



메이플스토리st 웹3 게임

Joyful Maplestory NFT

GitHub: <https://github.com/hyenne/BEB-05-JMT>

LIVE: <https://jmt-maplestory.com>

이혜인 강인국 김도영 김병일 박기백



Table of Contents

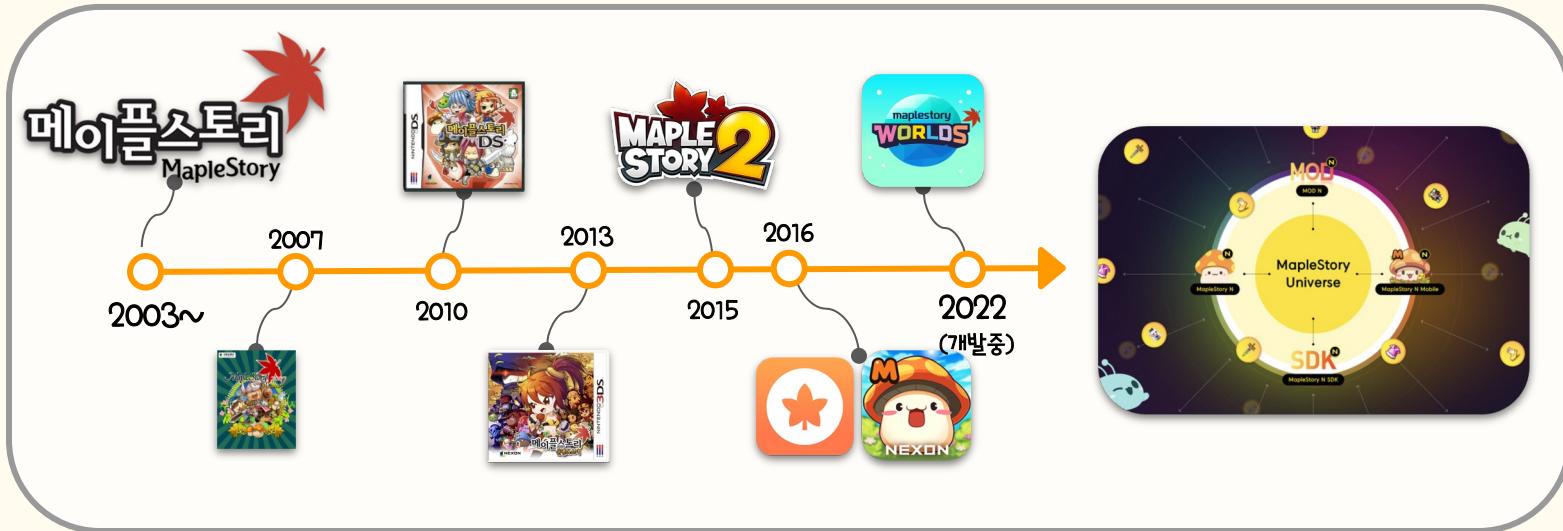
- 1. About JMT**
- 2. Tech Stack**
- 3. Demo**
- 4. JMT Team & QnA**
- 5. Future Works**

01 About JMT

메이플st 웹3 게임 기획 의도

01 About JMT

즐거운 메이플스토리 JMT 소개



메이플스토리는 2003년 부터 파생 서비스를 런칭, 블록체인과 메타버스까지 확장 중

→ NFT와 디파이를 활용하여 메이플스토리를 블록체인 웹3 게임으로 개발 도전

01 About JMT

토큰 구성



ERC20

인게임 재화 JMT 토큰 및
수수료 vJMT 토큰



ERC721

고유한 캐릭터



ERC1155

무기 및 강화서

01 About JMT

토큰 이코노미



Game Currency JMT

- 무기, 캐릭터 민팅 및 장비 강화 등 게임 컨텐츠를 즐기는데 소비
- 랭킹, 전투 승리 등을 통하여 수령



Invest Currency vJMT

- 사용자는 게임 컨텐츠를 통해 소비한 JMT 수수료만큼 vJMT 수령
- vJMT를 이용한 디파이를 통해 JMT 토큰 수령

01 About JMT

토큰 이코노미

Game

민팅 / 거래 / 보상



JMT

De-Fi

Token Swap

Stacking

Deposit

LP
(유동성 풀)



vJMT



02 Tech Stack

기술 스택

02 Tech Stack

사용 기술 스택

Front End



Back End



Smart Contract



SOLIDITY



TRUFFLE



Ganache



Testing



Operation



Pinata



API

<https://maplestory.io/>

02 Tech Stack

폴리곤 네트워크



ethereum

초당 20TPS

건당 10\$ 이상



polygon

초당 65,000TPS

건당 0.03\$

03 Demo

기술 시연

03 Demo

로그인



03 Demo

첫 토큰 스왑



03 Demo

첫 민팅 회원가입 및 장착



트랜잭션

- 캐릭터 랜덤 민팅
- 첫 무기 랜덤 민팅
- 장착

DB

```
mysql> select * from User;
+----+-----+-----+-----+-----+
| id | username | address | charId | weaponId |
+----+-----+-----+-----+-----+
| 1  | great   | 0xac870889c6c70305186afe8131d6f264728d8027 | 21    | 0        |
| 2  | cool    | 0x2ef3ea4722727b4302fcc6dc2e8c556feea1edcc | 28    | 200     |
+----+-----+-----+-----+-----+
2 rows in set (0.02 sec)

mysql>
```

03 Demo

랜덤 민팅

캐릭터: ERC721

무기: ERC1155



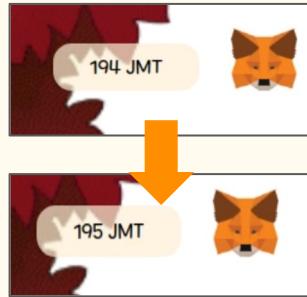
03 Demo

전투



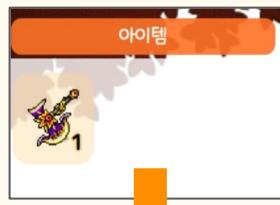
03 Demo

전투 보상 수령



보상 확률

- 1 JMT 90%
- 스크롤 70%



스크롤 민팅 확률

- 강화 주문서 100: 5%
- 강화 주문서 90: 10%
- 강화 주문서 60: 20%
- ...

03 Demo

인벤토리 캐릭터 및 무기 장착

캐릭터 장착



무기 장착



03 Demo

인벤토리 무기 강화



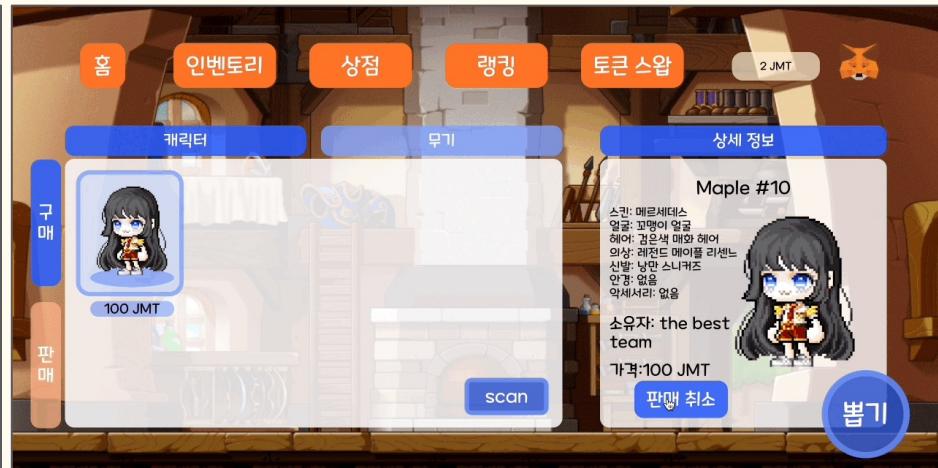
03 Demo

마켓 판매 및 판매 취소

판매



판매 취소



03 Demo

마켓 구매 및 수수료

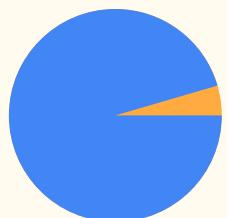
판매



구매



거래금액



수수료



токен 정보

유동성 관리



03 Demo

랭킹 보상

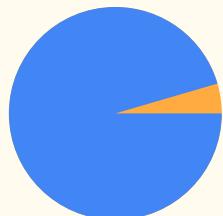
거래 수수료 보상 반영



랭킹 유저 폴리곤 스캔



거래금액

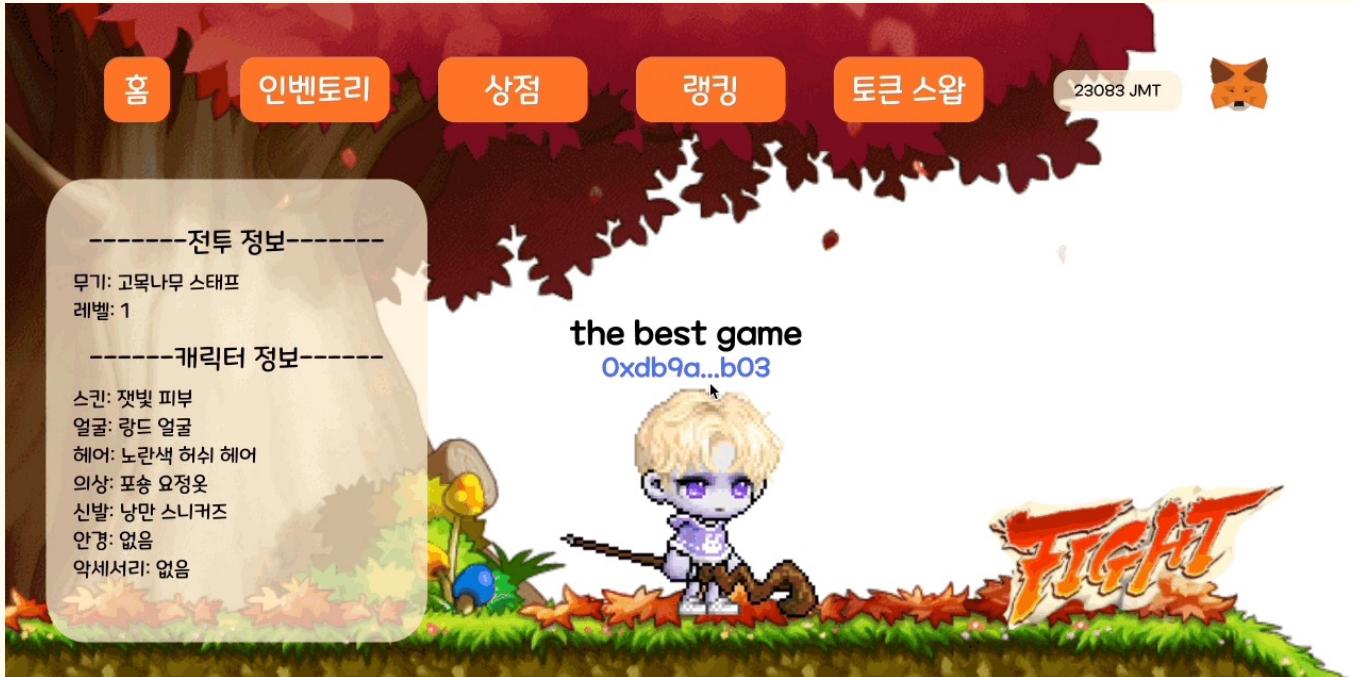


수수료



03 Demo

폴리곤 스캔 유저 계정



03 Demo

폴리곤 스캔 인벤토리

캐릭터: ERC721

무기: ERC1155



03 Demo

폴리곤 스캔 마켓

캐릭터: ERC721

무기: ERC1155



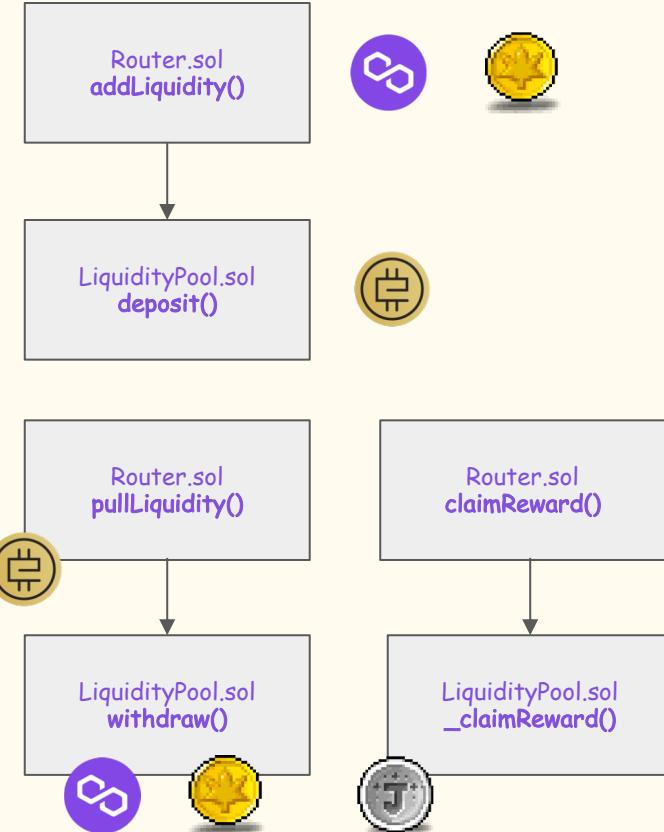
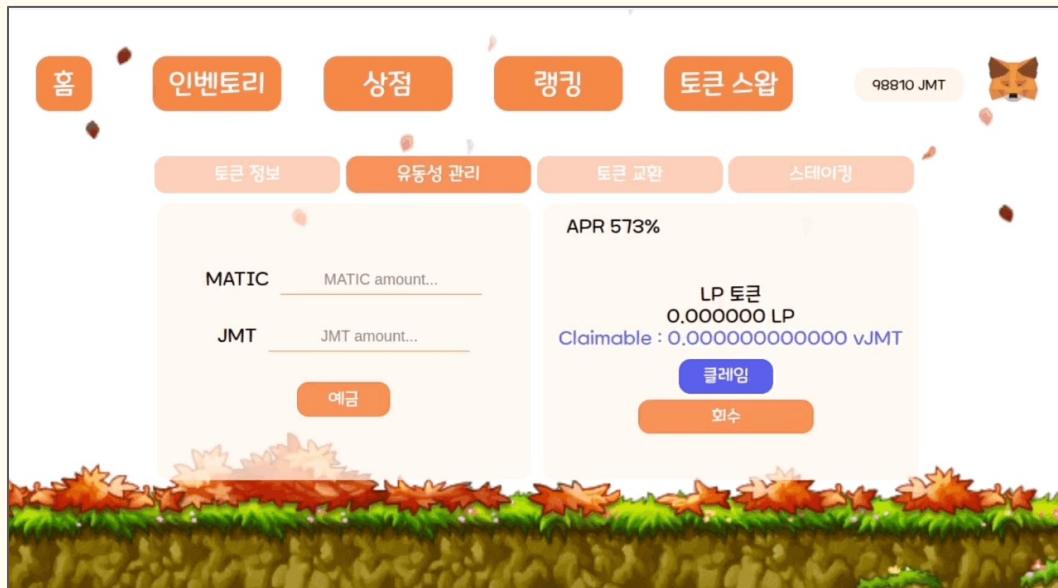
03 Demo

토큰 정보



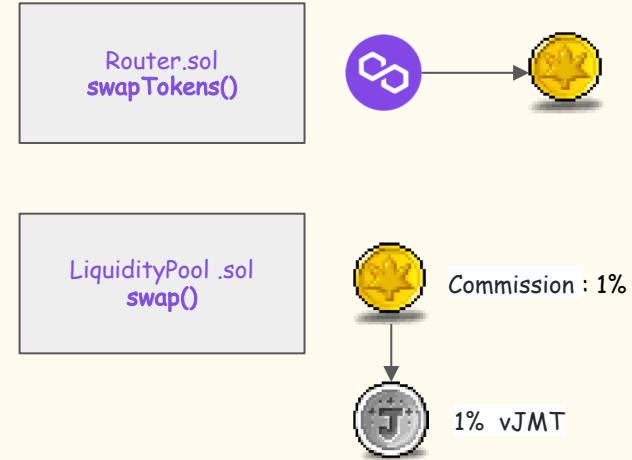
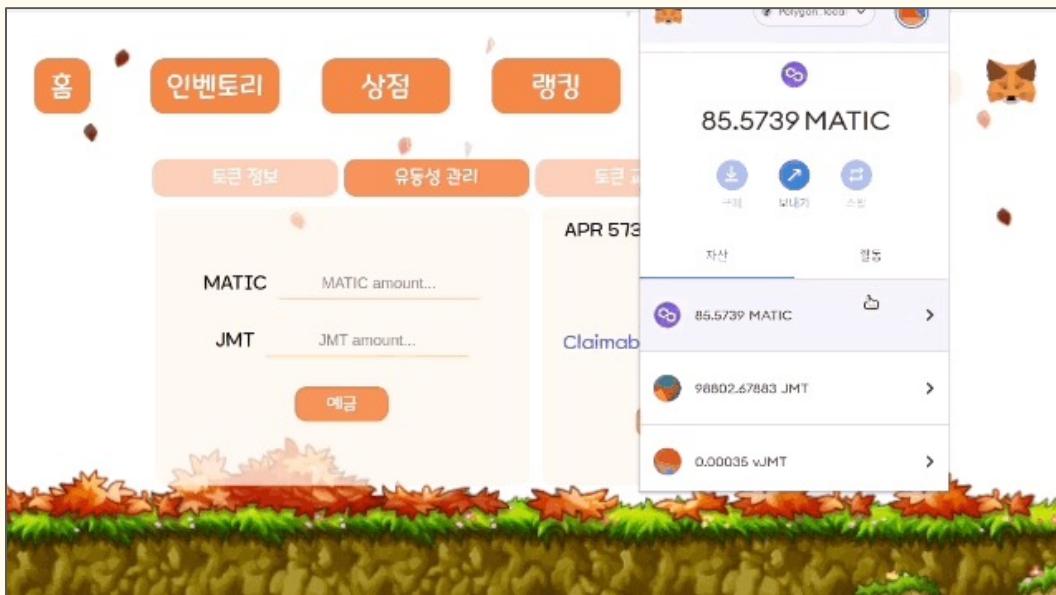
03 Demo

유동성 제공



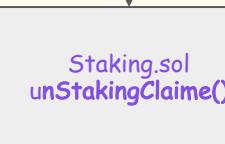
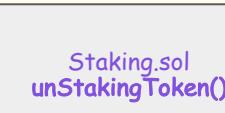
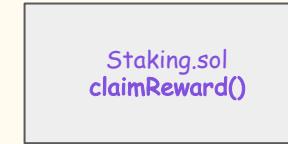
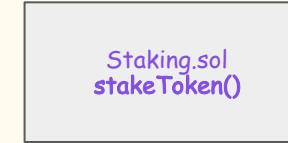
03 Demo

토큰 스왑



03 Demo

스테이킹



TimeLock: 3일

04 JMT Team & QnA

팀원 소개 및 QnA

04 JMT Team and Qna

팀원 소개

- ERC721 기반 캐릭터 NFT, ERC1155 기반 아이템 시스템 구현 (민팅/거래/인벤토리/강화/랭킹 보상)
- 메이플 애셋 연동 및 서버/프론트엔드 /컨트랙트 코드 구조 정립
- 폴리곤 테스트넷/메인넷, AWS를 이용해 서버 배포

- 스왑 페이지 css, 스테이킹 페이지 api 연결 및 기능 구현
- 토큰 스왑시 vJMT 토큰 지급 컨트랙트 구현
- 로딩 구현
- 인벤토리 페이지 구현

- 전투 페이지 구현
- 전투 컨트랙트 구현
- 전투 이미지 API 구현
- JMT 토큰, 강화서 랜덤 보상 컨트랙트 구현 및 API 구현

- JMT, vJMT 컨트랙트 구현
- 스왑, 스테이킹, 유동성풀, 라우터 컨트랙트 구현
- Swap Front 개발
- Polygon node 구성 및 테스트
- 토큰 이코노미 설계

- 마켓 페이지
- 폴리곤 배포 조사
- 토큰 이코노미 설계



04 JMT Team and Qna

Q1. 혜인

Trustless 민팅을 구현하며 어려웠던 점?

04 JMT Team and Qna

A1. 혜인

시중 NFT프로젝트의 민팅 알고리즘 종류

1. tokenURI를 IPFS를 쓰지않고 별도 api를 쓴다.
2. tokenURI를 owner가 수정할 수 있는 함수를 만들어 놓는다.
3. 단순히 tokenId를 1부터 10000까지 iterate한다.

```
--  
124 function mint(uint numberOFTokens) public payable {  
125     uint256 ts = totalSupply();  
126     require(saleIsActive, "Sale must be active to mint tokens");  
127     require(numberOFTokens <= MAX_PUBLIC_MINT, "Exceeded max token purchase");  
128     require(ts + numberOFTokens <= MAX_SUPPLY, "Purchase would exceed max tokens");  
129     require(PRICE_PER_TOKEN * numberOFTokens <= msg.value, "Ether value sent is not correct");  
130  
131     for (uint256 i = 0; i < numberOFTokens; i++) {  
132         _safeMint(msg.sender, ts + i);  
133     }  
134 }
```

이더스캔에서 확인한 Doodle 프로젝트의 mint 함수

- => 토큰 데이터가 언제든 개발자의 의해 바뀔 수 있음
=> 위와 같음
=> 결과를 미리 알거나, 민팅시점에 토큰 데이터가 불투명한 경우

```
48     function mintMapleNFT() public payable returns (uint256) {  
49         require(token.balanceOf(msg.sender) >= mintPrice, "ERC721: recipient lack of erc20 balance");  
50         require(maxTokenNum >= totalSupply(), "ERC721: all nfts are minted");  
51         // waitForMint는 0~4999까지 준비된 tokenId가 들어있는 배열  
52         _shuffleMintArray(); // waitForMint[-1]과 waitForMint[랜덤인덱스]의 위치를 바꾼다.  
53         uint256 minted = waitForMint[waitForMint.length - 1];  
54         waitForMint.pop();  
55  
56         _mint(msg.sender, minted);
```

JMT 프로젝트의 mint 함수

구현한 민팅 알고리즘

1. tokenId 별 토큰 데이터는 민팅 전에 IPFS에 공개돼 있어야 한다.
2. 민팅의 tokenId는 incremental이 아닌 random order로 변해야 한다. (keccak이 아닌 오라클 사용)
3. 개발자는 임의로 tokenURI를 바꿀 수 없어야 한다.

04 JMT Team and Qna

Q2. 인국

스마트 컨트랙트를 api로 연결하면서 어려웠던점 또는 배운점은?

04 JMT Team and Qna

A2. 인국

스마트 컨트랙트를 api로 연결하면서 어려웠던점 또는 배운점은?

```
// 보상 가능한 수량
function claimableReward() public view returns(uint256) {
    const viewReward = async(address) => {
        const web3 = new Web3(window.ethereum);
        const StakingContract = await new web3.eth.Contract(
            STAKING_CONTRACT_ABI,
            STAKING_CONTRACT_ADDR
        );
        const result = await StakingContract.methods.claimableReward().call({from:address});
        return result;
    }
}
```

-스마트컨트랙트와 API 흐름 이해.

- send(), call() 사용 이해.

-스마트 컨트랙트에서의 변수 설정과 payable, view 이해.

```
//토론 스테이킹
function stakeToken(uint256 stakeAmount) public payable Claimable(stakeAmount) returns(bool){
    const vJMTStaking = async(amount, address) => {
        try{
            const web3 = new Web3(window.ethereum);
            var BN = web3.utils.BN;
            const _amount = new BN(String(amount)).mul(new BN(String(10**18))).toString();
            const StakingContract = await new web3.eth.Contract(
                STAKING_CONTRACT_ABI,
                STAKING_CONTRACT_ADDR
            );
            const result = await StakingContract.methods.stakeToken(_amount).send({
                from:address,
                gas: 1500000,
                gasPrice: '3000000'
            });
        }
    }
}
```

04 JMT Team and Qna

Q3. 도영

전투를 구현하면서 깨달은 점?

A3. 도영

전투를 구현하면서 깨달은 점?

```
function setFight(address _addr, uint _userstrength, uint _matchingstrength) public {
    require( msg.sender == _addr, "Incorrect Address");
    uint rand = randMod(100);
    if ( _userstrength > _matchingstrength ) {
        result = "User Win!!!";
    } else if ( rand <= rewardProbability ) { // 10%의 확률로 유저에게 크리티컬 발동
        _userstrength = _matchingstrength + 1;
        result = (_userstrength >= _matchingstrength? "User dramatic Win !!": "User Lose...");
    } else if ( _userstrength == _matchingstrength ) { // 비겼을 때, 50%확률로 승리
        result = ( rand <= rewardProbability2 ? "User dramatic Win !!": "User Lose...");
    } else {
        result = "User Lose...";
    }
}
```

```
function setFight(address _addr, uint _userstrength, uint _matchingstrength) public {
    require( msg.sender == _addr, "Incorrect Address");
    uint rand = randMod(100);
    if ( _userstrength > _matchingstrength ) {
        result = "User Win!!!";
    } else if ( rand < rewardProbability ) { // 10%의 확률로 유저에게 크리티컬 발동
        _userstrength = _matchingstrength + 1;
        result = (_userstrength >= _matchingstrength? "User dramatic Win !!": "User Lose...");
    } else if ( _userstrength == _matchingstrength ) { // 비겼을 때, 50%확률로 승리
        result = ( rand < rewardProbability2 ? "User dramatic Win !!": "User Lose...");
    } else {
        result = "User Lose...";
    }
}
```

- ‘>=’, ‘<=’ 부등호를 쓰면 사실상, 두 가지 일을 처리해서 가스비가 많이 든다는 것을 발견
--> ‘=’ 연산자를 제거하고 비교 값 중 하나를 증가시켜 가스비를 절약

04 JMT Team and Qna

Q4. 병일

유동성 풀을 구현하면서 어려웠던 점?

04 JMT Team and Qna

A4. 병일

스왑풀과 유동성을 구현하면서 어려웠던 점?

Swap 프로세스 구현

```
uint256 currentJMTBalance = jmtToken.balanceOf(address(this)); // 풀에 가지고있는 컨트랙트 코인 보유량
uint256 _jmtAmountMinusTax = _jmtAmount - ((2 * _jmtAmount) / 100); // (갖고싶은 토큰양 - 수수료)
uint256 addedBalance = jmtReserve + _jmtAmountMinusTax; // 현재 jmt보유량 + (갖고싶은 토큰양 - 수수료) = 추가된 총 밸런스
require(addedBalance == currentJMTBalance, "DID_NOT_TRANSFER");

uint256 x = product / (_jmtReserve + _jmtAmountMinusTax); // 미더가 product를 맞추기위한 비율
amountToTransfer = ethReserve - x; // 유저가 받을 eth
amountToTake = (1 * amountToTransfer) / 100; // 지불해야하는 수수료
totalAmountToTransfer = amountToTransfer - amountToTake; // 받을 토큰 - 수수료 = 최종 get eth
|
// 미더를 안전하게 보내는 방법
// https://kushgoyal.com/ethereum-solidity-how-use-call-delegatecall/ -> 참고
(bool success, ) = account.call{value: totalAmountToTransfer}("");
require(success, "TRANSFER_FAILED");
```

LiquidityPool Reserve 동기화 문제

```
function _update() private {
    uint32 blockTimestamp = uint32(block.timestamp % 2**32);
    uint32 timeElapsed;
    unchecked { // 새로운 블록이라면 오버플로우가 생김
        timeElapsed = blockTimestamp - lastBlockTimestamp;
    }
    if (timeElapsed > 0) {
        ethReserve = address(this).balance;
        jmtReserve = jmtToken.balanceOf(address(this));
        lastBlockTimestamp = blockTimestamp;
    }
}
```

04 JMT Team and Qna

Q5. 기백

마켓 작업 시 기억에 남았던 것은 무엇인가요?

04 JMT Team and Qna

A5. 기백

마켓 작업 시 기억에 남았던 것은 무엇인가요?

- Market contract.sol - sell_function()
 - Promise 객체 send / contract sell의 혼동
- Polygon deploy investigation
 - polygon의 matic과 wETH간의 연동
 - erc20과 matic 변환

```
.send({  
  from: '0x12345678901234567890',  
  gas: 1500000,  
  gasPrice: '30000000000'  
})  
=> msg.value로 널어감
```

```
.sell({  
  from: '0x12362344623462435720',  
  price: '3000000000'  
})  
=> market_contract.sol 널어감
```

05 Future Works

개선 방안

05 Future Works

개선 방안 기능



JMT TEAM

[로그인] 다양한 지갑 연결.

[랜덤요소] 랜덤 요소에 chainlink를 적용하여 오라클 문제 해결 시도.

[히스토리] 이용자가 전투 및 거래 History를 볼 수 있는 기능.

[전투] 하루에 할 수 있는 전투 횟수 제한.

[게임 시스템] 레벨 시스템 발전. 레벨에 따라 보상 지급량과 강화서 확률 변화 등.

[스왑] Factory 패턴을 적용하여 다른 토큰들과의 스왑을 진행 할 수 있게 구조 변경.

유동성 풀에 복리(APY) 구조를 넣어 사용자에게 더 높은 이자율을 제공.

다음▶

대화 그만하기

05 Future Works

개선 방안 기능



JMT TEAM

[컨트랙트 조회 및 트랜잭션 생성]

사용자가 많아질 경우를 대비하여, 5분 단위로 조회, 업데이트 하는 방식.

erc1155의 경우 batch를 사용하여 서버에서 특정 시간에 쌓인 여러 사용자의 요청을 한꺼번에 처리.

가스비 소모를 줄이기 위한 컨트랙트 코드 개선.

[트랜잭션 ux 개선]

폴리곤 테스트넷에 라이브 배포시 로컬과 다르게 레이턴시로 인해 프론트엔드 코드 버그 발생.

수정했지만, 처음부터 백그라운드 로딩처럼 레이턴시를 감안해서 설계했다면 훨씬 더 매끄러운 ux를 제공할 수 있었을 것으로 기대.

다음▶

대화 그만하기

05 Future Works

개선 방안 서비스



JMT TEAM

[OpenSea 컬렉션 등록]

추후 외부에서 거래 가능하도록 tokenURI 설계해놓았음.

[블록체인 커뮤니티]

- > 길드 컨텐츠.
- > 길드에 속한 사람들은 전투시 레어한 확률로 얻는 보석을 나눠 갖게 됨.
- > 길드에 속한 사람들이 많을 수록 확률이 높지만 대신 효과가 나눠져서 장점&단점 존재.
- > 나누는 효과나 당첨 확률을 길드 사람 수에 따라 다르게 할 수도 있겠다는 생각.

다음▶

대화 그만하기



JMThank You

