1.访问https://chainlist.org/

搜索80001

添加网络到metamask

2.合约有3个：greeter，mynft，player，可以直接用player的

CONTRACT_ADDRESS = 0xE15b04371E81ac6986683e00d9975ad0D2519b31

CONTRACT_ADDRESS2 = 0x29644099867812852041AFE6Eb4A5B8D67B9d590

CONTRACT_ADDRESS3 = 0x238e289247CD6122EdCd65216A9b8bE4e8f963BA

https://mumbai.polygonscan.com/address/0x238e289247cd6122edcd65216a9b8be4e8f963ba

3.合约测试过程（代码库在https://github.com/jhfnetboy/pure-Game)

4.建议直接使用front前端代码来调用合约库（有范例，例如fetchWalkerName等，搜索下即可）

合约部分和游戏设定、经济模型有关，合约和前端的交互，就是import contract的abi，通过web3交互

这部分已经在https://github.com/jhfnetboy/pure-game-front 跑通

5.未来合约要本地化部署，这部分经验在https://github.com/jhfnetboy/rarity_like_game

6.其他部分可以结对编程，快速传递一些经验和技巧

以下是合约调试过程，参考

==========

合约完成了初步Walker的基础NFT创建✅

进行合约试验第三步，属性随机生成/人工分配后并存储❎

进行合约试验第三步，建立物品NFT合约以及限量发行（滞后发行？），建立金币和多签钱包，运营工会控制发行❎

进行合约试验第四步，建立人物的inventory（1155协议），包含ERC20和721❎

========

合约试验

1.greeter发布到poly

```
yarn add  @openzeppelin/contracts
yarn add dotenv

cat .env
PRIVATE_KEY=fc4758bc5074bd671f12903e7dbd17bcb95e1e27b84320mb51e8b3b26eb2bae
CONTRACT_ADDRESS = 0xE15b04371E81ac6986683e00d9975ad0D2519b31
WALLET_ADDRESS=0x0bE9d4386f58022c27B0Fb035c46de7c5800938F
YOUR_METADATA_CID=QmNmv3TGpzaDaX92rX9fzRch2FHeFQqBW5k51z1p7kHBVM

echo ".env">> .gitignore

set default network to matic in hardhat.config.js:
module.exports = {
  solidity: "0.8.7",
  defaultNetwork: "matic",
  paths: {
    artifacts: './src/artifacts',
  },
  networks: {
    matic: {
      url: "https://rpc-mumbai.maticvigil.com",
      accounts: [PRIVATE_KEY]
    },

npx hardhat run scripts/test_deploy.js --network matic
```

2.js方式获取greeter链上合约的greeting文本

```
 npx hardhat run scripts/test_deploy.js --network matic

 test_deploy:
 const Greeter = await hre.ethers.getContractFactory("Greeter");
console.log('get contract')
const greeter = await Greeter.deploy("Test for deploy on matic");
await greeter.deployed();
console.log("Contract deployed to:", greeter.address);

output:
get contract
Contract deployed to: 0x3864582dF220cCD30846857d260ad7D183DAE85B
```

部署到poly链上后：https://mumbai.polygonscan.com/address/0x3864582dF220cCD30846857d260ad7D183DAE85B

本地js链上调用：

```
npx hardhat run scripts/greeter-test.js --network matic

greeter-test:
async function main() {
  const Contract = await hre.ethers.getContractFactory("Greeter");
  const contract = Contract.attach("0x3864582dF220cCD30846857d260ad7D183DAE85B");
  const msg = await contract.greet();
  console.log("Get this:", msg);
}

output:
Get this: Test for deploy on matic
```

3.NFT合约试验

要移除contract下的其他sol，以及artifacts下的sol目录，只做NFT

```
npx hardhat run scripts/MyNFT-deploy.js --network matic

MyNFT-deploy.js:
async function main() {
  const NFT = await hre.ethers.getContractFactory("MyNFT");
  const nft = await NFT.deploy();
  await nft.deployed();
  console.log("MyNFT NFT Factory Contract deployed to:", nft.address);
}

output:
MyNFT NFT Factory Contract deployed to: 0xE15b04371E81ac6986683e00d9975ad0D2519b31
```

基于NFT工厂合约，mint几个NFT

```
npx hardhat run scripts/mint-mynft.js --network matic


mint-mynft.js:
const CONTRACT_ADDRESS = process.env.CONTRACT_ADDRESS;
const WALLET_ADDRESS = process.env.WALLET_ADDRESS;
const URI = "ipfs://"+process.env.YOUR_METADATA_CID
const ConractFactory = "MyNFT"
```

```
async function main() {
  const NFT = await hre.ethers.getContractFactory(ConractFactory);
  const contract = NFT.attach(CONTRACT_ADDRESS);
   const newID = await contract.mint(WALLET_ADDRESS, URI);
//   console.log("NFT minted:", contract);
  console.log("NFT minted with ID: ", newID);
}

output: 返回的不是id，是一个结构体


NFT minted with ID:  {
  hash: '0xd7cee872191c2b58ff643758e33fae2965d7092420952e7cf7707ee124143682',
  type: 0,
  accessList: null,
  blockHash: null,
  blockNumber: null,
  transactionIndex: null,
  confirmations: 0,
  from: '0x0bE9d4386f58022c27B0Fb035c46de7c5800938F',
  gasPrice: BigNumber { _hex: '0xb2d05e00', _isBigNumber: true },
  gasLimit: BigNumber { _hex: '0x01ecca', _isBigNumber: true },
  to: '0xE15b04371E81ac6986683e00d9975ad0D2519b31',
  value: BigNumber { _hex: '0x00', _isBigNumber: true },
  nonce: 3,
  data:
'0xd0def521000000000000000000000000be9d4386f58022c27b0fb035c46de7c5800938f000000000000
0000000000000000000000000000000000000000000000000040000000000000000000000000000000000000
0000000000000000000000000035697066733a2f2f516d4e6d76335447707a6144615839327258396 67a52
6368324648654651714257356b35317a3170376b4842564d0000000000000000000000',
  r: '0x202294abc7aa5ee6ba2ae749b5178934d29749592d32a3aca52edafad4844f18',
  s: '0x196f5b7d35490c470fdac082790464a01b63c18b8841cc8d0f00c8d96c5cd2ad',
  v: 160037,
  creates: null,
  chainId: 80001,
  wait: [Function (anonymous)]
}
```

检查我（钱包：）的mint的链上NFT信息

```
npx hardhat run scripts/check-mynft.js --network matic

check-mynft.js:
async function main() {
  const NFT = await hre.ethers.getContractFactory("MyNFT");
  const contract = NFT.attach(CONTRACT_ADDRESS);
```

```
  //检查某钱包地址有几个NFT
  const num = await contract.balanceOf(WALLET_ADDRESS)
  console.log("Ownend NFT number: ", parseInt(num,16));
  //检查第一个NFT是哪个钱包持有
  const owner = await contract.ownerOf(1);
  console.log("Owner:", owner);
  //检查第一个NFT的URI（图片网址）是啥
  const uri = await contract.tokenURI(1);
  console.log("URI: ", uri);
}

output:
Ownend NFT number:  BigNumber { _hex: '0x02', _isBigNumber: true }
Owner: 0x0bE9d4386f58022c27B0Fb035c46de7c5800938F
URI:  ipfs://QmNmv3TGpzaDaX92rX9fzRch2FHeFQqBW5k51z1p7kHBVM
```

4.Walker合约

1》基于NFT修改的简单合约，创建人物Walker NFT，无六属性

mint需要提供：name（不唯一，但NFT唯一）✅

性别、职业、属性（先不提供，默认随机）

修改合约ing

观察MyNFT合约代码，代入NFTtoken的数据URI，从构造函数传入：

```
pragma solidity ^0.8.7;
// SPDX-License-Identifier: UNLICENSED
import "@openzeppelin/contracts/token/ERC721/ERC721.sol";
import "@openzeppelin/contracts/utils/Counters.sol";
import "@openzeppelin/contracts/access/Ownable.sol";
contract MyNFT is ERC721, Ownable {
  using Counters for Counters.Counter;
  using Strings for uint256;
  Counters.Counter private _tokenIds;
  mapping (uint256 => string) private _tokenURIs;

  constructor() ERC721("MyNFT", "MNFT") {}
  function _setTokenURI(uint256 tokenId, string memory _tokenURI)
    internal
    virtual
  {
    _tokenURIs[tokenId] = _tokenURI;
  }
  function tokenURI(uint256 tokenId)
    public
    view
```

```
    virtual
    override
    returns (string memory)
  {
    require(_exists(tokenId), "ERC721Metadata: URI query for nonexistent token");
    string memory _tokenURI = _tokenURIs[tokenId];
    return _tokenURI;
  }
  function mint(address recipient, string memory uri)
    public
    returns (uint256)
  {
    _tokenIds.increment();
    uint256 newItemId = _tokenIds.current();
    _mint(recipient, newItemId);
    _setTokenURI(newItemId, uri);
    return newItemId;
  }
}
```

测试mint时传入不同参数（例如玩家名称name），ok

```
npx hardhat run scripts/check-mynft.js --network matic
balanceOf,Ownend NFT number:  3
ownerOf 1,Owner: 0x0bE9d4386f58022c27B0Fb035c46de7c5800938F
tokenURI 1,URI:  ipfs://QmNmv3TGpzaDaX92rX9fzRch2FHeFQqBW5k51z1p7kHBVM
tokenURI 2,URI:  ipfs://QmNmv3TGpzaDaX92rX9fzRch2FHeFQqBW5k51z1p7kHBVM
tokenURI 3,URI:  ipfs://QmNmv3TGpzaDaX92rX9fzRch2FHeFQqBW5k51z1p7kHBVMTest3 for NFT
Private info
name:  MyNFT
symbol:  MNFT
ownerOf:  0x0bE9d4386f58022c27B0Fb035c46de7c5800938F
ownerOf2:  0x0bE9d4386f58022c27B0Fb035c46de7c5800938F
```

不赘述，测试结果

```
npx hardhat run scripts/check-WalkerNFT.js --network matic
balanceOf,Ownend NFT number:  1
ownerOf 1,Owner: 0x0bE9d4386f58022c27B0Fb035c46de7c5800938F
tokenWalkerName 1,name:  myWalkerName
name:  Walker
symbol:  WAT
```

=========

进行合约试验第二步，修改名称以及记录功勋值（回合体力本地计算）

1.增加功勋值记录属性，初始0

2.修改名称，需要增加public修改方法；修改功勋，需要增加public方法

3.测试效果(新部署的合约)

```
npx hardhat run scripts/check-WalkerNFT.js --network matic
balanceOf,Ownend NFT number:  1
ownerOf 1,Owner: 0x0bE9d4386f58022c27B0Fb035c46de7c5800938F
tokenWalkerName 1,name:  myWalkerName
tokenWalkerName 1,name:  {
  hash: '0x6361b5df2cc358ea765f4232d4c23fbfd2cafb735e863574fd2e5e7d9db7ccbb',
  type: 0,
  accessList: null,
  blockHash: null,
  blockNumber: null,
  transactionIndex: null,
  confirmations: 0,
  from: '0x0bE9d4386f58022c27B0Fb035c46de7c5800938F',
  gasPrice: BigNumber { _hex: '0xee6b2800', _isBigNumber: true },
  gasLimit: BigNumber { _hex: '0x77de', _isBigNumber: true },
  to: '0x238e289247CD6122EdCd65216A9b8bE4e8f963BA',
  value: BigNumber { _hex: '0x00', _isBigNumber: true },
  nonce: 10,
  data:
'0x320f136a0000000000000000000000000000000000000000000000000000000000000001000000000000000000000000000000000000000000000000000000000000004000000000000000000000000000000000000000000000000000000000000000000000000000000000000000d6d794e616d65497353687561690000000000000000000000000000000000000000',
  r: '0x5a84ed82cf5615343d83a0ebe6ebd57934628d87ea758613f524e02ccda49523',
  s: '0x00917a7e92dc21ca2528c93a46f150dab986805cf82e521580a68f5525a0c61c',
  v: 160038,
  creates: null,
  chainId: 80001,
  wait: [Function (anonymous)]
}
tokenWalkerName 1,name:  myNameIsShuai
name:  Walker
symbol:  WAT
```

贡献值测试

```
npx hardhat run scripts/check-WalkerNFT.js --network matic
balanceOf,Ownend NFT number:  1
ownerOf 1,Owner: 0x0bE9d4386f58022c27B0Fb035c46de7c5800938F
tokenWalkerName 1,name:  myNameIsShuai
```

```
walkerContribution 1 is:  BigNumber { _hex: '0x00', _isBigNumber: true }
_setWalkerContribution 1 set:  {
  hash: '0xea699a7d9523c01fadf1e99761bf9e885b57e0e72f2c653569464a76be3b6126',
  type: 0,
  accessList: null,
  blockHash: null,
  blockNumber: null,
  transactionIndex: null,
  confirmations: 0,
  from: '0x0bE9d4386f58022c27B0Fb035c46de7c5800938F',
  gasPrice: BigNumber { _hex: '0x014c5cb407', _isBigNumber: true },
  gasLimit: BigNumber { _hex: '0xac80', _isBigNumber: true },
  to: '0x238e289247CD6122EdCd65216A9b8bE4e8f963BA',
  value: BigNumber { _hex: '0x00', _isBigNumber: true },
  nonce: 11,
  data:
'0x820bfa210000000000000000000000000000000000000000000000000000000000000001000000000000
00000000000000000000000000000000000000000000000f',
  r: '0x524a45d92e0d8d8be4a9cd245858be2d780eed37c8ba87e8571227092cba1c92',
  s: '0x47e7f71854925944cc7f79999092e1283a3ae98557e7bbd853caab28b0f774be',
  v: 160038,
  creates: null,
  chainId: 80001,
  wait: [Function (anonymous)]
}
walkerContribution 1 is:  BigNumber { _hex: '0x0f', _isBigNumber: true }
name:  Walker
symbol:  WAT
```

0x0f的转换：

```
  const co2 = await contract.walkerContribution(1);
  const co22 = (hre.ethers.BigNumber.from(co2)).toNumber()
  console.log("walkerContribution 1 is: ", co22);
```

18decimal

```
const balance = getBalance().toNumber() / 1E18

or

console.log(web3.fromWei(balance.toNumber(), "ether" ) );

const ethBalance = web3.utils.fromWei(await ethProvider.getBalance(accounts[0]),
'ether');
console.log(ethBalance);
```

=======

Quest进度

Quest等副本信息，存储于中心化网站，是使用客户端工具，众创生成并上传的数据包

因此Quest的数据结构是：用&分隔

副本id&quest-id&event-id&choicelist

范例：未做的event和未选择的关键choice?

12&Q3&E1-2-5，存档记录解释：

1》第12个副本

2》在完成Quest 3 的过程中

3》Quest 3完成了事件1，2，5（在完成中代表事件链条没有满足，需要继续探索）

1.任务可重复做？不限制，但有些任务的必须物品有数量限制

2.存档记录什么？因为任务无限制重复做，所以只记录当下在做的Quest和已完成事件（一个Quest的是一个松散事件链条）

3.一个时间内只能做一个Quest，但可以做多个Event（松散Event，不归属任何Quest）

4.做Quest或者Event，或者具体的每个Choice交互，都是为了探索世界，获得道具（未来获得经验）

合约+前端调用

人生模拟器：http://remake.solaking.com/

1.创建人物，钱包登录后获取人物信息✅

先进入page页面,给button加click事件

```
                        <div className="text-blue mt-1 ml-0" id="choice1"
                        onClick={() => {selectChoice(1)}}
                        >1.{event.data[0].choices['f']}</div>
```

然后调用的function，直接在页面写async

```
function selectChoice(num: number){
       const choiceTxt = document.getElementById("choice"+num.toString())
       const txt1 = document.getElementById("choice1")

async function approveMultiAdventure() {
       if (!chainId) return
       await approve(MULTIADVENTURE_CONTRACT[chainId])
   }
```

另外一种写法

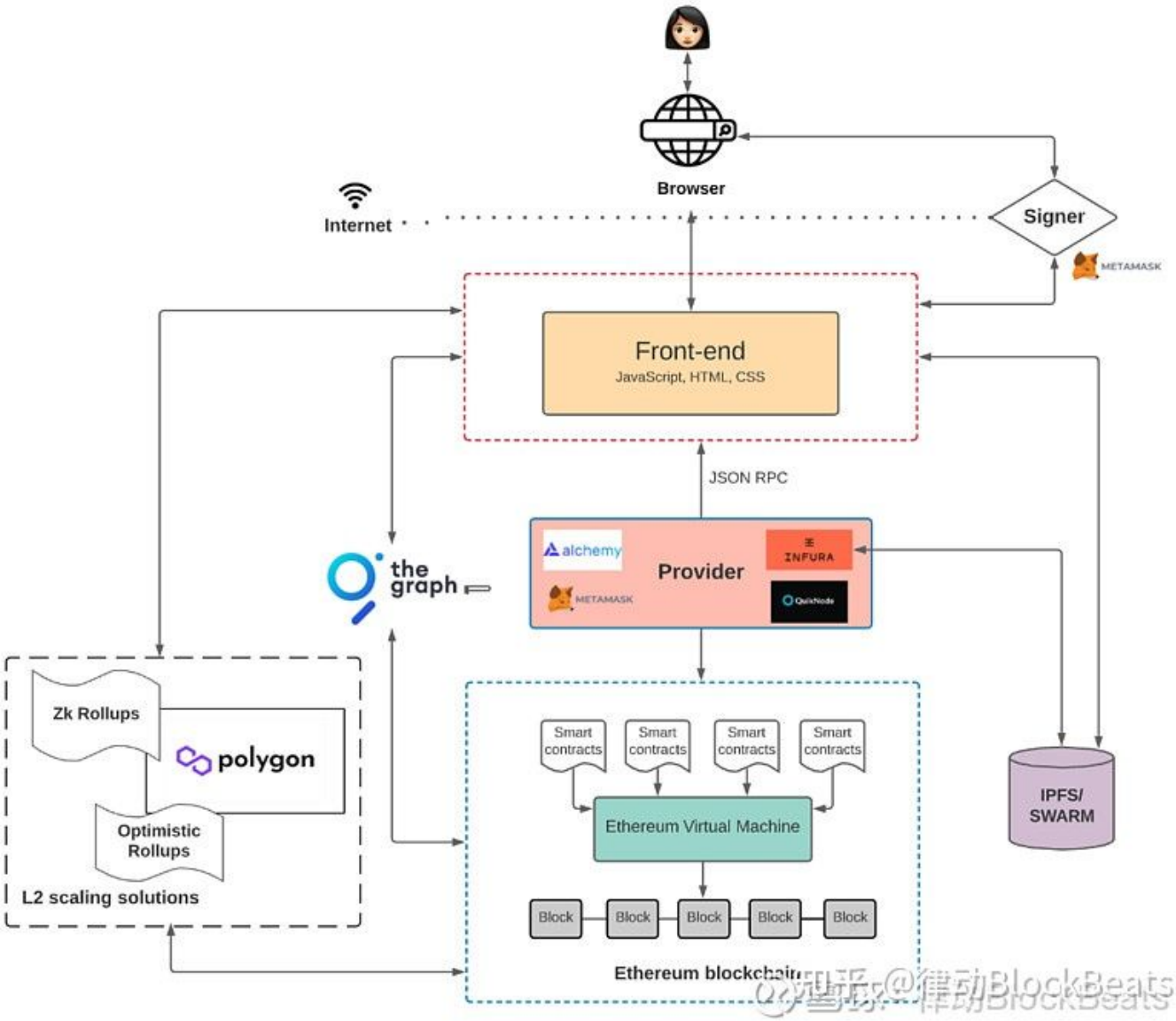
```
export方式
todo，还没研究
```

架构和技术栈准备

1.文档整理了部分

2.流程待梳理、讨论、优化

3.框图参考这个，后面优化



5.其他链上数据

https://mumbai.polygonscan.com/tx/0x37e2617dffd29fad10686d51f1ec69fbaf2ecd12ff35c95c87cf4394546ddfb8

https://mumbai.polygonscan.com/address/0xe15b04371e81ac6986683e00d9975ad0d2519b31

https://mumbai.polygonscan.com/token/0x2964409986781285204 1AFE6Eb4A5B8D67B9d590