#ourPlace

Whitepaper for ourPlace - The Ethereum Based Collaborative Canvas

Thanks: Thanks to Stephen Grider for making his smart contract course available on Udemy, and for every other person who has contributed even the tiniest bit to troubleshooting and tutorials.

You Can Do It: This is my first go of it, so if you have programmed a little bit in another language, then you could probably learn how and then build ourPlace in about 2-3 months. If you have wanted to learn smart contracts just do it!

Ethereum: I truly believe Ethereum has a chance at creating some really interesting, society changing, cooperative infrastructure. That being said: What is built on top of Ethereum... not necessarily decentralized. Very excited for the future of ETH!

###Idea:

The basic idea is that users can change the color of a pixel on a public canvas for a small fee. This idea is a combination of the r/Place Reddit experiment in 2017, and the 'Million Dollar Homepage' conceived in 2005. In r/Place users were limited to how frequently they could change a pixel, and in the Million Dollar Homepage each pixel could be bought and owned; they were both a 1000x1000 pixel canvas.

At ourPlace you can place one pixel per block, pick any Hex #RRGGBB color, and choose from one of the two canvases. The two canvases exist because it seemed like many of the creations were 'flag shaped' or could fit on a flag shaped canvas. The small flag shaped canvas (120x72) would allow for a collaborative theme to the ourPlace landing page, and the medium canvas (500x500) is available for the collage-scape – similar to the originals. In the future I hope to release a large/traditional canvas (1000x1000).

###Challenges:

1. Moderation - Any time users can make their own pictures there is potential for abuse. Every major social media company is struggling with moderation. There is probably a lot of room for more MAAS companies. At first I was worried I would have to monitor the canvases constantly and think more about the 'Hot Dog-Not Hot Dog' problem. However I have come to the decision that I think it should be all right because of these reasons:
    a. As administrator I have a clearing function capable of wiping blocks of pixels. A down side is the community trust factor by having that in. All I can do is ask for trust that I will use my power only for good.
    b. Pixelization will make it hard to create a truly disturbing image. Hateful images very possible but disturbing seems like high effort.
    c. Cost per pixel and transaction fees will make individual efforts expensive.
2. Canvas Size - It has been a challenge realizing that there is not an easy way to work with large arrays, both within the smart contract and externally making calls upon the data. Some things I have learned:
    a. For ourPlace I ended up creating an empty fixed-size array in the constructor and populating it with each transaction. This is definitely the best and only way to make this contract work. But setting a default value has to be done on the front end.
    b. The maximum bytes9 array size I have been able to call from my contract thus far is approximately 250,000. Theoretically you should be able to read any size of array but I have not been able to. (Related but seperate issue: <https://github.com/paritytech/parity-ethereum/issues/6293>)
    Possible solutions:
        i. Create my own Ethereum node and make calls to my node. Versus currently using Infura.
        ii. Maybe my promises making contract calls are just timing out (I think I ruled this out though).

3. Restricting to one transaction/address/block
    a. Ended up mapping addresses to block.number in the transaction
    b. Other ideas
        i. A way to observe pending transactions? Notes: Internal Transactions, subscribe to events with web3
        ii. Wait two blocks/delay block execution to check and update pixels? Notes: Does not solve problem.
        iii. Record nonce? Notes: Nope. No control over increment.
        iv. Track accounts into a mapping? Notes: This link is a bad example, not possible to clear mapping. <https://ethereum.stackexchange.com/questions/7859/contracts-receiving-multiple-identical-transactions>
        v. Store addresses in an array and clear using a timer? Notes: Increasingly cost more to run the contract until cleared. Clear after X contracts or blocks.
4. Checking Hex codes
    a. Found a regex-to-solidity tool called Solregex <https://github.com/gnidan/solregex>
    b. Decided to compile the hex check as a separate library, must adjust byte code <https://ethereum.stackexchange.com/questions/6927/what-are-the-steps-to-compile-and-deploy-a-library-in-solidity/6951>

###Other Random Thoughts:

1. Bug/Contact System
    a. I considered adding a bug/notification smart contract. Might look into it in the future as an OS project, it seems like something that should have been done by now.
2. Cryptographic messages
    a. You could probably do some cold war era secret messaging by using pixels and a pre-defined code. For example if pixels (1,1)(1,5)(30,42)(30,43) each have hex codes #rrggbbaa, #rrggbbaa, #rrggbbaa, #rrggbbaa. Using a hex to string converter you could send messages like victoriast9pm (766963746f726961737439706d). You would need to send the message at off peak times in duplicate or on unpopular pixels to be most reliable. Obviously there's better ways to send secret messages but whatever.