A Decentralized, Collaborative Pixel Art Platform on Ethereum

Eric B Martin
Department of Computer Science
Colorado State University
Fort Collins, CO, USA
Email: ebmartin@colostate.edu

Federico Larrieu
Department of Computer Science
Colorado State University
Fort Collins, CO, USA
Email: flarrieu@colostate.edu

Victor Berggren
Department of Computer Science
Colorado State University
Fort Collins, CO, USA
Email: victor.berggren@colostate.edu

Abstract

Ethr/place is a decentralized, collaborative pixel art platform built on the Ethereum blockchain, inspired by Reddit's r/place project. Ethr/place leverages the benefits of blockchain technology, such as decentralization, immutability, and smart contract capabilities, to create a transparent, secure, and engaging platform for users to collaborate on digital art. We hope to address the challenges encountered in the original r/place project, such as bot interference by implementing additional logic and measures to ensure fairness and genuine user participation. Furthermore, Ethr/place introduces novel incentive mechanisms and potential monetization strategies, including a referral program and revenue-sharing model, to reward users for their contributions and foster active community engagement. This paper outlines the design, implementation, and key features of Ethr/place, highlighting its advantages over the traditional r/place collaborative art platform and discussing potential challenges and future developments. Through the integration of blockchain technology and community-driven governance, Ethr/place seeks to establish itself as a fun reimagination of the original r/place success.

CCS Concepts

- Human-centered-computing
  - Collaborative and social computing systems and tools
- Software and its engineering
  - Smart contracts
- Decentralized blockchain
  - Layer 2 solutions
  - Ethereum
- Security and privacy
  - Decentralized systems security
- Applied computing
  - Digital art
  - NFT

Keywords

Ethereum, DApps, Collaborative Art, Blockchain, Layer 2 Solutions, Smart Contracts, Community Engagement, Tokenization, Pixel Art

I. Introduction

In April 2017, Reddit introduced a unique collaborative art project called r/place as part of their annual April Fools' Day event. This project captured the imagination of millions of users, who worked together to color individual pixels on a large canvas with time constraints. The result was a diverse and fascinating mosaic of images, logos, and symbols, representing various communities, interests, and countries. The r/place experiment demonstrated the immense potential of collective creativity and the power of community-driven collaboration. Building upon the captivating foundation laid by r/place, this paper presents Ethr/place, a decentralized version of the original project that leverages the benefits of blockchain technology. Ethr/place aims to create a more transparent, secure, and engaging platform for users to collaborate on digital art by harnessing the Ethereum blockchain's decentralization, immutability, and smart contract capabilities. In addition to replicating the core features of r/place, Ethr/place addresses some of the challenges encountered in the original project, such as bot interference and the influencer effect, by implementing additional logic and measures to ensure fairness and genuine user participation. Furthermore, Ethr/place introduces novel incentive mechanisms and potential monetization strategies that reward users for their contributions and foster active community engagement. This paper outlines the design, implementation, and key features of Ethr/place, highlighting its advantages over traditional collaborative art platforms and discussing potential use cases and future developments. Through the integration of blockchain technology and

community-driven governance, Ethr/place seeks to establish itself as a leading platform for decentralized digital art creation and collaboration in the era of Web 3.0.

II. Product Description

Ethr/place is an innovative, decentralized platform built on the Ethereum blockchain that enables users to create and collaborate on pixel art in real-time. Inspired by Reddit's r/place project, Ethr/place aims to harness the power of blockchain technology to provide a secure, transparent, and engaging experience for users worldwide. The platform leverages a Layer 2 solution **TBD (Solana, Polygon, Arbitrum?)** to minimize transaction fees and increase scalability, ensuring a smooth and cost-effective user experience. Smart contracts are employed to manage pixel placement, user interactions, and **potential** token transactions, providing a secure and decentralized environment for creativity and collaboration. To incentivize user participation and enhance the platform's utility, Ethr/place will implement a **referral program** that rewards users for inviting friends or other users to join the platform. Referrers receive a percentage of their referrals' pixel placement fees as a monetary incentive. Alongside this, Ethr/place will host an auction for the final produced NFT will employ a **revenue-sharing model** that distributes a portion of the revenue received from the NFT to users based on their contributions, such as the number of pixels placed. This model ensures that users are fairly compensated for their efforts and encourages active participation in the platform's development and decision-making processes. Ethr/place aims to redefine the collaborative art experience by combining the benefits of decentralization, blockchain technology, and community-driven governance. By fostering creativity, collaboration, and use engagement, Ethr/place seeks to establish itself as a alternative platform to the now defunct Reddit r/place.

II. Motivation

The r/place subreddit served as a unique and captivating collaborative project and social experiment, in which participants were given the opportunity to color a single pixel on a large canvas with time constraints. This exercise aimed to bring together individuals to create a contemporary artwork piece, demonstrating the power of collective creativity and cooperation.

Although the r/place project showcased several aspects akin to a ledger, its centralized nature left room for improvement. By eliminating the central authority and integrating the project with blockchain technology, the resulting platform would transform into a truly social experiment. This new approach would not only strengthen the collaborative aspect of the project, but also ensure that the resulting artwork remains immutable and securely archived on the blockchain.

The motivation behind the Ethr/place project is to build upon the foundation set by r/place, enhancing the original concept by leveraging the benefits of decentralization, transparency, and immutability provided by the Ethereum blockchain. By doing so, Ethr/place aims to foster an even more engaging and collaborative environment, creating a platform that stands as a testament to the power of community-driven art in the digital age.

The original r/place project faced certain challenges, such as the use of bots and the disproportionate influence of certain participants, which could potentially disrupt the collaborative nature of the platform. In order to mitigate these issues and promote a more equitable and engaging environment, Ethr/place can incorporate additional logic and measures. By implementing CAPTCHAs or user authentication, we can help ensure that only genuine users contribute to the platform, hopefully restricting automated actions so that the project can maintain its core focus on human collaboration and creativity.

III. Project Complexity/Challenges

The Ethr/place project, being a decentralized version of the original r/place built on the Ethereum blockchain presents certain complexities in its design, development, and implementation. These complexities stem from integrating blockchain technology, addressing the challenges encountered in the original project, and incorporating novel features to enhance user experience and engagement.

1. Designing and implementing smart contracts: Developing secure and efficient smart contracts to manage pixel placement, user interactions, and potential token transactions while ensuring seamless integration with the Ethereum blockchain.

2. Integrating Layer 2 solutions: Selecting and incorporating an appropriate Layer 2 solution, such as Optimism, zkSync, Solana, Arbitrum, or Polygon, to address scalability and transaction fee concerns without compromising security or decentralization.

3. Ensuring security and privacy: Addressing potential security and privacy risks related to user data, smart contract vulnerabilities, and platform interactions, and implementing measures to safeguard the platform and its users.

4. Developing an intuitive and responsive user interface: Creating a user interface that is visually appealing, easy to navigate, and responsive to user actions, while integrating blockchain functionalities such as wallet connections and transaction confirmations.

5. Balancing user experience with monetization strategies: Implementing monetization strategies that generate revenue for the platform while preserving a positive user experience and fostering genuine collaboration and engagement.

6. Bot prevention and rate limiting: Navigating potential misuse of the platform will be a challenge. Addressing bots with CAPTCHAs and pixel placement rate limiting should help prevent misuse.

These complexities and challenges highlight the need for careful planning, design, and execution to successfully develop Ethr/place as

a leading decentralized, collaborative pixel art platform on the Ethereum blockchain.

IV. Project Goals

1. Goal: Develop a decentralized, collaborative pixel art platform on the Ethereum blockchain.

- Expected Result: A fully functional Ethr/place platform that allows users to create and collaborate on pixel art in real-time.

- How to Reach: Design and implement smart contracts, integrate Layer 2 solutions, and develop a user-friendly interface that connects to the Ethereum blockchain.

2. Goal: Address the challenges faced by the original r/place project, such as bot interference and the influencer effect.

- Expected Result: A fair and engaging platform that encourages genuine user participation and minimizes the impact of bots and undue influence.

- How to Reach: Implement bot prevention and detection mechanisms, introduce rate-limiting features, ensure incentive balancing, and enable community moderation.

3. Goal: Implement user incentives and monetization strategies that balance revenue generation with a positive user experience.

- Expected Result: A sustainable platform that rewards users for their contributions while generating revenue to support development, maintenance, and marketing efforts.

- How to Reach: Integrate a referral program, establish a revenue-sharing model, and explore partnerships or licensing opportunities for monetization.

4. Goal: Foster community engagement and decentralized governance.

- Expected Result: An actively engaged user base that participates in decision-making processes and contributes to the platform's growth and success.

- How to Reach: Enable voting and proposal mechanisms for platform updates, allocate incentives for governance participation, and encourage user involvement in content moderation and flagging.

5. Goal: Ensure the security, privacy, and scalability of the Ethr/place platform.

- Expected Result: A secure and privacy-respecting platform that can accommodate a growing user base and adapt to increasing demand.

- How to Reach: Perform thorough security audits, implement privacy measures, and leverage Layer 2 solutions for improved scalability.

By focusing on these project goals and working diligently towards the expected results, the Ethr/place project aims to establish itself as a leading platform for decentralized digital art creation, collaboration, and community engagement on the Ethereum blockchain.

V. Methods

1. Front-end Development

To create a seamless user interface, we employed JavaScript with the React-JS library to develop a single-page application. The website interface is designed to be user-friendly, featuring three main buttons: Canvas, About, and Wallet. These buttons provide the following functionalities:

- Canvas: Directs users to the pixel art canvas, where they can view the current canvas and add pixels.
- About: Redirects users to the project's whitepaper.
- Wallet: Displays the wallet associated with the user's browser.

To interface with the Ethereum network, we incorporated web3.js, ethers, and truffle libraries. The web3.js library enables interaction with local or remote Ethereum nodes using HTTP, IPC, or WebSocket [5]. Truffle serves as a development environment, testing framework, and asset pipeline for Ethereum, facilitating the compilation and deployment of our smart contracts. Ganache was employed as a private Ethereum blockchain testing environment, allowing us to deploy our smart contract and monitor transactions and output.

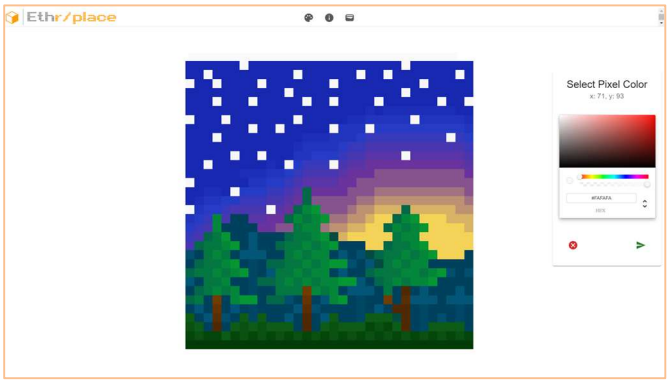

***Figure 1**: The Ethr/place website canvas page which allows the user to view the current canvas and place a pixel as a transaction.*

2. Smart Contracts

At this stage of development, we have designed a smart contract that handles pixel placement and a contract that will ensure the contracts run on the Polygon blockchain. Future contracts will address the canvas storage and retrieval, NFT minting, and auctions for the final image and timelapse video NFTs. The information

about the smart contracts we have completed so far and the contracts we have in the pipeline are presented in this section.

2.1 Pixel Placement Smart Contract

The pixel placement smart contract, shown below in Contract 1, manages the placement of pixels on the canvas. Here we attempt to store the timestamp of the last pixel placed by each user. The updatePixel function checks if the current timestamp is greater than or equal to the last timestamp plus the minimum interval (5 minutes in this case). If this condition is not met, the transaction is reverted with an error message indicating that the rate limit has been exceeded.

```
pragma solidity ^0.8.0;

import "@openzeppelin/contracts/utils/math/SafeMath.sol";
import "@openzeppelin/contracts/access/Ownable.sol";

contract PixelCanvas {
    uint256 public constant MINIMUM_INTERVAL = 5 minutes;

    struct Pixel {
        uint256 x;
        uint256 y;
        uint8[3] color; // RGB color
        address owner;
    }

    mapping(uint256 => mapping(uint256 => Pixel)) private
pixelMap;
    mapping(address => uint256) private
lastPixelPlacedTimestamp;

    event PixelUpdated(uint256 x, uint256 y, uint8[3] color,
address owner);

    function updatePixel(uint256 x, uint256 y, uint8[3] memory
color) public {
        uint256 currentTime = block.timestamp;
        uint256 userLastTimestamp =
lastPixelPlacedTimestamp[msg.sender];

        require(currentTime >= userLastTimestamp +
MINIMUM_INTERVAL, "Rate limit exceeded");
        require(x < 1000, "X coordinate out of bounds");
        require(y < 1000, "Y coordinate out of bounds");

        pixelMap[x][y] = Pixel(x, y, color, msg.sender);
        lastPixelPlacedTimestamp[msg.sender] = currentTime;
        emit PixelUpdated(x, y, color, msg.sender);
    }

    function getPixel(uint256 x, uint256 y) public view returns
(Pixel memory) {
        require(x < 1000, "X coordinate out of bounds");
        require(y < 1000, "Y coordinate out of bounds");
        return pixelMap[x][y];
    }
}
```

Contract 1: *Pixel Placement smart contract that interacts with our front-end.*

2.2 Polygon Network Contract

The following contract (Contract 2) inherits from the pixel placement contract and will be deployed on the Polygon network. The original contract already addresses the core functionality, so no additional logic is required.

```
// This contract inherits from the `PixelCanvas` contract, and no
additional logic is required.
// The original contract already handles the core functionality.

// You need to update the `truffle-config.js` file to include the
Polygon network configuration
// (e.g., using Polygon's Mumbai testnet).

pragma solidity ^0.8.0;

import "./PixelCanvas.sol";

contract PixelCanvasPolygon is PixelCanvas {
    constructor() PixelCanvas() {}
}
```

Contract 2: *Contract that inherits from the pixel placement contract and will be deployed on the Polygon network.*

Integrating the Polygon Layer 2 network for handling pixel placement transactions in Ethr/place provides numerous advantages over conducting these transactions on the Ethereum main net. The primary motivation for using Polygon is to mitigate the high gas fees and latency associated with Ethereum transactions, which would otherwise hinder user participation in the platform since the fun comes from placing multiple pixels in mass coordination. By leveraging Polygon's scalability, Ethr/place can accommodate a larger number of transactions at a significantly lower cost, enabling more users to engage in the collaborative art creation process without being constrained by prohibitive transaction fees. Additionally, Polygon's faster block times contribute to a more seamless and responsive user experience, ensuring that pixel placements are promptly updated on the canvas.

2.3 NFT Minting Contract (incomplete)

To create the NFT minting contract, we will utilize the ERC-721 token standard for non-fungible tokens on the Ethereum network. This standard ensures that our NFTs are compatible with existing wallets, marketplaces, and other services in the Ethereum ecosystem. The NFT minting contract will inherit from the OpenZeppelin ERC-721 contract [7], which provides a secure and robust implementation of the standard.

The minting contract will include functions for minting the final canvas and timelapse video as separate NFTs. Each NFT will have a unique token ID and associated metadata, such as the canvas dimensions, the list of contributing artists, and the total number of pixels placed. The contract will also include a function for transferring ownership of the NFTs to the auction contract, which will handle the subsequent auction process.

2.4 Auction Contract (incomplete)

The auction contract will manage the bidding and sale of the minted NFTs. To implement the auction functionality, the contract will include functions for setting the auction parameters, such as the start and end times, reserve price, and bid increments. Additionally, the contract will handle the submission and validation of bids, ensuring that each new bid is higher than the previous bid and meets the specified bid increment criteria.

At the end of the auction, the contract will automatically transfer ownership of the NFT to the highest bidder and distribute the auction proceeds to the appropriate recipients. A revenue-sharing model can be employed to allocate a portion of the auction proceeds to users based on their contributions to the canvas, such as the number of pixels placed. This mechanism ensures that users are fairly compensated for their efforts and encourages active participation in the platform.

2.5 Canvas Contract

Storing and retrieving the entire pixel canvas in a cost-effective manner poses a significant challenge, especially when dealing with large canvases, such as 1000x1000 or 10,000x10,000 pixels. A dedicated "Canvas" contract can be implemented to manage the storage and retrieval of the pixel canvas data while minimizing storage costs and optimizing access times. Creating a method to retrieve and store the canvas in a cost-effective manner is proving difficult. Some of our potential solutions are discussed in the Challenges section.

The Canvas contract will also include functions for updating and querying pixel data. Users can call the updatePixel function to modify a pixel's color, and the contract will update the canvasData mapping accordingly.

3. Front-end and Blockchain Integration

The front-end implementation is currently in the testing phase and interacts with the deployed PixelCanvas.sol contract on the Ganache blockchain. The code snippet in Code Block 1 showcases how the front-end communicates with the smart contract.

```
import Web3 from "web3";
import pixelCanvasABI from '../abi/PixelCanvas.json'
import {useState} from "react";

// Development network (Ganache) to be up and running or else
application won't work
// The following variables need to be adjusted to that of your
local network

let web3 = new Web3(new
Web3.providers.WebsocketProvider('ws://localhost:7545')); //
Ganache RPC Server
const contractAddress =
'0xf6e1Ef912AFa14d6a6C949Aa4ac1F49148f28818';
const contract = new web3.eth.Contract(pixelCanvasABI.abi,
contractAddress);

export default function useContract(canvasContext) {

    const [account, setAccount] =
useState(getUsersWalletAddress());
    const context = {account};

    contract.events.PixelUpdated({}, (error, event) => {
        if (error) console.log(error);
        const {x, y, color} =  event.returnValues;
        canvasContext.changePixel(
            parseInt(x),
            parseInt(y),
            [parseInt(color[0]), parseInt(color[1]),
parseInt(color[2])]
        );
    });

    return {
        updatePixel: (x, y, color) => updatePixel(x, y, color,
context),
        getPixel: (x, y) => getPixel(x, y, context)
    }
}

function updatePixel(x, y, color, context) {
    const {account} = context;
```

```
    contract.methods.updatePixel(x, y, color).send({ from:
account, gas:'800000' }) // Had to set this gas to fix out of
gas error
        .on('transactionHash', (hash) => {
            // console.log(`Transaction hash: ${hash}`);
        })
        .on('receipt', (receipt) => {
            // console.log(`Receipt: ${JSON.stringify(receipt,
null, 2)}`);
        })
        .on('error', (error) => {
            console.error(error);
        });
}

function getPixel(x, y, context) {
    const {account} = context;
    contract.methods.getPixel(x, y).call({ from: account })
        .then((result) => {
            // console.log(`Pixel data: ${JSON.stringify(result,
null, 2)}`);
        })
        .catch((error) => {
            console.error(error);
        });
}
```

Code Block 1: *Contract, Blockchain, Website interaction.*

4. Deployment

After launching the website, we compile the smart contracts using Truffle and deploy them to the Ganache blockchain. This allows us to test transactions and verify the blockchain logs.

In the Ganache environment, we can observe pixel placement transactions and contract calls, as demonstrated by the images below:

Figure 2: *Pixel placement transaction*

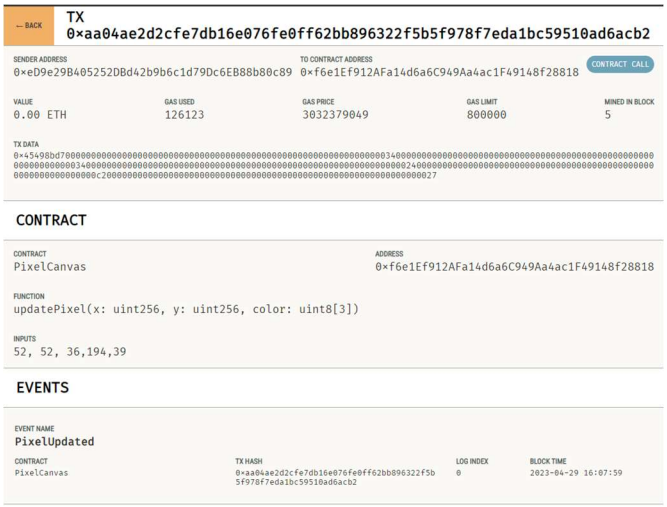

Figure 3: *Contract call for the pixel placement transaction.*

VI. Challenges

1. Canvas Contract: Efficient Storage and Retrieval

Developing the "Canvas" contract to store and return the entire pixel canvas posed a significant challenge due to the potentially large size of the canvas, ranging from 1000x1000 to 10,000x10,000 pixels. Storing and retrieving this amount of data on the Ethereum blockchain can be costly in terms of gas fees, and we have not tested the cost effectiveness on the Polygon blockchain, but we are positive it will be far more scalable and cost-effective. Ensuring that placing a single pixel is not expensive is of utmost importance if we want this project to succeed.

One potential solution for storage we are considering is to use a sparse data structure that only stores pixel color data for pixels modified by users, as described in Section 2.1. By employing a mapping to store the canvas data and emitting events each time a pixel is updated, it may be possible to minimize storage costs. Retrieving the entire canvas can be expensive in terms of gas costs when performed on-chain. Instead, we can implement an off-chain solution to reconstruct the canvas using event logs. By querying the event logs off-chain, we can obtain a list of all the pixel updates that have occurred on the canvas. The front-end application can then offload the task of reconstructing the canvas to the user's local machine by applying the updates in order. This approach significantly reduces the gas costs associated with canvas retrieval, as it moves the data processing workload from the blockchain to the user's local machine. Do we want to do this off-chain though? Will that affect the appeal of the project?

2. Contract and Front-End Security

Ensuring the security of the contract and front-end components of Ethr/place also posed a challenge. Vulnerabilities in the contract code or front-end implementation could be exploited by malicious actors to manipulate the pixel canvas or compromise user information. Being as this is our first project in the blockchain space, we have to put in our due diligence to ensure that the contract interactions are

secure. We are attempting to keep the project as simple as possible to minimize the chances of a severe vulnerability.

On the front-end side, we faced challenges in securely managing user interactions with the Ethereum network, as well as ensuring the safe handling of user information. We integrated web3.js and ethers libraries to facilitate secure interactions with the blockchain but when money is involved, we need to be vigilant in our secure coding practices and audit our code prior to launch.

3. Monetization

Monetizing the Ethr/place platform presents several challenges, primarily due to the need to strike a balance between affordability for users and generating revenue for platform maintenance and development.

3.1 Pixel placement costs

One of the main challenges is ensuring that the cost of pixel placement remains low enough to encourage user participation while still generating revenue. If the cost of placing a pixel is too high, users may be deterred from participating in the platform, which would hinder its growth and potential for monetization. Conversely, if the cost is too low, the platform may struggle to generate sufficient revenue to cover operational costs and support ongoing development. This is why we need to ensure that the canvas retrieval and storage is cost effective. Otherwise, we will have to use an off-chain solution for the canvas which may reduce the appeal of the project for some.

3.2 Pixel Update Costs

To address the challenge of monetizing pixel placements, the platform is considering implementing a fee structure where updating an existing pixel (changing its color) costs more with each subsequent update. While this strategy could help generate additional revenue, it may also discourage users from making updates if the costs become prohibitively high which could make the platform feel like a “pay to win” game. The original idea for this “increasing price for each pixel color update” was to dissuade people from continuously changing certain pixels and zones. Striking the right balance between encouraging user engagement and generating revenue through pixel updates is a critical challenge for the platform. We believe that asking the community for input on this would be the best approach.

3.3 Auction Revenue Distribution

Another monetization challenge involves the distribution of auction revenue. The platform plans to take a cut of the auction proceeds for the final NFT, with the remainder distributed among users based on their contributions. This model ensures that users are fairly compensated for their efforts while also generating revenue for the platform. However, determining the appropriate share for the platform and the users can be a challenge, as an excessively high platform share could discourage users from participating, while a low share could limit the platform's revenue generation.

4. Rate-Limiting

To ensure a level playing field for all users and prevent any single user from dominating the canvas, the platform aims to enforce a rate limit of one pixel placement per user every five minutes. Implementing this constraint on the blockchain level can be challenging but one approach we have discussed is to enforce rate limiting within a smart contract is by maintaining a mapping of user addresses to their last pixel placement timestamps. This is seen in the contract in section 2.1. Whenever a user attempts to place a pixel, the smart contract checks if the required minimum time interval has passed since their last pixel placement. If the minimum interval has not passed, the transaction will be reverted. This check will increase the gas cost per transaction, so we need to do a further investigation into whether or not this is the best approach, so we are considering other solutions as well.

One potential solution to rate limiting is to enforce these constraints on the front-end level. However, this approach has its own challenges, as front-end enforcement can be easily bypassed by users with programming skills or by using browser extensions and custom scripts. Ensuring that front-end enforcement is robust and resistant to bypass attempts is a significant challenge.

VII. Schedule

Week 1 (April 10 - April 14):

~~- Finalize project scope and objectives~~

~~- Assemble the development team~~

~~- Define roles and responsibilities~~

~~- Select a Layer 2 solution~~

Week 2 (April 17 - April 23):

~~- Design the system architecture and smart contracts~~

~~- Begin developing the user interface~~

~~- Begin smart contract design~~

Week 3 (April 24 - April 30):

- Complete smart contract development
    - ~~Pixel Placement Contract~~
    - ~~Layer 2 Contract~~
    - Auction Contract
    - NFT Minting Contract
    - Canvas Contract

~~- Finish user interface development, including wallet integration and transaction handling~~

- Implement bot prevention and rate-limiting features

- Review and finalize the incentive mechanisms and monetization strategies

Week 4 (May 1 - May 7):

~~- Turn in final report and presentation for CS458~~

- Perform thorough security audits and address any vulnerabilities

- Test the platform for scalability and performance

Week 5 (May 8 - May 14):

- Launch a beta version of Ether/place for user testing and feedback

- Monitor platform performance and user engagement

- Address any bugs or issues identified during beta testing

Week 6 (May 15 - May 21):

- Incorporate user feedback and make any necessary adjustments to the platform

- Prepare marketing materials and promotional campaigns

Week 7 (May 22 - May 28):

- Finalize the platform for public release

- Launch marketing campaigns and community engagement initiatives

- Release Ether/place to the public

Week 8 (May 29 - June 4):

- Monitor user adoption and platform performance

- Address any post-launch issues or concerns

- Begin planning for future updates and feature enhancements

VI. Conclusion

In this paper, we presented Ethr/place, a decentralized, collaborative pixel art platform inspired by Reddit's r/place project. By employing the Ethereum blockchain, Layer 2 solutions, and smart contracts, Ethr/place addresses some of the challenges encountered in the original project, such as bot interference and the influencer effect, while offering unique incentive mechanisms and potential monetization strategies.

The integration of a user-friendly front-end with a robust back-end powered by smart contracts provides a transparent, secure, and engaging platform for users to participate in digital art creation and collaboration. Ethr/place has the potential to redefine the collaborative art experience by combining the benefits of decentralization, blockchain technology, and community-driven governance.

Future developments of the platform as we continue development will include the addition of smart contracts for NFT minting, canvas management solutions, and auctions, as well as the deployment of the platform on the Layer 2 Polygon network for improved scalability and reduced transaction fees. As the project continues to evolve, we will keep updating this document which is located in the Ethr/place site ‘about’ section. Keeping this project transparent will be a way to allow the community to contribute to its development and success.

REFERENCES

[1] Simpson, Brian; Lee, Matt; Ellis, Daniel (13 April 2017). "How We Built r/Place". Upvoted. Archived from the original on 17 April 2017. Retrieved 1 July 2020.

[2] Rappaz, Jérémie (2018). "Latent Structure in Collaboration: The Case of Reddit r/place". International AAAI Conference on Web and Social Media. 12. arXiv:1804.05962. doi:10.1609/icwsm.v12i1.15013. S2CID 4941892. Archived from the original on 2 July 2020. Retrieved 1 July 2020.

[3] Voon, Claire (12 April 2017). "More Than a Million Strangers Collaborate, Pixel by Pixel, on a Digital Canvas". Hyperallergic. Archived from the original on 14 June 2020. Retrieved 10 April 2020.

[4] Litherland, Kristina Torine (2018). "Together you can create something more: Social Structures and Practice of 21st Century Skills in Mass Collaboration." Master thesis, University of Oslo. p. 8. Retrieved 12 June 2022.

[5] Web3.js - Ethereum javascript API¶,” web3.js - Ethereum JavaScript API - web3.js 1.0.0 documentation. [Online]. Available: https://web3js.readthedocs.io/en/v1.8.2/. [Accessed: 29-Apr-2023].

[6] “Truffle,” npm. [Online]. Available: https://www.npmjs.com/package/truffle?source=post_page----------. [Accessed: 29-Apr-2023].

[7] “ERC 721 - OpenZeppelin Docs,” docs.openzeppelin.com. https://docs.openzeppelin.com/contracts/2.x/api/token/erc721