Connect Four

connecting blocks on the blockchain

The Concept

Let's create a simple turn-based game for 2 players.

- Connect Four
- Front-End necessary

Let them play for ETH and we take a cut.

- Connect four is a solved game!
- Restrictions
- Incentives

$$\begin{bmatrix} 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 2 & 2 & 2 & 2 & 0 & 0 \\ 1 & 2 & 1 & 1 & 2 & 0 & 0 \\ 1 & 2 & 1 & 2 & 2 & 0 & 0 \\ 2 & 1 & 2 & 1 & 1 & 0 & 0 \\ 2 & 1 & 1 & 1 & 2 & 0 & 0 \end{bmatrix}$$

The Concept

Let’s create a simple turn-based game for 2 players.

- Connect Four
- Front-End necessary

Let them play for ETH and we take a cut.

- Connect four is a solved game!
- Restrictions
- Incentives

Adjustments

Solved game:

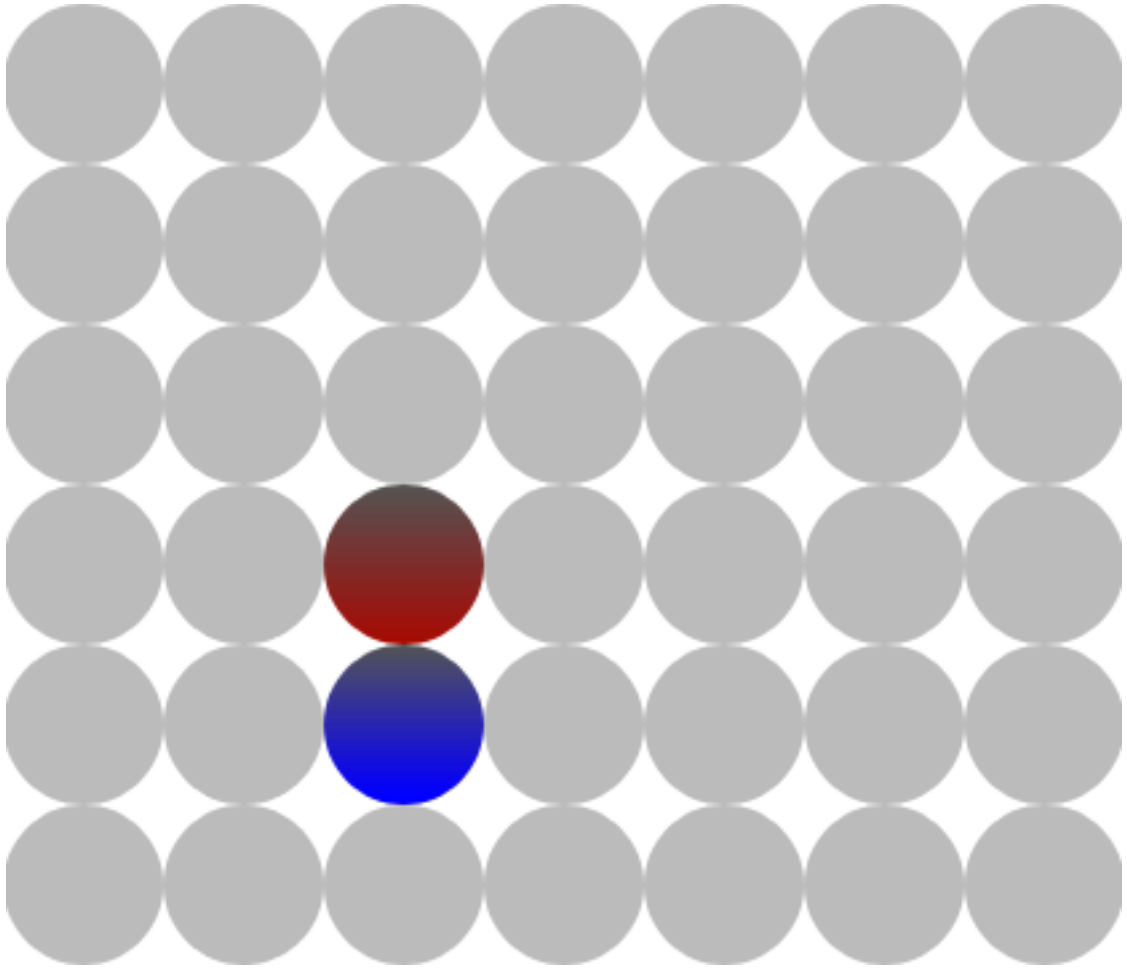
- Modify the rules and make it harder!

Restrictions:

- Bet can be 0.01 to 5.00 ETH

Incentives:

- Incentive for losing player to give up



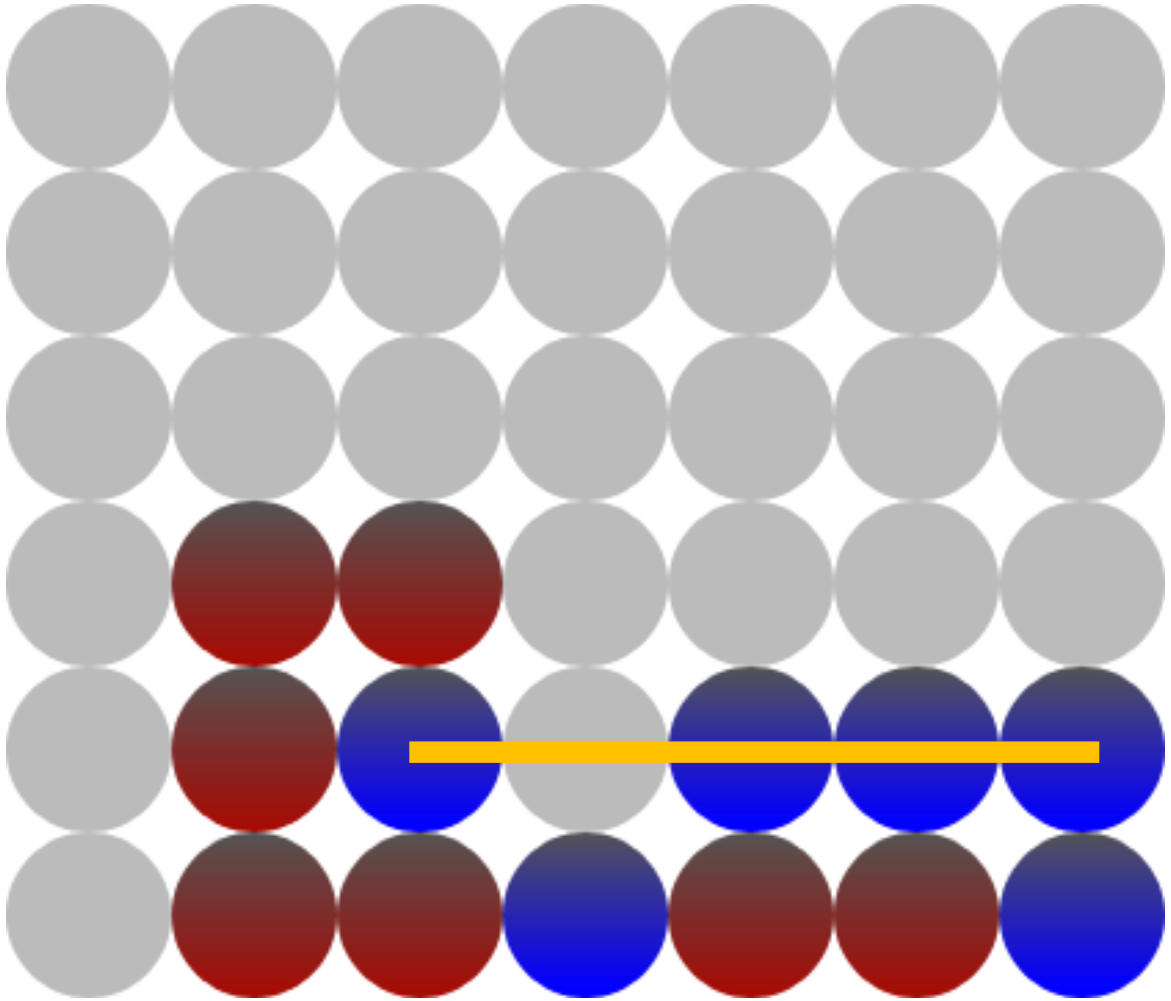
The Smart Contract

What the smart contract does:

- Provides the rule set and enforces it
- Checks for equal bets
- **Sets up the game (PRNG)**
- **Most challenging function: has a player won the game?**
- Payout

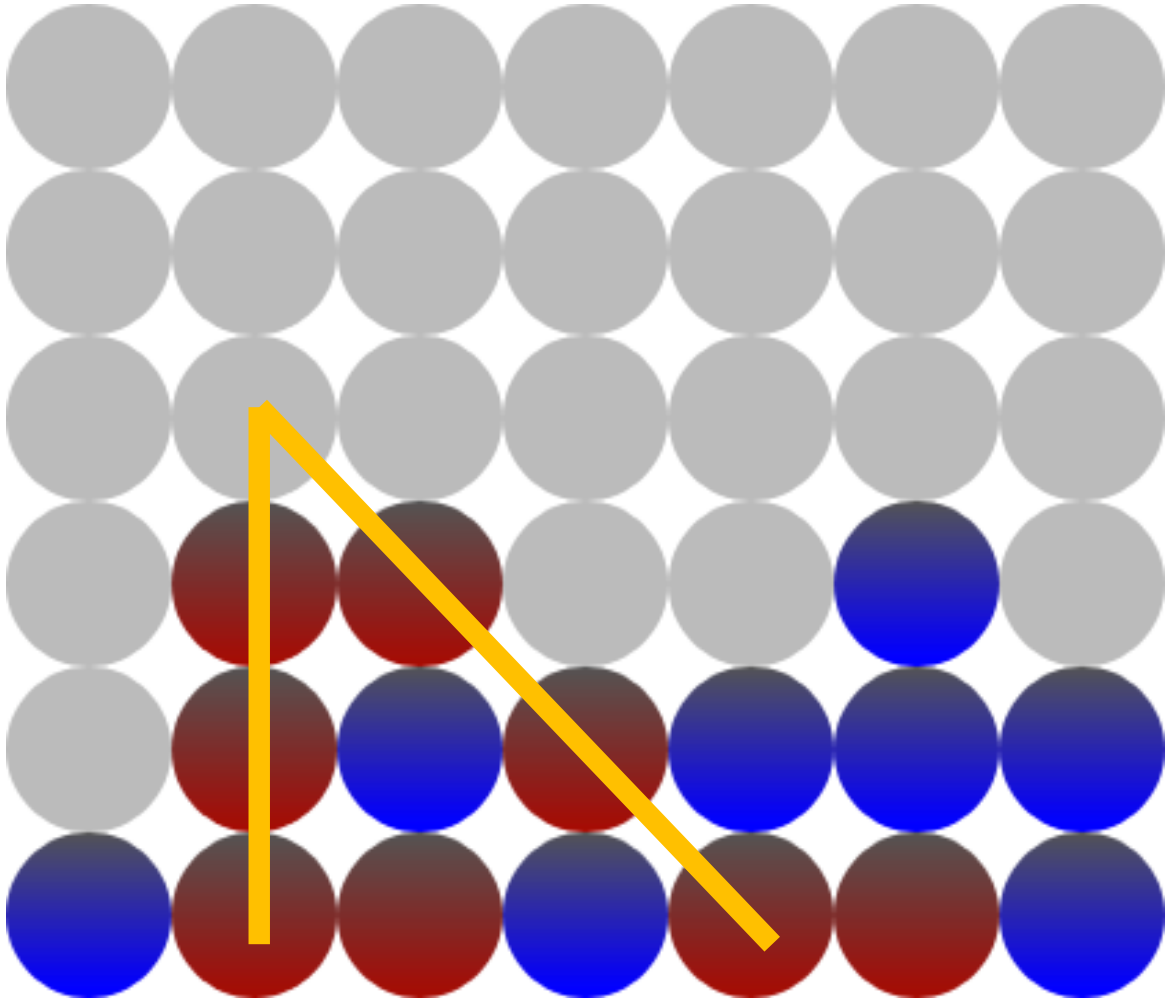
The Rules in General

- Player 1 sets bet (0.01 – 5.00 ETH)
- Player 2 matches the bet
- Player 2 starts the game with another transaction (PRNG)
- The game randomly assigns one stone per player and determines the starting player
- Goal: connect 4 stones of your color in a row, column or diagonal
- When the time per turn runs out, any party can end the game. The idle player loses!
- Giving up gives the losing player a small amount of ETH
- The winner takes it all! (Well, most of it…)



The Rules in General

- Player 1 sets bet (0.01 – 5.00 ETH)
- Player 2 matches the bet
- Player 2 starts the game with another transaction (PRNG)
- The game randomly assigns one stone per player and determines the starting player
- Goal: connect 4 stones of your color in a row, column or diagonal
- When the time per turn runs out, any party can end the game. The idle player loses!
- Giving up gives the losing player a small amount of ETH
- The winner takes it all! (Well, most of it…)



Pseudo-Random Number Generator

- Don’t use block variables of a past block
- Block variables of current block?
  - Block hash? …not possible
  - Others? …possible but not safe
- Our solution:
  - Two-step approach

Pseudo-Random Number Generator

$$x_{n+1} = (a \times x_n + c) \bmod m$$

```
function rand() private returns (uint256) {
    // use randomSeed as X_0
    // X_(n+1) = a*X_n + c mod m
    // a = 48271, c = 0, m = 2**32-1 (=4294967295)
    currentRandomNumber = (48271 * currentRandomNumber) % 4294967295;
    return currentRandomNumber;
}
function setUpGame() private {
    currentRandomNumber = uint(player2JoinBlockHash ^ bytes32(uint256(startCoinbase) << 96));
    //nextRandomNumber = rand() % n
```

Function: checkVictoryCondition

```
function checkVictoryCondition(uint8 _col, uint8 _row) private {
(...)
    // left side
    currentStoneOwner = activePlayer;
    currentCol = _col;
    while (currentStoneOwner == activePlayer && currentCol >= 0) {
        victoryPoints++;
        if (currentCol > 0) {
            currentCol--;
            currentStoneOwner = grid[currentCol][_row];
        } else
            break;
    }
(...)
```

Security Problems

ATTENTION:

- Set bet = 0 before paying out money in the code
- Otherwise, choosing the right gas limit could be leveraged to outwit the smart contract
- Attacker could withdraw funds over and over again
- See The DAO

Let’s Play

on 4blocks.ch