YHT:区块链游戏生态

摘要

YHT 是区块链游戏生态系统的代币，类似一种权证，持有其可享受到整个 YHT 游戏生态系统的收益分红。YHT 采用创新的游戏即挖矿的方式进行代币分发，其分发系统是一个以太坊上的彩票合约，每期分发固定总数的代币，进行投注的玩家可按照投注比例获得对应比例的代币。目前代币合约、代币分发合约均已部署到以太坊[1]上并且完全开源，也已预留开放接口，后续还将会有更多的游戏接入 YHT 生态系统。未来也将部署到更多的公链[2]上，扩展 YHT 生态系统的应用边界。

1. 简介

带有投注性质的游戏历史悠久，从公元前3000年的美索不达米亚[3]直到如今，都有大量的人对此乐此不疲[4]。但是其往往需要可信赖的中心进行仲裁判定盈亏，由于利益的牵扯，不透明、黑箱操作、甚至卷款跑路都时有发生。采用去中心化的区块链系统可有效规避这些风险，开源的智能合约能够带来最大程度的透明。

所以YHT游戏生态应运而生，其不但能通过区块链智能合约[5]解决由来已久的弊病，还将运用通证经济激活、连接整个生态。玩家参与游戏的同时，也通过游戏即挖矿的方式获得代币，持有代币就可享受整个生态系统的收益分红。

2. 代币分发

2.1 彩票合约

YHT分发系统是一个彩票合约，规则如下:

(1) 每周开奖三期，周二、周四、周六开奖(UTC+0)①;

(2) 按照投注比例分配胜率，随机出唯一获胜者;

(3) 获胜者将独得奖池的41.77%;

(4) 奖池的18%会进入下期奖池;

(5) 奖池的39%②会作为分红分配给代币YHT的持有者;

(6) 奖池的1.23%作为费率用以维持合约的运行③以及社区建设;

随机数通过oraclize[6]服务接口产生确保高度的安全性[7]。

---

① 视以太坊运行情况可能有所提前或延迟。

② 推荐人 7%返还奖励会从此部分中扣除，因而实际可能不足 39%，返还奖励最多持续 15 期。

③ 使用了 oraclize 的定时触发以及随机数生成服务，确定最终获胜者的核心函数开销与投注人数成正比，因而所消耗的 gas 可能较多。

2.2 YHT合约

通过投注每期彩票，即可获得 YHT，规则如下

(1) 每期固定总数271828[④]个；

(2) 每期结束时按照当前投注比例进行分配[⑤]；

(3) 每314期产量减半[⑥]；

(4) 每期额外发行3.82%的代币赠予创始团队用于支持YHT游戏生态的持续发展。

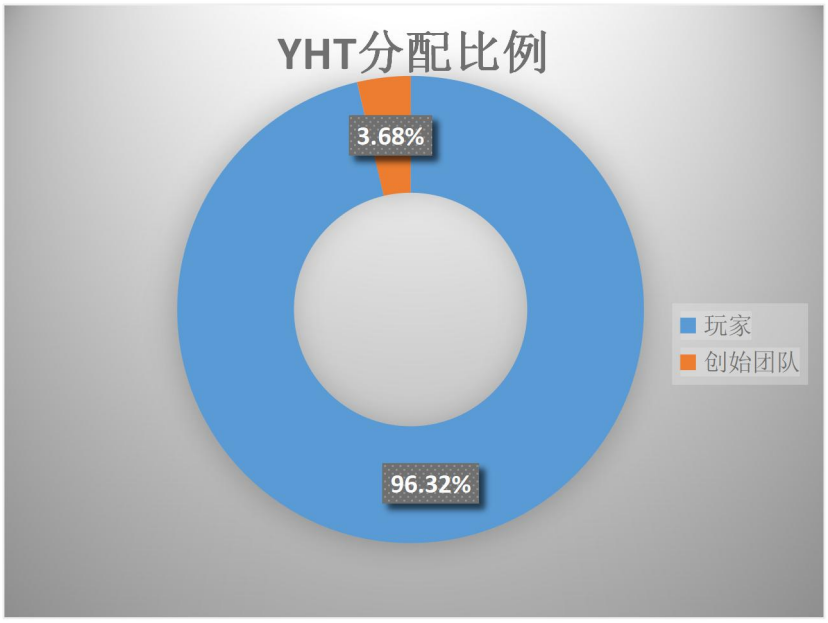

图 1 YHT 分配比例示意图

3. 合约开放接口

YHT合约开放接口声明如下:

```
Contract YHTOpenInterface {
   function transferExtraEarnings(address addr) external payable;
   function transferBonusEarnings() external payable returns(uint256);
}
```

transferExtraEarnings函数：外部合约可通过调用此函数向指定用户地址存入收益。例如某游戏玩法可通过此函数将“大奖”奖励分配给最终胜利者。

transferBonusEarnings函数：外部合约可通过调用此函数将收益分配给所有YHT持有者。例如某游戏玩法可将其累计费率收益通过调用此函数进行分红分配。

注：彩票合约就是通过这两个接口完成大奖分配以及分红发放。

---

④ 省略了小数部分，实际完整是 271828.182845904523536028。
⑤ 先分发 YHT 然后计算分红。
⑥ 每期分配产量减半至小于 100 个就会停止，即 YHT 分发完毕。

4. 快照系统

代币持有人的地址数最终将变得十分巨大，使得发送分红的合约负担所有的调用开销几乎不再可行，因此我们开发了快照系统，记录YHT持有人每个奖励段的代币持有情况。任何代币数量发生变化时我们都会记录快照，通过快照可快速推断玩家实际累计分红收益，在玩家需要提取收益时，才会计算、合并分红所得。这使得发送分红的接口调用维持在常量开销，所以大量、频繁地分红分发才成为可能，未来才能够接入更多的游戏。

核心数据结构以及部分算法如下：

```
struct BalanceSnapshot {
   uint256 balance;                    // balance of snapshot
   uint256 bonusIdBegin;               // begin of bonusId
   uint256 bonusIdEnd;                  // end of bonusId
}

struct User {
   ....
   ....
   BalanceSnapshot[] snapshots;    // the balance snapshot array
   uint256 snapshotsLength;           // the length of balance snapshot array
}

/**
 * @dev record a snapshot of balance
 * with the bonuses information can accurately calculate the earnings
 */
function balanceSnapshot(address addr, uint256 bonusRoundId) private {
   uint256 currentBalance = balances[addr];
   User storage user = users_[addr];
   if (user.snapshotsLength == 0) {
      user.snapshotsLength = 1;
      user.snapshots.push(BalanceSnapshot(currentBalance, bonusRoundId, 0));
   } else {
      BalanceSnapshot storage lastSnapshot = user.snapshots[user.snapshotsLength - 1];
      assert(lastSnapshot.bonusIdEnd == 0);

      // same as last record point just updated balance
      if (lastSnapshot.bonusIdBegin == bonusRoundId) {
         lastSnapshot.balance = currentBalance;
```

```
        } else {
            assert(lastSnapshot.bonusIdBegin < bonusRoundId);

            // if this snapshot is not the same as the last time, automatically merges part of the earnings
            if (bonusRoundId - lastSnapshot.bonusIdBegin < kAutoCombineBonusesCount) {
                uint256 amount = computeRoundBonuses(lastSnapshot.bonusIdBegin, bonusRoundId, lastSnapshot.balance);
                user.bonusEarnings = user.bonusEarnings.add(amount);

                lastSnapshot.balance = currentBalance;
                lastSnapshot.bonusIdBegin = bonusRoundId;
                lastSnapshot.bonusIdEnd = 0;
            } else {
                lastSnapshot.bonusIdEnd = bonusRoundId;
                if (user.snapshotsLength == user.snapshots.length) {
                    user.snapshots.length += 1;
                }
                user.snapshots[user.snapshotsLength++] = BalanceSnapshot(currentBalance, bonusRo
        undId, 0);
            }
        }
    }
}

/**
  * @dev get bonuses
  * @param begin begin bonusId
  * @param end end bonusId
  * @param balance the balance in the round
  * Not use SafeMath, it is core loop, not use SafeMath will be saved 20% gas
  */
function computeRoundBonuses(uint256 begin, uint256 end, uint256 balance) view private returns(uint256) {
    require(begin != 0);
    require(end != 0);

    uint256 amount = 0;
    while (begin < end) {
```

```
        uint256 value = balance * bonuses_[begin].payment / bonuses_[begin].currentTotalSupply;
        amount += value;
        ++begin;
    }
    return amount;
}

/**
* @dev compute snapshot bonuses
*/
function computeSnapshotBonuses(User storage user, uint256 lastBonusRoundId) view private returns(uint256) {
    uint256 amount = 0;
    uint256 length = user.snapshotsLength;
    for (uint256 i = 0; i < length; ++i) {
        uint256 value = computeRoundBonuses(
        user.snapshots[i].bonusIdBegin,
        i < length - 1 ? user.snapshots[i].bonusIdEnd : lastBonusRoundId,
        user.snapshots[i].balance);
        amount = amount.add(value);
    }
    return amount;
}
```

5. 发展、运营

YHT社区采用开源软件去中心化协作的方式进行迭代开发，以及后续运营，不会创立公司实体，不存在中心化办公，因而可以保持极低的开发与运营成本。欢迎更多有兴趣的伙伴参与进来，一起构建、完善YHT社区。

6. 结论

我们在区块链构建了去中心化的游戏投注系统，并采用投注即挖矿的方式分发YHT代币，使用YHT代币激励连接游戏参与者。设计并预留了合约的开放接口，使得更多的游戏可以接入YHT游戏生态。开发了快照系统，解决以后大量游戏接入带来的巨量、频繁的分红分配问题。采用开源软件去中心化协作的方式进行社区的后续开发与运营，最终构建一个持续、稳定的区块链游戏生态。

附录：合约开源地址

https://etherscan.io/address/0xe7FE07E428a1Fa4cDEBacD6f5dF8DD053966c4a8#code

https://etherscan.io/address/0x2D04c7051112C47Ee74A41C723F791b499Aa6B1a#code

参考文献

[1] Vitalik Buterin. Ethereum: a next generation smart contract and decentralized application platform (2013). URL {https://github.com/ethereum/wiki/wiki/White-Paper}, 2018.

[2] EOS.IO Technical White Paper v2. URL {https://github.com/EOSIO/Documentation/blob/master/TechnicalWhitePaper.md},2018.

[3] Schwartz, David (January 7, 2013). Roll The Bones: The History of Gambling. Winchester Books. ISBN 978-0615847788.

[4] Wikipedia contributors. "Problem gambling." Wikipedia, The Free Encyclopedia. Wikipedia, The Free Encyclopedia, 12 Jul. 2018. Web. 25 Aug. 2018.

[5] Wikipedia contributors. (2018, July 14). Smart contract. In Wikipedia, The Free Encyclopedia. Retrieved 13:42, August 25, 2018, from https://en.wikipedia.org/w/index.php?title=Smart_contract&oldid=850226891.

[6] Oraclize. The leading oracle service for smart contracts and blockchain applications. URL {https://docs.oraclize.it/#home}, 2018.

[7] Oraclize. A Scalable Architecture for On-Demand, Untrusted Delivery of Entropy. URL {http://www.oraclize.it/papers/random_datasource-rev1.pdf}, 2018.