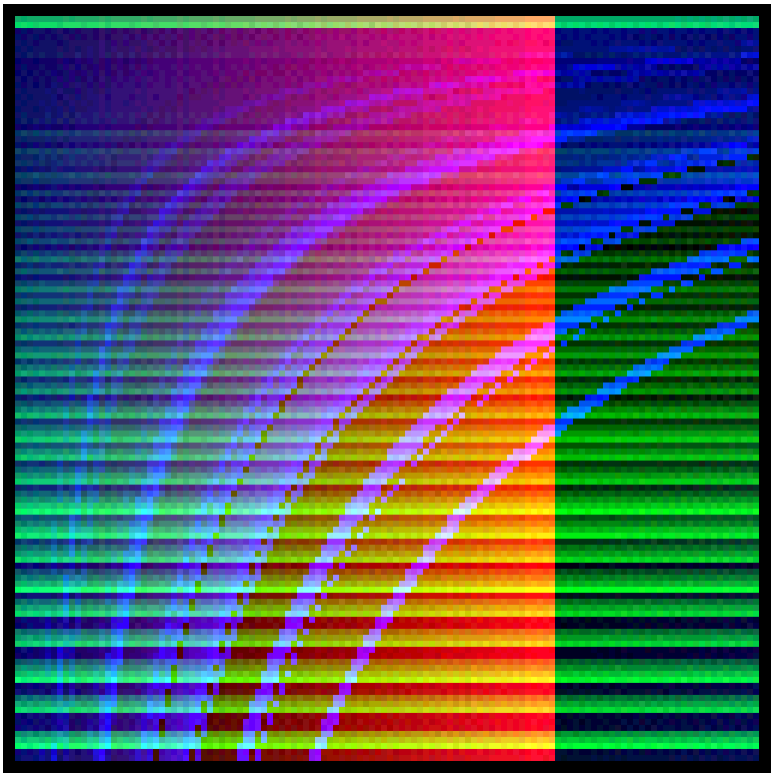
ART CONTRACTS

A Tezos DApp by @MathMakesArt

@mathmakesart mathmakesart@gmail.com
https://mathmakes.art https://artcontracts.net
mathmakesart.tez **artcontracts.tez**



“1”, the first public proof-of-concept for this project.
*Minted on HicEtNunc (**OBJKT#112092**) on 2021-06-03.*

Available for 1 XTZ until the burn date, 2021-06-30.

TABLE OF CONTENTS

INTRODUCTION TO TECHNOLOGY

Art Contracts is a project facilitating storage of art directly on the Tezos blockchain.

A prototype was built **(OBJKT#112092)** which displays on-chain static image data within an "HTML NFT" frame on HicEtNunc.

Any arbitrary data can be stored on-chain, specifically in hexadecimal form (as a bytestring). The type of art stored is limited only by the ability for that data to be translated back and forth between its original form and the hexadecimal form.

In the future, it's likely that this project will support not only static images but also animations, videos, text, audio, and even generative & interactive artworks.

Beyond simply supporting a variety of media types, the primary goal of this project is to explore the applications of on-chain art technology, and to build utilities that other developers and artists can implement into their own projects. This includes compression tools to lower the storage costs for deployment.

Some exciting possible utilities include collaborative art (multiple people editing on-chain data) and art that evolves over time without human intervention (changes in on-chain data facilitated purely by smart contract interactions). There are also many applications in the world of "collectibles", due to the permanence lent by blockchain storage.

SURVEY

Hello! Firstly, thank you for taking the time to open this document! I know not everyone will have the time to read it all, so I'm putting this survey on the first page.

CLICK HERE FOR THE SURVEY

Or go to https://forms.gle/Sp3JexeYFpWycCWc8

If you have the time, please consider filling out the survey.

It will remain open until 2021/07/02. This will ensure that even the latest buyers of the "1" NFT (**OBJKT#112092**) still have at least 2 days to complete the survey.

This survey is the first opportunity to participate in platform governance. By filling it out, you will be voting on the future development and structure of this project.

NOTE: Already less than a day after the launch of this survey, it has been met with its first Sybil attacker (https://en.wikipedia.org/wiki/Sybil_attack). A single person has filled out the survey 9 times, using a variety of fake Twitter identities.

As a result, I have been forced to enable the "Google Account" requirement for form submission. If you aren't signed in, you won't be able to access the form.

This Google Account information is NOT collected as part of the survey and will never be used or shared. It's simply being enabled as an extra anti-Sybil layer.

The "1" HicEtNunc NFT (OBJKT#112092) will also serve as the "governance token" for this survey. This means that people who have bought the NFT will be given a **5X multiplier** to the value of their vote. This is done in order to prioritize the opinions of users who have taken an active stake in the platform's success.

Users who purchase multiple copies of **OBJKT#112092** will get an **extra multiplier, equal to the square root of the number of copies owned**. This "quadratic funding" rule is added to ensure that 10 individual people, each owning 1 NFT copy each, have more voting power than a single buyer of 10 NFT copies.

Check out https://wtfisqf.com/ for a great explanation of Quadratic Funding from GitCoin, or click here to read the original QF paper by V. Buterin et al.

In the future, an actual governance token will be distributed to users of the platform. The exact reward mechanism is to-be-determined, **but the first recipients of this governance token will be users who vote in the survey!**

Even if you do not own any copies of OBJKT#112092, you can still vote in the survey and receive future governance token rewards in exchange for your vote!

This future governance token is not intended as a financial asset or investment, but as a means of distributing voting rights. It has no guarantee of any future value whatsoever.

FREQUENTLY ASKED QUESTIONS:

Q: Where can I view the proof-of-concept?

A: The on-chain data is displayed as an image within this HTML NFT on HicEtNunc (**OBJKT#112092**)

Q: How does it work?

A: There are a series of steps, divided into separate programs. Below I will give a quick summary of these steps:

1. A preprocessing program (Python) can optionally downscale the image if its resolution is too high
2. A compression program (Python) applies lossy compression via restricting the color space, and uses dithering (Floyd-Steinberg) to improve visual appearance.
3. A contract (written in SmartPy, compiled to Michelson, and deployed) takes arbitrary data as input, in the form of a key-value pair where keys are strings (name) and values are hexadecimal bytestrings (custom data). This data is saved on the Tezos blockchain. If desired, it can be easily modified/updated from the same contract in the future.
4. An interpreter (HTML webpage with custom JavaScript) is provided with specific pointer data referencing the deployed on-chain data.

Q: What is held on-chain? What is held on IPFS?

The underlying pixel data of the image is stored on-chain, as are the image dimensions, channel count, and bits-per-channel.

In the example NFT (**OBJKT#112092**) through HicEtNunc, the interpreter code is stored on IPFS. This is not a specific feature of the project I've built, but a result of minting the interpreter JavaScript on HicEtNunc within their "HTML NFT" feature.

The underlying framework for on-chain storage does not require IPFS for anything specific. The interpreter code can be embedded into any existing HTML webpage, or stored locally on the user's machine. It's also entirely possible to build more advanced interpreters which go beyond the limitations of JavaScript and HTML/CSS.

Q: What languages and tools do you use for development?

A: I currently write all my contracts in the SmartPy language, which compiles to Michelson. I do all my contract editing and deployment via the SmartPy Browser IDE.

For entrypoints and API calls I use better-call.dev

Locally, I use Visual Studio Code along with Anaconda in order to set up Python environments and run Python code. I also use VSCode for all other local code editing, which mainly amounts to JavaScript in the case of this project.

Q: Is there a cost to storing data on chain?

A: Yes. The costs come from several factors, but the most straightforward (and typically the most expensive) are the storage fees charged by the Tezos protocol.

To store data on-chain, a user must either deploy a new smart contract or interact with an existing smart contract. In both cases there is a 0.25 XTZ storage fee for every 1 KB (a Kilobyte is 1024 bytes) of on-chain storage used by a given smart contract operation. A single contract interaction can yield at most 32 KB of on-chain storage, with a storage fee of 8.00 XTZ.

It is possible to store images (and other files) on-chain, even if they are larger than 32 Kilobytes. This would require slight modifications to the compression and interpretation code, but it is very achievable.

Beyond the storage fee, there are also gas fees (paid to bakers who facilitate the transactions) and platform fees (charged by the application itself).

In this documentation, the term "**base fees**" will be used to describe the **sum of gas fees and storage fees**. The total cost of a DApp interaction is the sum of base fees and platform fees.

Later in this document, in the "Discussion of Fees" section, you can find several charts which display storage fee calculations for a variety of image sizes.

Q: Is it possible to store [insert type of file] on chain?

A: The short answer is yes! Fortunately there are no limits to the specific type of file that you can store on-chain.

The longer answer is more complicated. It comes down to a few questions like:

- Is the file size small enough that on-chain deployment is affordable? (a 1GB video would cost you 250,000 XTZ)
- How much can the data be compressed?
- Is lossy compression (giving up quality) acceptable?
- Can the data be reliably converted to and from a bytestring (hexadecimal) format?

The system that I've built is currently only capable of storing static (non-animated) image files. However, with some relatively simple modifications, the code could support the display of **text**, **animations/videos** (of small enough filesize), and any javascript-based **generative art**.

One common request I've had so far is whether on-chain storage of **sounds** and **music** is possible. It's quite possible, though the reconstruction of playable sound from a hexadecimal string is not as straightforward as image reconstruction. I think **JavaScript-based NFTs with sound** will likely be an easier project to support, versus pure sound files.

Q: How can I use this platform to deploy my art?

A: Unfortunately at the moment of publication, deployment of art is not yet open to the public.

There are many development tasks remaining before this DApp will be a ready for outside users. Much of this also depends on the preferences of the users themselves.

I have identified a list of potential improvements which can be made to the current prototype. **You can fill out this survey to vote on which development tasks should be prioritized.** If you want to deploy art ASAP, be sure to respond to the survey in order to make your voice heard.

Eventually when development has progressed enough, the platform will enter a "closed beta" period, during which a maximum of 50 users will be invited to publish some of the first art on the platform. Up to 25 of these users will be buyers of the "FIREWORKS" NFT project on HicEtNunc, which can be found here (https://www.hicetnunc.xyz/tz/tz1c6836nz23Htq4EGa9YVfZwJNsbNWaSbwF).

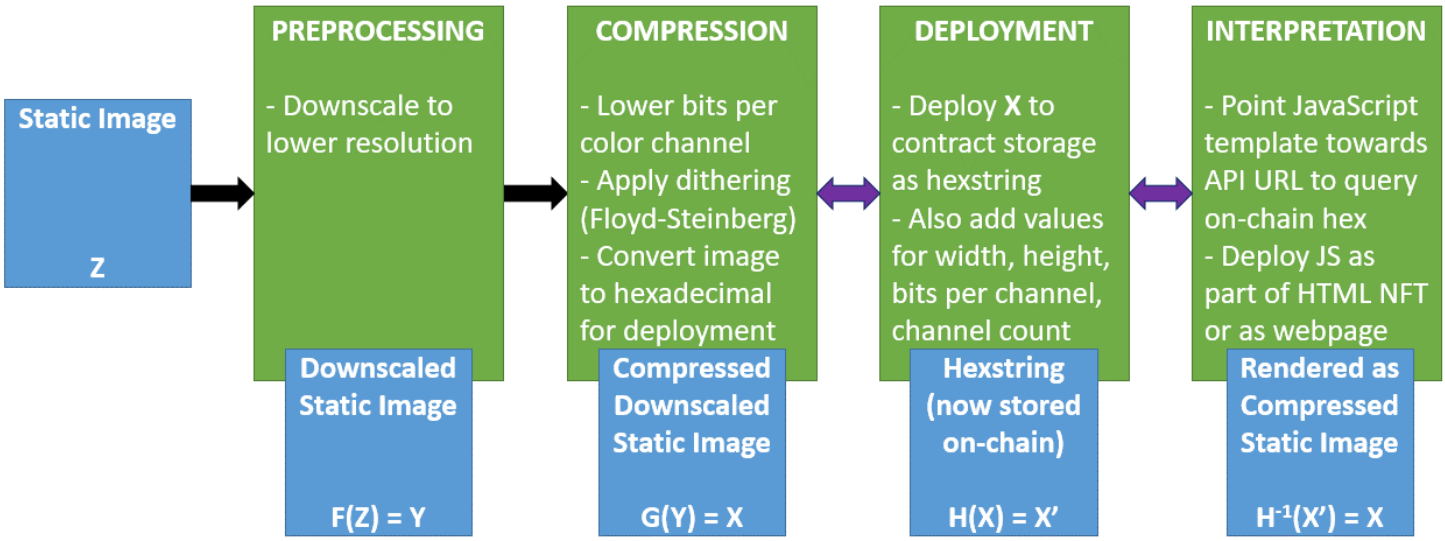
Q: What are the planned features for this platform?

A: See the **Potential Features** document for a detailed list.

Have a question of your own? Tweet it to @ArtContracts!

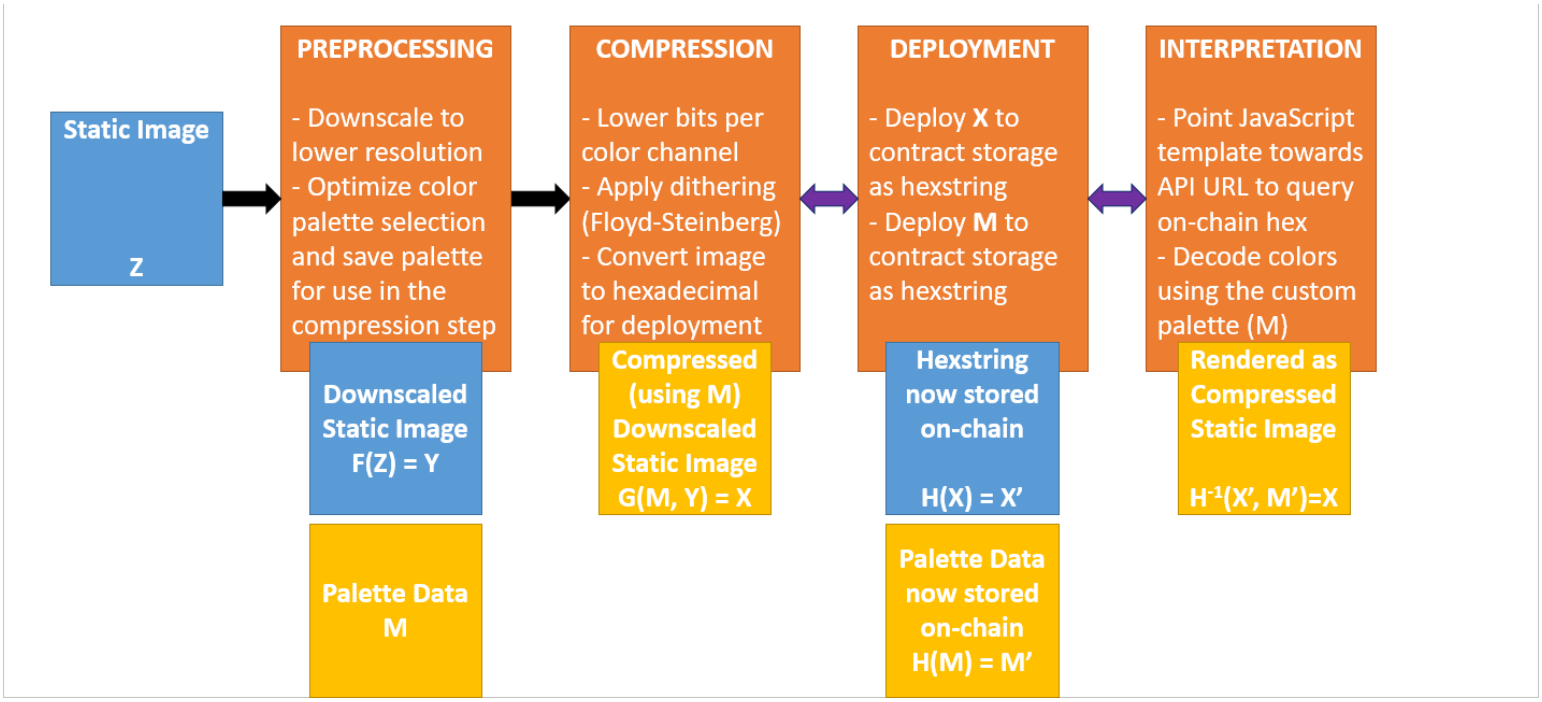
Current Project State

FIGURE 1: Explanation of Process for On-Chain Image Support



The above diagram explains how a static (non-animated) image can be transformed into a compressed bytestring (hexadecimal) representation, stored on-chain, and decoded by an interpreter program for viewing by the end-user.

FIGURE 2: Explanation of Process with Custom Color Palettes



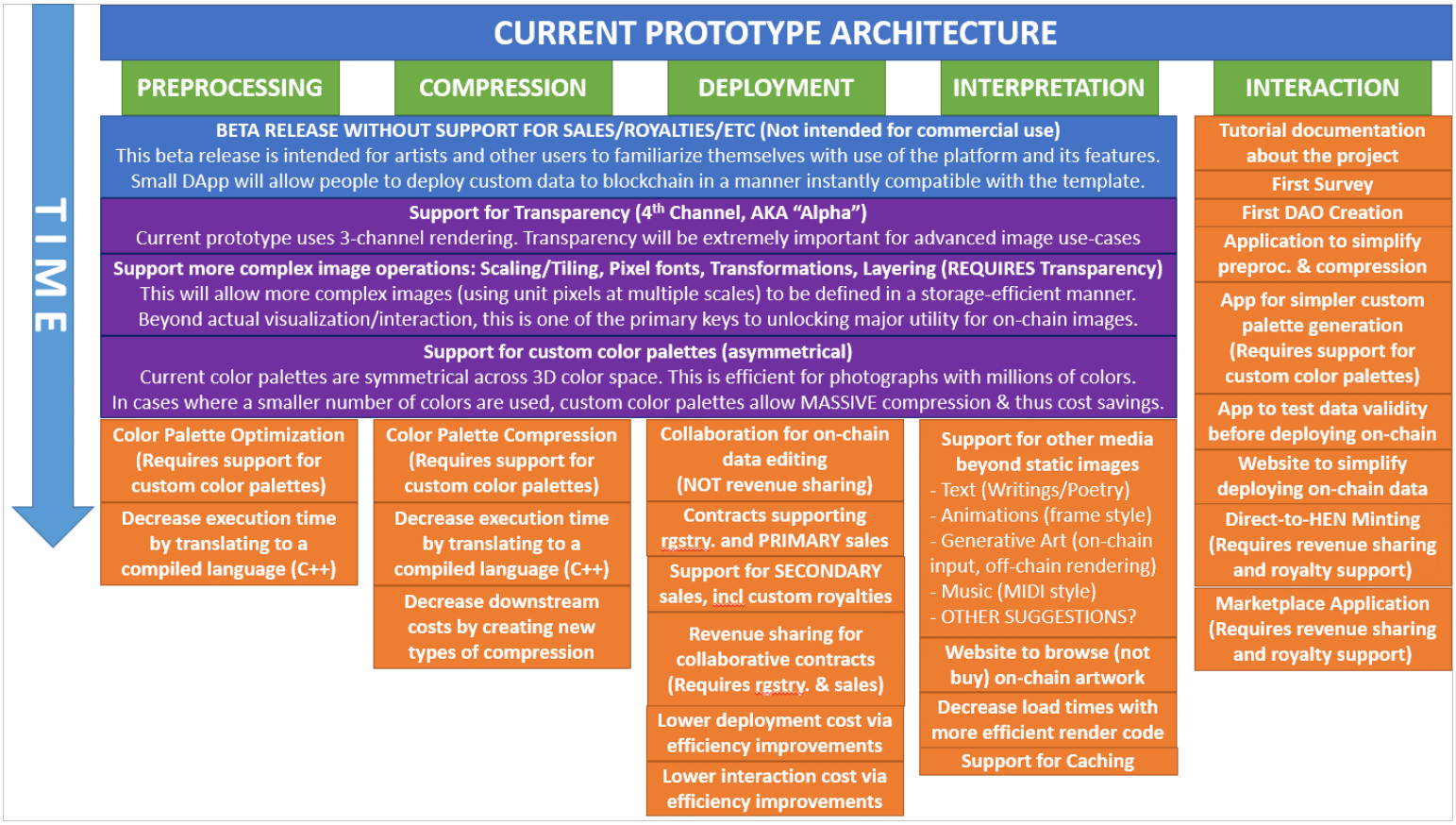
The above diagram explains an extension of the architecture from the previous page, with modifications to include a packet of data representing the "color palette" of an image.

If properly optimized, custom color palettes can significantly reduce the number of binary digits required to represent the color of each pixel. **In 2-color images, size (and thus storage costs) can be reduced by a factor of up to 8x. And even images with over 4,000 unique colors can still see up to 4x reduction in the on-chain storage costs when properly optimized.**

The above point makes it clear why this single feature is incredibly valuable: It can lower the costs for many users!

Chronology of Planned Features

FIGURE 3: Development Timeline Diagram



The above diagram organizes a number of potential features and development tasks to show their category within the project code and any dependence on other development tasks.

On the following pages, a more detailed outline explains the features mentioned in the diagram above.

BASE CONTENTS

1. **PREPROCESSING**: Downscale of image, with or without interpolation (done)
2. **COMPRESSION**: Compressor & ditherer (done)
3. **DEPLOYMENT**: Data contract (done)
4. **INTERPRETATION**: Code for HEN “HTML minting” of Image data (done)
5. **INTERACTION**: Additional features which extend the base functions above

FEATURES REQUIRING CHANGES TO BASE CONTENTS

6. **Support for transparency**
7. **Support for more complex image-related operations**
    a. Scaling and tiling (integer multipliers)
    b. Layering
        i. Multiple images layered on top of each other, ordered
        ii. Take transparency into account as well
    c. Transformations
        i. Translation (integer pixel increments)
        ii. Rotation (90 degree increments)
        iii. Reflection (across horizontal or vertical axis)
    d. Pixel Fonts
        i. Likely achieved using extra .js file(s) as “extensions”
        ii. Addition of text characters to a layered image
        iii. Supplied as text strings, not pixel layout definitions
        iv. Interpreter converts characters into desired layout
8. **Support for custom color palettes**
    a. Major development task
    b. Very rewarding for some niches
        i. Lowers cost by improving storage efficiency
    c. Requires additional “palette optimization” software

FEATURES STEMMING DIRECTLY FROM COMPRESSION (2)

9. **Compression of Color Palettes**
    a. Requires support for custom color palettes
    b. This compression is the primary motivation for custom color palettes, as it can lead to significantly lower storage costs for artwork with a limited color palette
    c. In theory this same type of compression can be applied to other datatypes, but images are the most natural recipient.
10. **Improvements to efficiency**
    a. Up runtime efficiency by translating to a compiled language like C++
    b. Up storage efficiency through new better compression schemes

FEATURES STEMMING DIRECTLY FROM DEPLOYMENT (3)

11. **BETA RELEASE WITHOUT SUPPORT FOR SALES/ROYALTIES/ETC**
    a. Not intended for commercial use
    b. For artists and other users to familiarize themselves with platform
    c. Small DApp will allow people to deploy custom data to the Tezos blockchain, in a manner instantly compatible with the INTERPRETER.
    d. There will be platform fees, but the fees will be lowered in order to encourage experimentation
        i. Fee will be a percentage of base (storage and gas) costs
        ii. Fees will be evaluated on a per-wallet basis
        iii. Fee for a wallet will start at 100% (of base costs), but will fall to lower percentages as a user interacts with the contract more.
    e. Users will receive governance tokens in exchange for their participation in this beta
    f. Please note that while this is not intended for commercial use, it's still possible for you to mint your own NFTs on other platforms which make use of the on-chain data that you deploy
        i. For example, the HEN NFT "1" I minted
12. **Implementation of collaboration for on-chain data editing**

a. NOT revenue sharing, but multiple people having edit permissions

13. Native support for tokenization of on-chain data

a. A multi-asset FA2 contract will allow any user, when deploying custom data to the blockchain, to "tokenize" that data by associating it with a newly-minted token (user decides supply & other variables)
b. Because these are FA2 tokens, they can have all sorts of extended functionalities (e.g. swapping on a DEX, using as a governance token)
c. Upon creation, these tokens will be sent from the "minter contract" to the wallet address of the user who deployed the on-chain data

14. Contracts supporting registry of ownership and PRIMARY sales

a. Requires tokenization support
b. One or more smart contracts will serve as an on-chain ledger of ownership for ALL tokenized instances of on-chain data
c. Creators of these tokens will be allowed to list them for sale
    i. To begin with, only a single price

15. Contracts supporting SECONDARY sales, including custom royalties

a. Requires tokenization support
b. Requires registry of ownership and primary sales
c. One or more smart contracts will facilitate listing tokenized on-chain data for sale, similar to "SWAPS" on HEN.

16. Implementation of revenue sharing for collaborative contracts

a. This will allow certain operations to automatically split revenue between any number of recipients, defined by the deployer
b. Please note that in order for this functionality to be meaningful, a large number of other features must also be developed including
    i. Tokenization support
    ii. Registry of ownership and primary sales
    iii. Secondary sales and royalties

17. Improvements to efficiency

a. Decreasing deployment cost via contract efficiency improvements
    i. Deployment costs are paid by the artist or content creator, at the time that the data is first being stored on the blockchain
b. Decreasing interaction cost via contract efficiency improvements
    i. Interaction costs are paid by the user, buyer, or collector, for certain actions related to the on-chain data

FEATURES STEMMING DIRECTLY FROM INTERPRETATION (4)

18. Support for other types of media beyond static images

a. Also requires some changes to compressor program
b. Possible media to support, in order from easy to difficult
    i. Text (poetry/writings or direct display of hex/binary/base64)
    ii. Generative art (e.g. shaders using on-chain data as input)
    iii. Animations (GIF-style animations with each frame on-chain)
    iv. Music (may be possible through a MIDI-like structure)
    v. Other suggestions?

19. Website to browse (not buy) on-chain artwork

a. Requires Registry of Ownership from DEPLOYMENT category
b. This website would display information about every on-chain deployment done through this project's contracts
c. Could also have built-in interpreter code in order to render artworks for website users to view them
d. Could eventually be extended into a full marketplace website, but that's more on the "interaction" side of things

20. Improvements to efficiency

a. Lower processing cost of conversion/rendering
b. Lower total size of off-chain (interpreter) code
c. Caching: store interpreter code separately from the necessary "pointer" descriptors, allowing the exact same code to be reused for interpretation across multiple on-chain artworks
    i. This is also useful from a permanence standpoint. If one single interpreter file can enable viewing of a large number of artworks, that single file can be pinned on IPFS much more easily than a system of hundreds/thousands of separate interpreter templates can be pinned

FEATURES EXTENDING THE "INTERACTION" CATEGORY

21. DAO Creation and Governance Tokens

a. Long-term goal should be to have no single centralized "admin"
b. Decisions should be made collectively
    i. **Surveys like this** will help facilitate this until an official system for governance and voting has been established
c. Organization will be split into multiple DAOs, each with their own governance token
d. DAO examples:
    i. User DAO
        1. Any users of the platform can gain membership
        2. Tokens give voting rights on general issues
    ii. Creator DAO
        1. Artists and publishers of content can gain membership
        2. Tokens give all voting rights of User DAO, plus ability to vote in matters that specifically impact creators
        3. To prevent conflicts of interest, certain "Creator DAO" votes are off-limits to anyone who doesn't hold Creator DAO Tokens. This applies even to "Developer DAO" members.
    iii. Developer DAO
        1. Membership given to all contributors toward project dev
        2. Tokens give all voting rights of User DAO, plus ability to vote in matters that impact the platform as a whole
        3. To prevent conflicts of interest, some "Developer DAO" votes are limited only to holders of Developer DAO Tokens. This applies even to "Creator DAO" members.

22. Application to simplify PREPROCESSING and COMPRESSION steps

a. Simplify the steps for creators who aren't familiar with running Python code on their computer

23. Application for simpler custom palette generation

a. Requires support for custom color palettes

b. Simplify the steps for creators who aren't familiar with running Python code on their computer

24. Application to test validity of data before deploying on-chain

a. Could potentially save a lot of money in storage fees. Mistakes can be expensive when it comes to putting images and other art on-chain.

25. Web application for simpler UI when deploying on-chain data

a. Allowing users to avoid direct contract interactions

26. Direct-to-HicEtNunc Minting

a. Ability to mint HEN NFTs from a contract which stores on-chain data
b. Not just a template (that users would deploy themselves)
    i. But a series of contract interactions
    ii. All different user's art is minted on HEN from the same (contract) address(es)
c. When a HEN purchase is made, contract would send ownership of a resulting token into the caller's wallet

Discussion of Fees

Base Fees

In this documentation, the term "**base fees**" will be used to describe the **sum of gas fees and storage fees**. Gas and storage fees are inherent to the Tezos network and are not paid to the owners of smart contracts. The total cost of a DApp interaction is the sum of base fees and platform fees.

As mentioned in the FAQ, the Tezos blockchain has built-in storage costs of 0.25 XTZ for every 1 KB of on-chain data storage. This represents a majority of the base fees.

The remained of base fees come from the gas costs. These are much harder to estimate, as they result less from the specifics of storage and more from the underlying computational actions carried out by a smart contract to achieve its goals.

For on-chain storage of extremely small files, it's possible that the gas fees will be larger than the storage fees. However this is relatively unlikely.

Certain other operations, like the minting of tokens or NFTs, can add significant gas costs to the concept of on-chain storage.

Platform Fees

Platform fees are a detail which is yet to be determined.

There are two major categories of platform fee:

- Fees paid once up-front but never again
    - E.g. the "*1 XTZ per 10x10 square of pixels*" fee on **muralis.xyz** (by **@NFTPROTECTOR** and **@oktuorg**)
- Fees paid continually as a percentage of transaction value
    - E.g. the "*2.5% of all sales*" fee on HicEtNunc

I see merits to each option and I hope the **survey** will serve as a good starting point for potential user feedback on fees. I certainly don't want to overcharge users of the platform. Please vote to ensure your opinion is heard!

These fees are the primary means for me (and other future developers) to be rewarded for contributions to this project. Ideally I would charge nothing, but I also want to support my ability to build this project full-time.

I want to keep the platform as accessible as possible while still maintaining my ability to contribute to it full-time. So I am very open to suggestions of mechanisms to fund platform fees (and the gas and storage fees) for artists who can't afford it.

TABLE REPRESENTATIONS OF STORAGE FEES FOR IMAGE DATA

The diagrams on the following 2 pages are summarized below:

FIGURE 4: Overview of Storage Cost Analysis for Image Data

STORAGE COST

(Excludes gas and platform fees)

Minimum XTZ cost of on-chain data storage for square images, given:

- Side length (px)
- Bits per channel
- Channel count (3 vs. 4 channels)

Green is lowest cost

Red is highest cost

PURPLE means image is larger than 32 KB, so it needs multiple contracts

Bits per Channel (3-channel Image)

Square Side Length (px)	Monochr.	1	2	3	4	5	6	7	8
5 px	0.0008	0.0023	0.0046	0.0069	0.0092	0.0114	0.0137	0.0160	0.0183
10 px	0.0031	0.0092	0.0183	0.0275	0.0366	0.0458	0.0549	0.0641	0.0732
15 px	0.0069	0.0206	0.0412	0.0618	0.0824	0.1030	0.1236	0.1442	0.1648
20 px	0.0122	0.0366	0.0732	0.1099	0.1465	0.1831	0.2197	0.2563	0.2930
25 px	0.0191	0.0572	0.1144	0.1717	0.2289	0.2861	0.3433	0.4005	0.4578
30 px	0.0275	0.0824	0.1648	0.2472	0.3296	0.4120	0.4944	0.5768	0.6592
35 px	0.0374	0.1122	0.2243	0.3365	0.4486	0.5608	0.6729	0.7851	0.8972
40 px	0.0488	0.1465	0.2930	0.4395	0.5859	0.7324	0.8789	1.0254	1.1719
45 px	0.0618	0.1854	0.3708	0.5562	0.7416	0.9270	1.1124	1.2978	1.4832
50 px	0.0763	0.2289	0.4578	0.6866	0.9155	1.1444	1.3733	1.6022	1.8311
55 px	0.0923	0.2769	0.5539	0.8308	1.1078	1.3847	1.6617	1.9386	2.2156
60 px	0.1099	0.3296	0.6592	0.9888	1.3184	1.6479	1.9775	2.3071	2.6367
65 px	0.1289	0.3868	0.7736	1.1604	1.5472	1.9341	2.3209	2.7077	3.0945
70 px	0.1495	0.4486	0.8972	1.3458	1.7944	2.2430	2.6917	3.1403	3.5889
75 px	0.1717	0.5150	1.0300	1.5450	2.0599	2.5749	3.0899	3.6049	4.1199
80 px	0.1953	0.5859	1.1719	1.7578	2.3438	2.9297	3.5156	4.1016	4.6875
85 px	0.2205	0.6615	1.3229	1.9844	2.6459	3.3073	3.9688	4.6303	5.2917
90 px	0.2472	0.7416	1.4832	2.2247	2.9663	3.7079	4.4495	5.1910	5.9326
95 px	0.2754	0.8263	1.6525	2.4788	3.3051	4.1313	4.9576	5.7838	6.6101
100 px	0.3052	0.9155	1.8311	2.7466	3.6621	4.5776	5.4932	6.4087	7.3242
110 px	0.3693	1.1078	2.2156	3.3234	4.4312	5.5389	6.6467	7.7545	8.8623
120 px	0.4395	1.3184	2.6367	3.9551	5.2734	6.5918	7.9102	9.2285	10.5469
130 px	0.5157	1.5472	3.0945	4.6417	6.1890	7.7362	9.2834	10.8307	12.3779
140 px	0.5981	1.7944	3.5889	5.3833	7.1777	8.9722	10.7666	12.5610	14.3555
150 px	0.6866	2.0599	4.1199	6.1798	8.2397	10.2997	12.3596	14.4196	16.4795
160 px	0.7813	2.3438	4.6875	7.0313	9.3750	11.7188	14.0625	16.4063	18.7500
170 px	0.8820	2.6459	5.2917	7.9376	10.5835	13.2294	15.8752	18.5211	21.1670
180 px	0.9888	2.9663	5.9326	8.8989	11.8652	14.8315	17.7979	20.7642	23.7305
190 px	1.1017	3.3051	6.6101	9.9152	13.2202	16.5253	19.8303	23.1354	26.4404
200 px	1.2207	3.6621	7.3242	10.9863	14.6484	18.3105	21.9727	25.6348	29.2969
220 px	1.4771	4.4312	8.8623	13.2935	17.7246	22.1558	26.5869	31.0181	35.4492
240 px	1.7578	5.2734	10.5469	15.8203	21.0938	26.3672	31.6406	36.9141	42.1875
260 px	2.0630	6.1890	12.3779	18.5669	24.7559	30.9448	37.1338	43.3228	49.5117
280 px	2.3926	7.1777	14.3555	21.5332	28.7109	35.8887	43.0664	50.2441	57.4219
300 px	2.7466	8.2397	16.4795	24.7192	32.9590	41.1987	49.4385	57.6782	65.9180
320 px	3.1250	9.3750	18.7500	28.1250	37.5000	46.8750	56.2500	65.6250	75.0000
340 px	3.5278	10.5835	21.1670	31.7505	42.3340	52.9175	63.5010	74.0845	84.6680
360 px	3.9551	11.8652	23.7305	35.5957	47.4609	59.3262	71.1914	83.0566	94.9219
380 px	4.4067	13.2202	26.4404	39.6606	52.8809	66.1011	79.3213	92.5415	105.7617
400 px	4.8828	14.6484	29.2969	43.9453	58.5938	73.2422	87.8906	102.5391	117.1875
420 px	5.3833	16.1499	32.2998	48.4497	64.5996	80.7495	96.8994	113.0493	129.1992
440 px	5.9082	17.7246	35.4492	53.1738	70.8984	88.6230	106.3477	124.0723	141.7969
460 px	6.4575	19.3726	38.7451	58.1177	77.4902	96.8628	116.2354	135.6079	154.9805
480 px	7.0313	21.0938	42.1875	63.2813	84.3750	105.4688	126.5625	147.6563	168.7500
500 px	7.6294	22.8882	45.7764	68.6646	91.5527	114.4409	137.3291	160.2173	183.1055
512 px	8.0000	24.0000	48.0000	72.0000	96.0000	120.0000	144.0000	168.0000	192.0000

Bits per Channel (4-channel Image)

Square Side Length (px)	Monochr.	1	2	3	4	5	6	7	8
5 px	0.0008	0.0031	0.0061	0.0092	0.0122	0.0153	0.0183	0.0214	0.0244
10 px	0.0031	0.0122	0.0244	0.0366	0.0488	0.0610	0.0732	0.0854	0.0977
15 px	0.0069	0.0275	0.0549	0.0824	0.1099	0.1373	0.1648	0.1923	0.2197
20 px	0.0122	0.0488	0.0977	0.1465	0.1953	0.2441	0.2930	0.3418	0.3906
25 px	0.0191	0.0763	0.1526	0.2289	0.3052	0.3815	0.4578	0.5341	0.6104
30 px	0.0275	0.1099	0.2197	0.3296	0.4395	0.5493	0.6592	0.7690	0.8789
35 px	0.0374	0.1495	0.2991	0.4486	0.5981	0.7477	0.8972	1.0468	1.1963
40 px	0.0488	0.1953	0.3906	0.5859	0.7813	0.9766	1.1719	1.3672	1.5625
45 px	0.0618	0.2472	0.4944	0.7416	0.9888	1.2360	1.4832	1.7303	1.9775
50 px	0.0763	0.3052	0.6104	0.9155	1.2207	1.5259	1.8311	2.1362	2.4414
55 px	0.0923	0.3693	0.7385	1.1078	1.4771	1.8463	2.2156	2.5848	2.9541
60 px	0.1099	0.4395	0.8789	1.3184	1.7578	2.1973	2.6367	3.0762	3.5156
65 px	0.1289	0.5157	1.0315	1.5472	2.0630	2.5787	3.0945	3.6102	4.1260
70 px	0.1495	0.5981	1.1963	1.7944	2.3926	2.9907	3.5889	4.1870	4.7852
75 px	0.1717	0.6866	1.3733	2.0599	2.7466	3.4332	4.1199	4.8065	5.4932
80 px	0.1953	0.7813	1.5625	2.3438	3.1250	3.9063	4.6875	5.4688	6.2500
85 px	0.2205	0.8820	1.7639	2.6459	3.5278	4.4098	5.2917	6.1737	7.0557
90 px	0.2472	0.9888	1.9775	2.9663	3.9551	4.9438	5.9326	6.9214	7.9102
95 px	0.2754	1.1017	2.2034	3.3051	4.4067	5.5084	6.6101	7.7118	8.8135
100 px	0.3052	1.2207	2.4414	3.6621	4.8828	6.1035	7.3242	8.5449	9.7656
110 px	0.3693	1.4771	2.9541	4.4312	5.9082	7.3853	8.8623	10.3394	11.8164
120 px	0.4395	1.7578	3.5156	5.2734	7.0313	8.7891	10.5469	12.3047	14.0625
130 px	0.5157	2.0630	4.1260	6.1890	8.2520	10.3149	12.3779	14.4409	16.5039
140 px	0.5981	2.3926	4.7852	7.1777	9.5703	11.9629	14.3555	16.7480	19.1406
150 px	0.6866	2.7466	5.4932	8.2397	10.9863	13.7329	16.4795	19.2261	21.9727
160 px	0.7813	3.1250	6.2500	9.3750	12.5000	15.6250	18.7500	21.8750	25.0000
170 px	0.8820	3.5278	7.0557	10.5835	14.1113	17.6392	21.1670	24.6948	28.2227
180 px	0.9888	3.9551	7.9102	11.8652	15.8203	19.7754	23.7305	27.6855	31.6406
190 px	1.1017	4.4067	8.8135	13.2202	17.6270	22.0337	26.4404	30.8472	35.2539
200 px	1.2207	4.8828	9.7656	14.6484	19.5313	24.4141	29.2969	34.1797	39.0625
220 px	1.4771	5.9082	11.8164	17.7246	23.6328	29.5410	35.4492	41.3574	47.2656
240 px	1.7578	7.0313	14.0625	21.0938	28.1250	35.1563	42.1875	49.2188	56.2500
260 px	2.0630	8.2520	16.5039	24.7559	33.0078	41.2598	49.5117	57.7637	66.0156
280 px	2.3926	9.5703	19.1406	28.7109	38.2813	47.8516	57.4219	66.9922	76.5625
300 px	2.7466	10.9863	21.9727	32.9590	43.9453	54.9316	65.9180	76.9043	87.8906
320 px	3.1250	12.5000	25.0000	37.5000	50.0000	62.5000	75.0000	87.5000	100.0000
340 px	3.5278	14.1113	28.2227	42.3340	56.4453	70.5566	84.6680	98.7793	112.8906
360 px	3.9551	15.8203	31.6406	47.4609	63.2813	79.1016	94.9219	110.7422	126.5625
380 px	4.4067	17.6270	35.2539	52.8809	70.5078	88.1348	105.7617	123.3887	141.0156
400 px	4.8828	19.5313	39.0625	58.5938	78.1250	97.6563	117.1875	136.7188	156.2500
420 px	5.3833	21.5332	43.0664	64.5996	86.1328	107.6660	129.1992	150.7324	172.2656
440 px	5.9082	23.6328	47.2656	70.8984	94.5313	118.1641	141.7969	165.4297	189.0625
460 px	6.4575	25.8301	51.6602	77.4902	103.3203	129.1504	154.9805	180.8105	206.6406
480 px	7.0313	28.1250	56.2500	84.3750	112.5000	140.6250	168.7500	196.8750	225.0000
500 px	7.6294	30.5176	61.0352	91.5527	122.0703	152.5879	183.1055	213.6230	244.1406
512 px	8.0000	32.0000	64.0000	96.0000	128.0000	160.0000	192.0000	224.0000	256.0000

- **Cell values give storage cost, in XTZ**
- The **X axis** of the table is the **bits per channel** (from 1 to 8)
  - Also has a “Monochrome” (B&W) reference column
- The **Y axis** is the **side length, in pixels**, of a square image

The left diagram (first page) shows storage fees for **3-channel** images (RGB), which do not support transparency.

The right diagram (second page) shows that of **4-channel** images (RGBA), which support transparency via Alpha channel.

FIGURE 5: Storage Cost Analysis for 3-Channel Image Data

Square Side Length (px)	Monochr.	Bits per Channel (3-channel Image) 1	2	3	4	5	6	7	8
5 px	0.0008	0.0023	0.0046	0.0069	0.0092	0.0114	0.0137	0.0160	0.0183
10 px	0.0031	0.0092	0.0183	0.0275	0.0366	0.0458	0.0549	0.0641	0.0732
15 px	0.0069	0.0206	0.0412	0.0618	0.0824	0.1030	0.1236	0.1442	0.1648
20 px	0.0122	0.0366	0.0732	0.1099	0.1465	0.1831	0.2197	0.2563	0.2930
25 px	0.0191	0.0572	0.1144	0.1717	0.2289	0.2861	0.3433	0.4005	0.4578
30 px	0.0275	0.0824	0.1648	0.2472	0.3296	0.4120	0.4944	0.5768	0.6592
35 px	0.0374	0.1122	0.2243	0.3365	0.4486	0.5608	0.6729	0.7851	0.8972
40 px	0.0488	0.1465	0.2930	0.4395	0.5859	0.7324	0.8789	1.0254	1.1719
45 px	0.0618	0.1854	0.3708	0.5562	0.7416	0.9270	1.1124	1.2978	1.4832
50 px	0.0763	0.2289	0.4578	0.6866	0.9155	1.1444	1.3733	1.6022	1.8311
55 px	0.0923	0.2769	0.5539	0.8308	1.1078	1.3847	1.6617	1.9386	2.2156
60 px	0.1099	0.3296	0.6592	0.9888	1.3184	1.6479	1.9775	2.3071	2.6367
65 px	0.1289	0.3868	0.7736	1.1604	1.5472	1.9341	2.3209	2.7077	3.0945
70 px	0.1495	0.4486	0.8972	1.3458	1.7944	2.2430	2.6917	3.1403	3.5889
75 px	0.1717	0.5150	1.0300	1.5450	2.0599	2.5749	3.0899	3.6049	4.1199
80 px	0.1953	0.5859	1.1719	1.7578	2.3438	2.9297	3.5156	4.1016	4.6875
85 px	0.2205	0.6615	1.3229	1.9844	2.6459	3.3073	3.9688	4.6303	5.2917
90 px	0.2472	0.7416	1.4832	2.2247	2.9663	3.7079	4.4495	5.1910	5.9326
95 px	0.2754	0.8263	1.6525	2.4788	3.3051	4.1313	4.9576	5.7838	6.6101
100 px	0.3052	0.9155	1.8311	2.7466	3.6621	4.5776	5.4932	6.4087	7.3242
110 px	0.3693	1.1078	2.2156	3.3234	4.4312	5.5389	6.6467	7.7545	8.8623
120 px	0.4395	1.3184	2.6367	3.9551	5.2734	6.5918	7.9102	9.2285	10.5469
130 px	0.5157	1.5472	3.0945	4.6417	6.1890	7.7362	9.2834	10.8307	12.3779
140 px	0.5981	1.7944	3.5889	5.3833	7.1777	8.9722	10.7666	12.5610	14.3555
150 px	0.6866	2.0599	4.1199	6.1798	8.2397	10.2997	12.3596	14.4196	16.4795
160 px	0.7813	2.3438	4.6875	7.0313	9.3750	11.7188	14.0625	16.4063	18.7500
170 px	0.8820	2.6459	5.2917	7.9376	10.5835	13.2294	15.8752	18.5211	21.1670
180 px	0.9888	2.9663	5.9326	8.8989	11.8652	14.8315	17.7979	20.7642	23.7305
190 px	1.1017	3.3051	6.6101	9.9152	13.2202	16.5253	19.8303	23.1354	26.4404
200 px	1.2207	3.6621	7.3242	10.9863	14.6484	18.3105	21.9727	25.6348	29.2969
220 px	1.4771	4.4312	8.8623	13.2935	17.7246	22.1558	26.5869	31.0181	35.4492
240 px	1.7578	5.2734	10.5469	15.8203	21.0938	26.3672	31.6406	36.9141	42.1875
260 px	2.0630	6.1890	12.3779	18.5669	24.7559	30.9448	37.1338	43.3228	49.5117
280 px	2.3926	7.1777	14.3555	21.5332	28.7109	35.8887	43.0664	50.2441	57.4219
300 px	2.7466	8.2397	16.4795	24.7192	32.9590	41.1987	49.4385	57.6782	65.9180
320 px	3.1250	9.3750	18.7500	28.1250	37.5000	46.8750	56.2500	65.6250	75.0000
340 px	3.5278	10.5835	21.1670	31.7505	42.3340	52.9175	63.5010	74.0845	84.6680
360 px	3.9551	11.8652	23.7305	35.5957	47.4609	59.3262	71.1914	83.0566	94.9219
380 px	4.4067	13.2202	26.4404	39.6606	52.8809	66.1011	79.3213	92.5415	105.7617
400 px	4.8828	14.6484	29.2969	43.9453	58.5938	73.2422	87.8906	102.5391	117.1875
420 px	5.3833	16.1499	32.2998	48.4497	64.5996	80.7495	96.8994	113.0493	129.1992
440 px	5.9082	17.7246	35.4492	53.1738	70.8984	88.6230	106.3477	124.0723	141.7969
460 px	6.4575	19.3726	38.7451	58.1177	77.4902	96.8628	116.2354	135.6079	154.9805
480 px	7.0313	21.0938	42.1875	63.2813	84.3750	105.4688	126.5625	147.6563	168.7500
500 px	7.6294	22.8882	45.7764	68.6646	91.5527	114.4409	137.3291	160.2173	183.1055
512 px	8.0000	24.0000	48.0000	72.0000	96.0000	120.0000	144.0000	168.0000	192.0000

FIGURE 6: Storage Cost Analysis for 4-Channel Image Data

Square Side Length (px)	Monochr.	Bits per Channel (4-channel Image) 1	2	3	4	5	6	7	8
5 px	0.0008	0.0031	0.0061	0.0092	0.0122	0.0153	0.0183	0.0214	0.0244
10 px	0.0031	0.0122	0.0244	0.0366	0.0488	0.0610	0.0732	0.0854	0.0977
15 px	0.0069	0.0275	0.0549	0.0824	0.1099	0.1373	0.1648	0.1923	0.2197
20 px	0.0122	0.0488	0.0977	0.1465	0.1953	0.2441	0.2930	0.3418	0.3906
25 px	0.0191	0.0763	0.1526	0.2289	0.3052	0.3815	0.4578	0.5341	0.6104
30 px	0.0275	0.1099	0.2197	0.3296	0.4395	0.5493	0.6592	0.7690	0.8789
35 px	0.0374	0.1495	0.2991	0.4486	0.5981	0.7477	0.8972	1.0468	1.1963
40 px	0.0488	0.1953	0.3906	0.5859	0.7813	0.9766	1.1719	1.3672	1.5625
45 px	0.0618	0.2472	0.4944	0.7416	0.9888	1.2360	1.4832	1.7303	1.9775
50 px	0.0763	0.3052	0.6104	0.9155	1.2207	1.5259	1.8311	2.1362	2.4414
55 px	0.0923	0.3693	0.7385	1.1078	1.4771	1.8463	2.2156	2.5848	2.9541
60 px	0.1099	0.4395	0.8789	1.3184	1.7578	2.1973	2.6367	3.0762	3.5156
65 px	0.1289	0.5157	1.0315	1.5472	2.0630	2.5787	3.0945	3.6102	4.1260
70 px	0.1495	0.5981	1.1963	1.7944	2.3926	2.9907	3.5889	4.1870	4.7852
75 px	0.1717	0.6866	1.3733	2.0599	2.7466	3.4332	4.1199	4.8065	5.4932
80 px	0.1953	0.7813	1.5625	2.3438	3.1250	3.9063	4.6875	5.4688	6.2500
85 px	0.2205	0.8820	1.7639	2.6459	3.5278	4.4098	5.2917	6.1737	7.0557
90 px	0.2472	0.9888	1.9775	2.9663	3.9551	4.9438	5.9326	6.9214	7.9102
95 px	0.2754	1.1017	2.2034	3.3051	4.4067	5.5084	6.6101	7.7118	8.8135
100 px	0.3052	1.2207	2.4414	3.6621	4.8828	6.1035	7.3242	8.5449	9.7656
110 px	0.3693	1.4771	2.9541	4.4312	5.9082	7.3853	8.8623	10.3394	11.8164
120 px	0.4395	1.7578	3.5156	5.2734	7.0313	8.7891	10.5469	12.3047	14.0625
130 px	0.5157	2.0630	4.1260	6.1890	8.2520	10.3149	12.3779	14.4409	16.5039
140 px	0.5981	2.3926	4.7852	7.1777	9.5703	11.9629	14.3555	16.7480	19.1406
150 px	0.6866	2.7466	5.4932	8.2397	10.9863	13.7329	16.4795	19.2261	21.9727
160 px	0.7813	3.1250	6.2500	9.3750	12.5000	15.6250	18.7500	21.8750	25.0000
170 px	0.8820	3.5278	7.0557	10.5835	14.1113	17.6392	21.1670	24.6948	28.2227
180 px	0.9888	3.9551	7.9102	11.8652	15.8203	19.7754	23.7305	27.6855	31.6406
190 px	1.1017	4.4067	8.8135	13.2202	17.6270	22.0337	26.4404	30.8472	35.2539
200 px	1.2207	4.8828	9.7656	14.6484	19.5313	24.4141	29.2969	34.1797	39.0625
220 px	1.4771	5.9082	11.8164	17.7246	23.6328	29.5410	35.4492	41.3574	47.2656
240 px	1.7578	7.0313	14.0625	21.0938	28.1250	35.1563	42.1875	49.2188	56.2500
260 px	2.0630	8.2520	16.5039	24.7559	33.0078	41.2598	49.5117	57.7637	66.0156
280 px	2.3926	9.5703	19.1406	28.7109	38.2813	47.8516	57.4219	66.9922	76.5625
300 px	2.7466	10.9863	21.9727	32.9590	43.9453	54.9316	65.9180	76.9043	87.8906
320 px	3.1250	12.5000	25.0000	37.5000	50.0000	62.5000	75.0000	87.5000	100.0000
340 px	3.5278	14.1113	28.2227	42.3340	56.4453	70.5566	84.6680	98.7793	112.8906
360 px	3.9551	15.8203	31.6406	47.4609	63.2813	79.1016	94.9219	110.7422	126.5625
380 px	4.4067	17.6270	35.2539	52.8809	70.5078	88.1348	105.7617	123.3887	141.0156
400 px	4.8828	19.5313	39.0625	58.5938	78.1250	97.6563	117.1875	136.7188	156.2500
420 px	5.3833	21.5332	43.0664	64.5996	86.1328	107.6660	129.1992	150.7324	172.2656
440 px	5.9082	23.6328	47.2656	70.8984	94.5313	118.1641	141.7969	165.4297	189.0625
460 px	6.4575	25.8301	51.6602	77.4902	103.3203	129.1504	154.9805	180.8105	206.6406
480 px	7.0313	28.1250	56.2500	84.3750	112.5000	140.6250	168.7500	196.8750	225.0000
500 px	7.6294	30.5176	61.0352	91.5527	122.0703	152.5879	183.1055	213.6230	244.1406
512 px	8.0000	32.0000	64.0000	96.0000	128.0000	160.0000	192.0000	224.0000	256.0000

Discussion of Royalties

One of the potential future features which I consider the most important is the ability for artists and other creators to receive royalties for the sale of their on-chain content.

You can find an introductory discussion of royalties in the previous "**Chronology of Planned Features**" section.

Unfortunately, implementing royalties requires a large number of prerequisite features. Namely, a set of contracts must be built in order to:

- Hold a ledger pointing to all data deployed on-chain through the platform, as well as relevant metadata such as the creator's wallet address
- Support the "tokenization" of on-chain data
- Track the supply and ownership of all tokenized data
- Facilitate "swaps" of these tokens, held by an "escrow contract" similar to the HicEtNunc swap functionality
- Support sales on the secondary market, including the splitting of revenue between the seller and royalty recipient (creator)

As is evident from the list above, there are a large number of required development activities to get royalties working. Regardless, I still think it's a highly important feature.

One of the biggest positive attributes of on-chain art storage is its permanence; it will always be available as long as the Tezos network exists!

For art that can stand the test of time, the artist should have a guarantee that their royalty revenue will be permanent as well.

Luckily, despite the large number of prerequisites, I think it's very feasible to build an efficient and robust royalty system for this project.

If you have strong opinions on royalties, be sure to fill out **the survey!**

Final note about royalties:

During the "Hicathon" event in late May 2021, the HicEtNunc platform announced an in-progress feature for splitting revenue (primary sales and royalties) between multiple users. **If this feature goes public on HEN soon, it will significantly lower the amount of programming required to develop a functional royalty system (along with other sale-related features) for ArtContracts.**

Discussion of Open-Sourcing

I am a very strong believer in open-source software. When it comes to my personal GitHub (under my real name, not the MathMakesArt pseudonym) I have open-sourced nearly every single one of my dozens of personal projects. In cases where I use private repositories, it's almost always due to storage of some confidential information.

This project represents my first ever attempt to monetize my own software development efforts. I do hope to earn some revenue in exchange for providing this DApp, but I also don't want any sort of personal financial incentive to get in the way of innovation within the Tezos community at large.

There is so much potential in the world of on-chain art, and I would love to see Tezos shine in that regard. It is my desire that the code I produce should be accessible to others who are learning blockchain development and/or attempting to build their own applications.

That being said, I don't want to enable any would-be competitors to just walk up and clone my project.

As a solution, I will be open-sourcing this project on a delayed schedule.

I am setting a personal rule for myself and any future developers who join this project:

As soon as a piece of code is integrated into a public release, whether it be a prototype NFT or a web app, a countdown of 12 months begins. The code must be published as open-source software on or before that deadline.

This will ensure that any and all innovations created by developers of this project will eventually belong to the entire Tezos community.

I am very open to community suggestions on this topic. If you have opinions, please make your voice heard through **the survey!**