周亚男 2020131062

完善合约代码

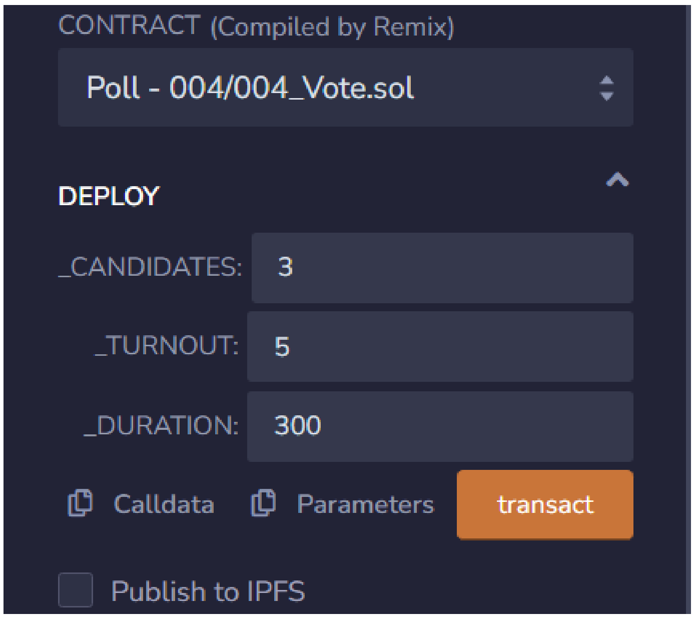
004_Vote.sol

```
// SPDX-License-Identifier: GPL-3.0


pragma solidity ^0.8.0;

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


contract Poll is Ownable {

    // 代表总的提案数

    uint8 public candidates;

    // 代表总的投票人数;

    uint public turnout;

    // 代表投票持续的秒数。

    uint public duration;

    // 记录了这一轮投票结束的时间戳

    uint public endTime;

    // 别标记目前投票是否已经开始/结束

    bool public started;

    bool public ended;

    // 表示已经投票人数。

    uint public votedNum;
    // 跟踪目前得票最高的候选人索引。
    uint8 public highestCandidate;
    // 最高候选得的票数
    uint public highestScore;
    // 存储了每个投票人针对各个提案做出选择的记录。
    mapping(address => uint8) public votedMap;
    // 存储了每个提案目前的总得票数。
    mapping(uint8 => uint) scoreMap;
    event Started();
    event Ended();

    constructor(uint8 _candidates, uint _turnout, uint
_duration) {
        candidates = _candidates;
        turnout = _turnout;
        duration = _duration;
    }

    function start() external onlyOwner {
        require(!started, "poll is over");
        require(candidates > 0, "invalid num");
        require(turnout > 0, "invalid num");
        endTime = block.timestamp + duration;

        started = true;


        emit Started();

    }


    function end() public {

        require(started, "poll not start");

        require(!ended, "poll is over");

        require(endTime <= block.timestamp, "time not
arrive");


        ended = true;


        emit Ended();

    }


    function _end() internal {

        require(started, "poll not start");

        require(!ended, "poll is over");

        ended = true;
        emit Ended();
    }

    function vote(uint8 candidateIndex) external {
        require(started, "poll not start");
        // require(!ended, "poll is over");
        require(block.timestamp < endTime, "time
invalid");
        require(votedNum <= turnout);
        require(
            candidateIndex > 0 && candidateIndex <=
candidates,
            "invalid num"
        );
        require(votedMap[msg.sender] == 0, "u have
already polled");

        votedMap[msg.sender] = candidateIndex;
        scoreMap[candidateIndex]++;
        votedNum++;
        if (scoreMap[candidateIndex] > highestScore) {
            highestCandidate = candidateIndex;
            highestScore = scoreMap[candidateIndex];
        }

        //自动触发
        if (votedNum == turnout) {
            _end();
        }
    }

    function getResult() external view returns (uint8) {
        require(ended);
        return highestCandidate;
    }
}
```

实验过程

1. 用户可以对多个提案的索引值创建投票合约；
2. 活动开始以后用户可以投票；
3. 活动结束以后用户不可以投票；
4. 每个用户仅可以投一票，记录每个用户投给了谁；
5. 记录每个提案总共得了多少票；
6. 记录最高得票提案的索引；
7. 记录最高得票提案的票数；

1. 部署



CONTRACT (Compiled by Remix)

Poll - 004/004_Vote.sol

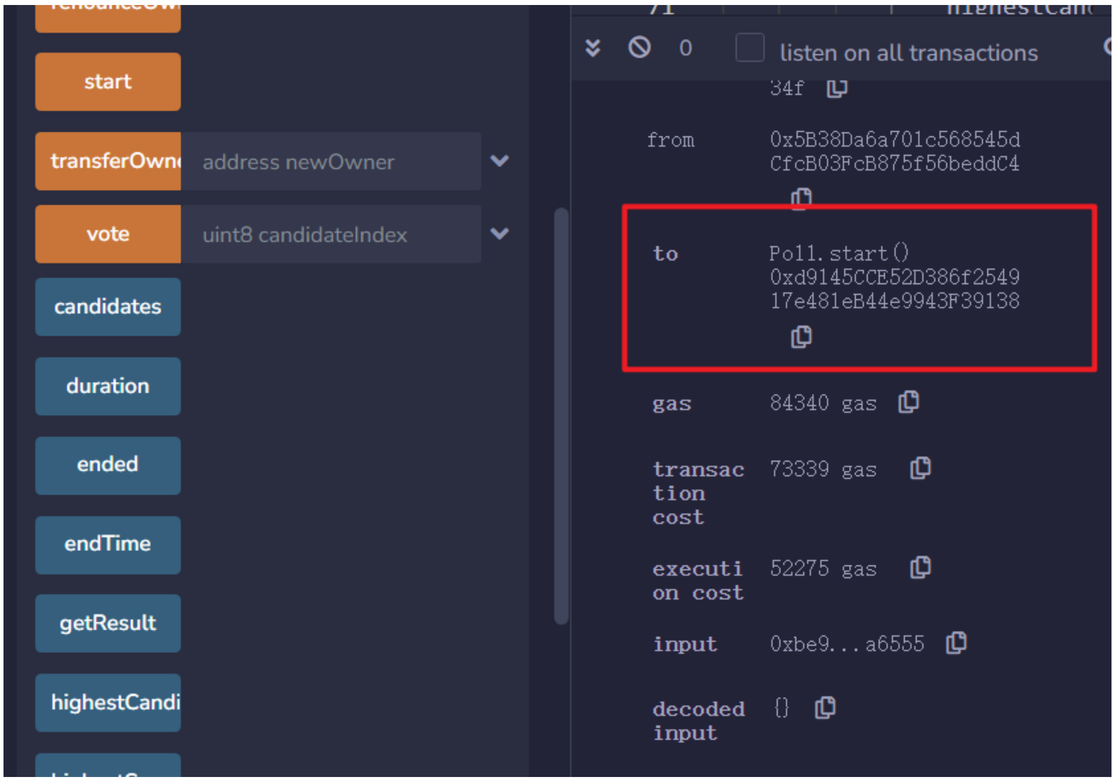
DEPLOY

_CANDIDATES: 3

_TURNOUT: 5

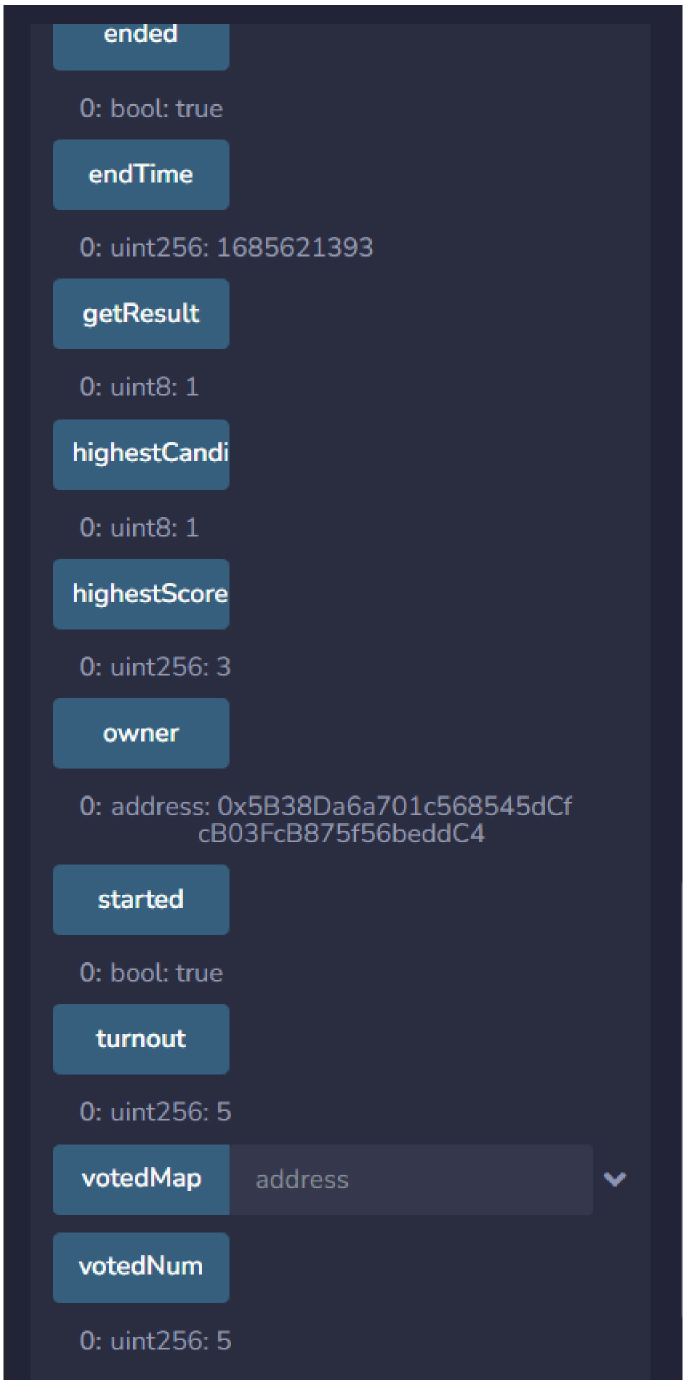
_DURATION: 300

2. 调用 start



```
34f
from              0x5B38Da6a701c568545d
                  CfcB03FcB875f56beddC4
to                Poll.start()
                  0xd9145CCE52D386f2549
                  17e481eB44e9943F39138
gas               84340 gas
transaction cost  73339 gas
execution cost    52275 gas
input             0xbe9...a6555
decoded input     {}
```

3. 使用 5 个账户，分别对 3 个提案进行投票；比如第一提案获得了最高票 3 票，



```
0: bool: true
```

```
0: uint256: 1685621393
```

```
0: uint8: 1
```

```
0: uint8: 1
```

```
0: uint256: 3
```

```
0: address: 0x5B38Da6a701c568545dCf
            cB03FcB875f56beddC4
```

```
0: bool: true
```

```
0: uint256: 5
```

```
0: uint256: 5
```