemyZombie.lossCount.add(1);

feedAndMultiply(\_zombieId, enemyZombie.dna, "zombie");

} else {

myZombie.lossCount = myZombie.lossCount.add(1);

enemyZombie.winCount = enemyZombie.winCount.add(1);

\_triggerCooldown(myZombie);

}

}

}

Summary of Changes: We gave the Crypto Zombie starter code a real boost by making it easier to change and nicer to use. On the front end, we spruced up the design and added fun zombie images to give it more personality. In the smart contracts, we took out the hard-coded values like cooldown times, DNA settings, and the level-up fee and replaced them with options that can be tweaked anytime. This way, the core features of creating, leveling, and battling zombies stay the same, but now everything’s more flexible and better suited for future changes.

Configurable Parameters:

cooldownTime: Can now be changed by the owner using the setCooldownTime() function.

dnaDigits and dnaModulus: Allow the owner to change the DNA configuration.

levelUpFee: Configurable fee for leveling up zombies, adjustable by the contract owner.

These changes allow the contract to be more flexible without hardcoded values, giving the owner the ability to adjust these parameters as needed.