Overwatch Tournament

BY CONTRACT CRAFTERS

BLSC Project - Yann Goulvestre - Ahmad Harkous - Clément Thollet

What is it ?

- This is a 1 vs 1 Overwatch tournament
- 8 players are playing
- Each player pays an entry fee
- There are quarter finals, semi-finals and a final
- The entry fee is given to the two finalists

Why did we invent this tournament ?

- Overwatch does not currently allow to play a real game in a tournament like this. Creating this kind of tournament could also attract more players who want to discover a new game mode.

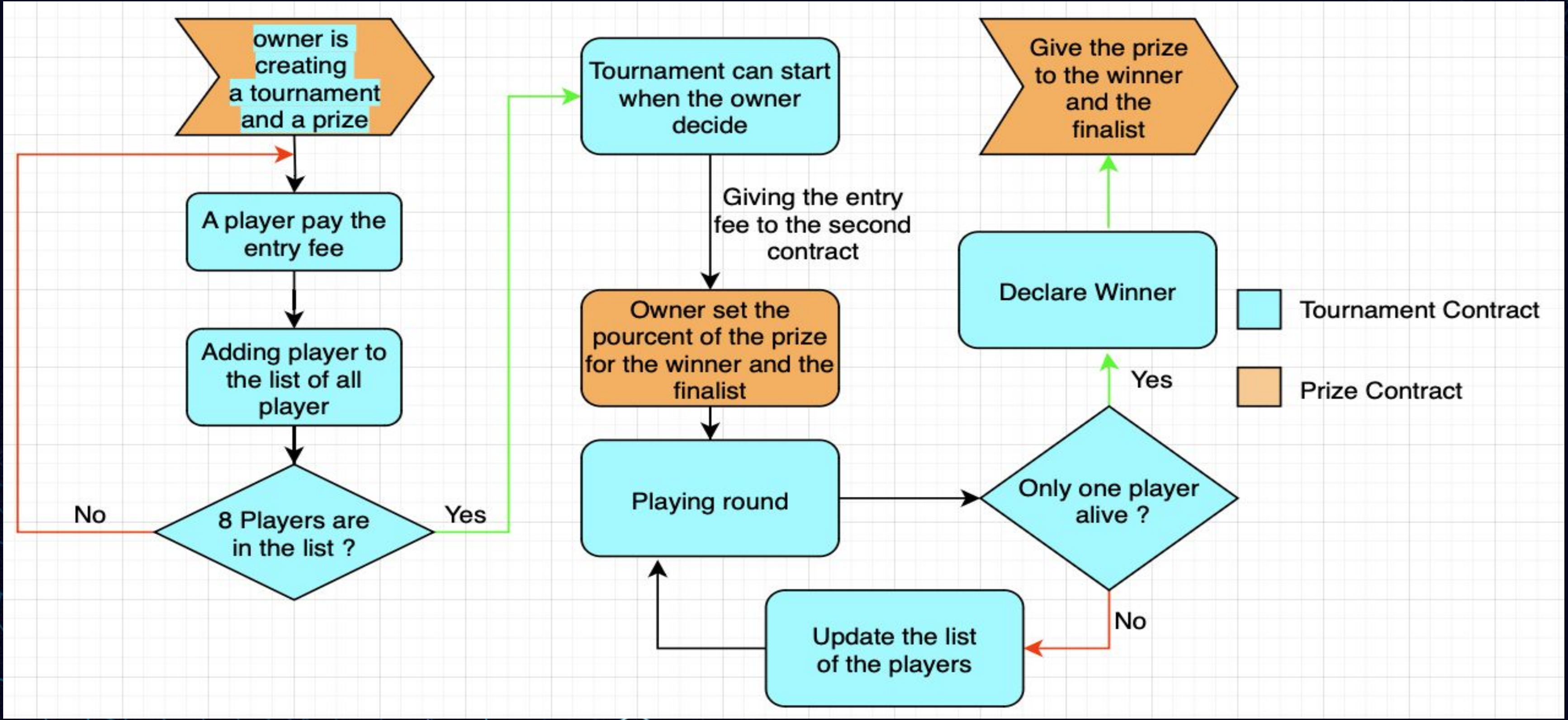
- There are more and more scam in this game, and creating a system of smart contract could relieve the players who would be certain to not be scammed.

- Creating a system of reward could attract more and more people, that could be good for the game and increase the number of players.

How do the smart contracts work ?

- The TournamentContract keeps track of players, tournament status, entry fee, and owner.
- 8 players join the tournament.
- The admin starts the tournament and transfers prize money to the PrizeContract.
- The admin sets the prize money in the PrizeContract.
- Players are updated by the admin during the tournament in the TournamentContract.
- The winner is declared in the TournamentContract.
- The admin distributes the prize money to the winner and the finalist in the PrizeContract and declare tournament as finished.
- The contracts enforce various conditions and assertions to ensure fairness and security.

Link between the two contracts


owner is creating a tournament and a prize
A player pay the entry fee
Adding player to the list of all player
8 Players are in the list ?
No
Yes
Tournament can start when the owner decide
Giving the entry fee to the second contract
Owner set the pourcent of the prize for the winner and the finalist
Playing round
Only one player alive ?
Yes
No
Update the list of the players
Declare Winner
Give the prize to the winner and the finalist
Tournament Contract
Prize Contract

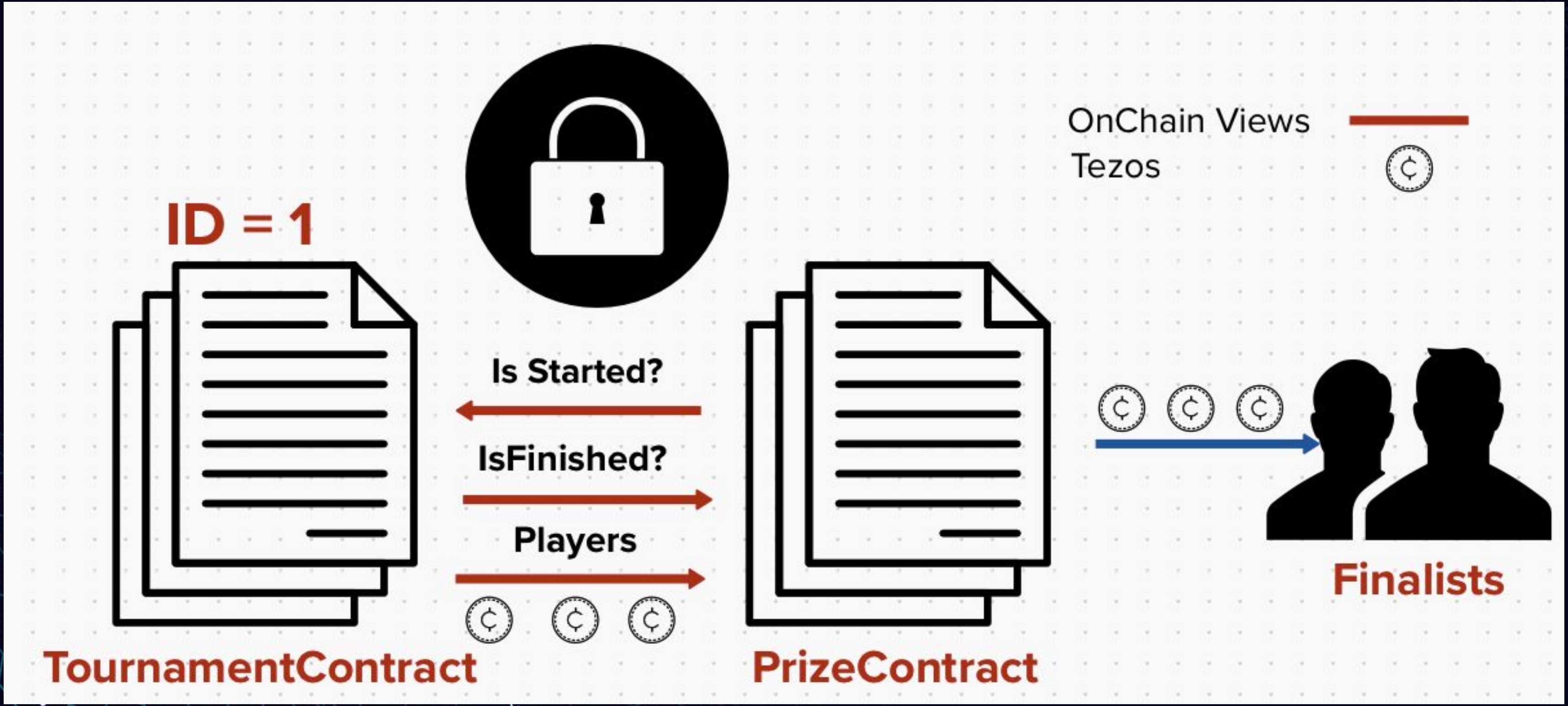
Why we used 2 SmartContracts/One Tournament ?

- Using 2 smartcontracts to handle many tournaments require more complex logic and can create a lot of flaws .

- In our approach, each contract represents a specific tournament and maintains its own storage for tracking tournament-related data. By adopting this approach, we achieve better isolation and modularity, which facilitates easier management , avoid cheating flaws and updates for individual tournaments

- However, it should be noted that deploying a new contract for each tournament incurs higher deployment costs in terms of gas fees. Additionally, maintaining multiple contracts may require additional effort for upkeep and management.

Avoiding Cheating Flaws



Avoiding Security Flaws

- Assertions are made to make sure of the values that are in the contracts.
- The command *sp.source* that causes security problems is replaced by the command *sp.sender*.
- We verify that the *sp.sender* is not the *self.data.owner* in order to avoid the owner to buy from himself.
- We check that the author cannot block any sale without more than one *sp.send* for two destinations.

How using the blockchain has benefits for the project ?

- The blockchain allow to avoid any kind of cheating.
- Players cannot join the tournament without paying and the tournament cannot be started without the correct amount of money and player.
- The smart contracts are immutable, so when players have paid, they cannot change their payment.
- At the end, the money is given to the right players.
- The smarts contracts are self-executing and operate automatically according to predefined rules. That reduces the risk of cheating.
- Players are sure to win the amount of money that is promised to the winner.

How the smart contracts would integrate with other components as part of a dApp?

To integrate the TournamentContract and PrizeContract with other components as part of a dApp, we would typically follow these steps:

1. Set up the front-end of the dApp: This involves creating a user interface (UI) using HTML, CSS, and JavaScript. The UI will interact with the smart contracts and display relevant information to the user.

2. Connect to the Tezos blockchain: We Use Taquito library to connect to the Tezos blockchain from the dApp.

3. Fetch data from the smart contracts: We use also Taquito to fetch data from the TournamentContract and PrizeContract, such as player information, tournament status,owner, prize money, and winner details. Then we display this data in the UI to provide real-time information to the users.

How the smart contracts would integrate with other components as part of a dApp?

4. Allow users to join the tournament: We Implement a feature in the dApp's UI that allows users to join the tournament by calling the join_tournament entry point of the TournamentContract. This will trigger a transaction on the Tezos blockchain and update the state of the contract.

5. We integrate the remaining entry points of the contracts to the UI, such as starting the tournament and setting prize money, updating the tournament progress, and declaring the winner and distributing the prize money.

6. Display transaction outcomes: After interacting with the smart contracts, we display the outcomes of the transactions to the users. This includes providing confirmation messages, displaying transaction hashes, and updating UI elements to reflect the updated state of the contracts.

What are our tests ?

- Our tests are made with *scenario.verify* in the “Testing” section.
- Each transaction of the tournament are tested.
- We check that the balance of the contract is correct and that no money is stolen during any transaction.
- We verify that the number of player is good at any moment of the tournament.
- We make sure that the winner and the finalists are the right players.
- All the tests are realized to avoid flaws.

Link to the SmartPy code :

If you have any questions, do not hesitate to contact us !