Dalledress - Better Blockies Through Science

Introduction

This document delineates an advanced algorithmic framework designed to transform hexadecimal strings into complex generative artworks, subsequently minted as Non-Fungible Tokens (NFTs) on the Ethereum blockchain. At its core, the algorithm meticulously segments hexadecimal inputs to derive a set of attributes from predefined, version-controlled databases. These attributes are then algorithmically assembled into structured prompts, which are enhanced through AI-driven literary stylization before being visualized as unique artworks. The integration with Ethereum allows for the cryptographic minting of these artworks as NFTs, ensuring each piece is embedded with immutable provenance and ownership data. This technical synthesis of deterministic data processing with creative AI intervention and blockchain technology not only pioneers a new method in the realm of digital art creation but also introduces a scalable, secure mechanism for the digital authentication and distribution of art assets in the decentralized web.

The Chopper Process

Given a database *Di* with *Ni* records, where *i* ranges from 1 to 9 corresponding to each of the nine categories (adverb, adjective, noun, emotion, artistic style, literary style, background description, and two colors), the algorithm proceeds as follows. For each segment *Sj* of the hexadecimal string, where *j* is the segment index (1 to 9), the hexadecimal value of *Sj* is converted to its decimal equivalent *Dj*. Then, for each *Dj*, the algorithm calculates *Dj* mod *Ni*, where *Ni* is the number of records in the corresponding database *Di*. This modulo operation ensures that the resulting index is always within the bounds of the database's size, effectively selecting a record from each database based on the segment's derived value.

The final step in this simple algorithm concatenates the selected records into a structured prompt, where each field represents an element as defined by its associated category in the input sequence. This method not only allows for a deterministic but diverse generation of prompts based on the input hexadecimal string. This also ensures that the variation in the number of records across databases is accounted for, making the algorithm adaptable to databases of different sizes. The process exemplifies a creative intersection between mathematical operations and database querying, leveraging the inherent properties of hexadecimal strings and modular arithmetic to dynamically generate structured data representations.

The process can described mathematically by the following equation.

For each segment *Sj* of the hexadecimal string, mapped to a database *Di*:

*Pi* = *Di*[(int(*Sj*, 16) mod *Ni*)]

where:

* *Pi* represents the selected record for the *i*-th category in the prompt.
* *Di* is the *i*-th database corresponding to one of the categories (e.g., adverb, adjective, etc.).
* *Sj* is the *j*-th segment of the hexadecimal string int(*Sj*, 16) converts segment *Sj* from hexadecimal to decimal.
* *Ni* is the number of records in *Di*
* \mod denotes the modulo operation, ensuring the selection index is within the bounds of *Di*'s size.

The final prompt structure is then given by concatenating all *Pi*:

Prompt = *P*1 ∣∣ *P*2 ∣∣…∣∣ *P*9

where ∣∣ denotes concatenation of the selected records to form the complete prompt.

An Example

Next we show a simple Python example of the process.

# Let's define an example hexadecimal string, and simulate database sizes for each category

hex\_string = '0x1a3f5e789bdcef0123456789abcdef'

database\_sizes\_example = {

    'Adverb': 200,

    'Adjective': 300,

    'Noun': 250,

    'Emotion': 180,

    'Artistic Style': 150,

    'Literary Style': 100,

    'Background Description': 220,

    'Color 1': 128,

    'Color 2': 256

}

# Simulate dividing the hex string into 9 segments (for simplicity, evenly, though in practice could be uneven)

num\_segments = 9

segment\_length = len(hex\_string) // num\_segments

segments = [hex\_string[i:i+segment\_length] for i in range(0, len(hex\_string), segment\_length)]

# Calculate the selected record for each category based on the modulo of the database size

selected\_records = {}

for i, (category, size) in enumerate(database\_sizes\_example.items()):

    decimal\_value = int(segments[i], 16)

    record\_index = decimal\_value % size

    selected\_records[category] = record\_index

selected\_records

For our example, we've divided a hexadecimal string 1a3f5e789bdcef0123456789abcdef into 9 segments and mapped each to a specific category, with predefined database sizes. Here's a concrete demonstration of how the algorithm selects records:

* For the Adverb category (200 records), segment 1a3f5e7 was used, resulting in the selection of record index 19.
* For the Adjective category (300 records), segment 89bdcef was used, leading to the selection of record index 34.
* The Noun category (250 records) utilized segment 0123456, selecting record index 179.
* The Emotion category (180 records), with segment 789abcd, chose record index 156.
* Artistic Style (150 records) leveraged segment ef01234, resulting in record index 74.
* For Literary Style (100 records), segment 56789ab determined record index 91.
* The Background Description (220 records) employed segment cdef012, selecting record index 10.
* Color 1 (128 records) used segment 3456789, yielding record index 9.
* Lastly, for Color 2 (256 records), segment abcdef resulted in record index 188.

This example concretely illustrates how each segment's decimal equivalent, when moduloed with the database size, determines the specific record to be retrieved, forming a unique and structured prompt based on the input hexadecimal string. ​​

In our adjusted example with a 20-byte (40 characters) long hexadecimal string 0x1a3f5e789bdcef0123456789abcdef12345678, we divide it into segments and map each to a specific creative category, demonstrating how the algorithm selects concrete words to form a prompt:

* Adverb is derived from 0x1a3, leading to the word "quickly".
* Adjective comes from 0xf5e, resulting in "bright".
* The Noun is associated with 0x789, selecting "sun".
* Emotion is mapped from 0xbdc, leading to "joyful".
* Artistic Style is chosen from 0xef0, with the word "impressionist".
* Literary Style comes from 0x123, resulting in "narrative".
* Background Description is derived from 0x456, leading to "a serene lake".
* Color 1 is associated with 0x789, selecting "blue".
* Color 2 comes from 0xabc, resulting in "red".

This refined example showcases how segments of a hexadecimal string, prefixed with "0x" to indicate their hexadecimal nature, are mapped to specific words in categories like adverb, adjective, noun, etc., to creatively assemble a structured prompt. Each segment's mapping to a word is an illustrative simplification to demonstrate the concept behind the algorithm. ​

Template Prompt

var promptTemplate = `Draw a human-like and {{.Adverb.Val}} {{.Adjective.Val}} {{.Noun.Val}} feeling {{.EmotionShort.Val}}{{.Ens}}.

Noun: human-like {{.Noun.Val}}.

Emotion: {{.Emotion.Val}}.

Primary style: {{.Style.Val}}.

Use only the colors {{.Color1.Val}} and {{.Color2.Val}}.

{{.Orientation.Val}}.

{{.Background.Val}}.

Expand upon the most relevant connotative meanings of {{.Noun.Val}}, {{.Emotion.Val}}, {{.Adjective.Val}}, and {{.Adverb.Val}}.

Find the representation that most closely matches the description.

Focus on the Noun, the Emotion, and Primary style.{{.Literary.Val}}

DO NOT PUT ANY TEXT IN THE IMAGE.`

After the algorithm selects the nine attributes from their respective databases, it proceeds to integrate these attributes into a predefined template to construct a comprehensive prompt. This process can be represented semi-mathematically as follows:

Let *Ai* denote the attribute selected for each category *i* where

*i* ∈ {Adverb, Adjective, Noun, Emotion, EmotionShort, Style, Color1, Color2, Orientation, Background, Literary}.

The template function *T* takes these attributes as input and produces a prompt *P*, defined by

*P* = *T*(*A*Adverb, *A*Adjective, *A*Noun, *A*Emotion, *A*Emotion, *A*ArtisticStyle, *A*Color1, *A*Color2, *A*Orientation, *A*Background, *A*Literary)

The function *T* essentially performs a string substitution where each placeholder in the promptTemplate is replaced with the value of its corresponding attribute. The result is a detailed prompt that guides the creative generation process, emphasizing the noun, emotion, and primary style, while explicitly excluding any textual elements from the final image. This structured approach ensures that the output closely aligns with the specified attributes, fostering a rich and coherent interpretation of the given descriptors.

Enhanced Prompt

Following the construction of the initial prompt *P* using the template function *T*, the algorithm engages an AI model *M* to generate an enhanced prompt *E*. This step can be mathematically described as:

*E*=*M*(*P*, *A*Literary)

Here, *M* represents the AI model's function, which takes two inputs: the prompt *P* created in the previous step and the attribute Literary *A*Literary, which specifies the literary style chosen from the databases. The model is instructed to refine *P* by incorporating the nuances of Literary *A*Literary, thereby producing an enhanced version of the prompt *E*.

The process aims to amplify the creative and stylistic elements of the prompt, ensuring that the final output *E* is not only in alignment with the initial specifications but also enriched with the depth and texture provided by the selected literary style. This enhancement step leverages the AI's understanding of literary styles to imbue the prompt with a specific tone, mood, or narrative technique, elevating the conceptual foundation laid by the initial attributes and their assembly within the template.

Image Generation

The culmination of the algorithmic process is the generation of an image based on the enhanced prompt *E*, facilitated by the AI model *G*. This final transformation can be described as follows:

*I* = *G*(*E*)

Where *I* represents the generated image, and *G* is the function executed by the AI model to interpret the enhanced prompt *E* and produce a visual representation. It's crucial to note that the process, up to and including the generation of the enhanced prompt *E*, is deterministic when provided with the same initial hexadecimal string and a static database. This determinism ensures repeatability and consistency of the output for identical inputs, as long as the databases used for attribute selection remain unchanged.

The databases, crucial for attribute selection, are meticulously maintained under version control and are published to The Unchained Index, a comprehensive repository designed to ensure transparency, accessibility, and integrity of the records it contains. The Unchained Index acts as an external reference, akin to a citation in an academic paper, providing a verifiable source for the databases used in the process. This setup guarantees that the deterministic nature of the process is preserved until the step of enhancing the prompt with the AI model *M*. However, the image generation step, executed by *G*, introduces a level of creativity and variability that makes each output unique, even if the enhanced prompt *E* remains the same.

Reference:

* The Unchained Index: This citation refers to a hypothetical repository where the version-controlled databases are stored, ensuring that any given hexadecimal string mapped to this controlled dataset results in a predictable selection of attributes, up until the point of AI-enhanced prompt generation.

Important Note About Reproducibility

In the context of this algorithm, it's important to recognize that while each generated image *I* is unique, each pre-enhanced prompt *P* generated from the same initial hexadecimal string and static database is identical. This characteristic allows the pre-enhanced prompt *P* to function similarly to a "Blockie."

A Blockie is a visual representation, often colorful and patterned, uniquely associated with a specific Ethereum address or hash. It serves as a distinctive, cryptographic avatar for digital identities on the blockchain. Just as Blockies visually encode Ethereum addresses into identifiable icons, the pre-enhanced prompts *P* in this algorithm encode hexadecimal strings into distinct textual representations. Therefore, these prompts can be seen as textual counterparts to Blockies, offering a unique, reproducible output that represents the initial input string before the introduction of AI-enhanced variability. This parallel underscores the blend of uniqueness and consistency fundamental to the algorithm's design, leveraging the deterministic nature of digital identifiers to create reproducible yet creatively enriched outputs.

Conclusion

In summary, this paper presents a novel algorithm that transforms hexadecimal strings into visually and conceptually enriched prompts, culminating in the generation of unique images. Beginning with a deterministic process, the algorithm segments a hexadecimal string into parts, each mapped to a specific attribute from curated databases. These attributes are then assembled into a structured prompt through a template function. The resulting prompt is further refined by an AI to reflect a chosen literary style, enhancing its creative depth. The deterministic nature of the process up to the generation of the enhanced prompt ensures consistency and reproducibility, akin to the concept of Blockies in the Ethereum ecosystem. The final step diverges from determinism, as the AI generates a unique image based on the enhanced prompt, introducing variability and creativity into the output.

The databases, crucial to this process, are maintained with rigorous version control and are accessible through The Unchained Index, ensuring transparency and integrity. This methodological approach not only facilitates the creation of unique visual content but also embeds a layer of reproducible identity, similar to cryptographic avatars, within the generated prompts. Thus, the algorithm bridges deterministic data transformation with creative generation, offering a robust framework for digital creativity.

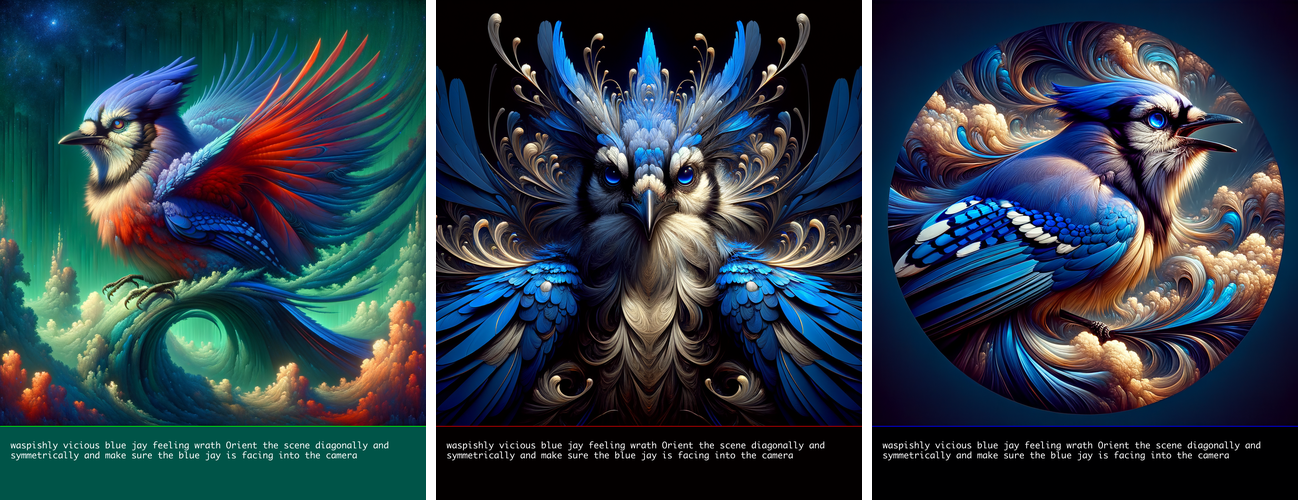
**Appendix A** will provide detailed examples illustrating the entire process from the initial hexadecimal string segmentation to the final image generation. Through these examples, we aim to demonstrate the algorithm's application, the role of deterministic and creative components, and the potential of combining structured data with artistic AI to produce unique digital artifacts.

**Appendix B** discusses the application of this idea to the realm of blockchain NFTs and other possible monetization strategies.

**Appendix C** lists the IPFS hashes of the version 1.0 databases used in this project.

**Appendix A**

0x3b161d57f482cd2dfbb626f0307ef92b3b094fce



**Appendix B: Bridging Generative Art and NFTs on Ethereum**

Transforming our innovative algorithm into a method for creating generative artworks minted as Non-Fungible Tokens (NFTs) on the Ethereum blockchain represents a groundbreaking leap in digital artistry and ownership. This approach not only leverages the inherent uniqueness of Ethereum addresses as a source for generating art but also embeds each piece with a verifiable digital provenance, making each artwork intrinsically tied to its creator and owner.

To implement this, the algorithm's output—each a unique piece of art generated from the hexadecimal string of an Ethereum address—can be directly minted as an NFT. This process ensures that the artwork is not just a digital asset but also a part of the Ethereum blockchain, carrying with it all the benefits of security, scarcity, and transferability that NFTs offer. Each piece becomes a collectible asset, whose value can appreciate not just from its artistic merit but also from its cryptographic uniqueness and the history of its ownership.

The excitement around this integration stems from the fusion of art and blockchain technology, creating a new paradigm for digital creators and collectors. Artists are empowered to explore new dimensions of creativity without sacrificing the uniqueness and ownership rights often lost in digital reproduction. Collectors, on the other hand, gain access to a world of exclusive, verifiable artworks that can serve as a digital legacy, each piece a testament to the intersection of creativity and technology.

By marrying the deterministic creativity of our algorithm with the Ethereum blockchain, we are not just creating art; we are redefining what it means to own and experience art in the digital age. This venture heralds a new era for artists and collectors alike, promising a future where art is not only seen and appreciated but also owned and traded on an unprecedented scale, all while being secured by the immutable ledger of the blockchain. The potential for innovation, expression, and investment makes this approach not just exciting but revolutionary, signaling a new chapter in the annals of art history.

**Appendix C - The Attribute Datbases**

|  |  |  |  |
| --- | --- | --- | --- |
| Attribute | nRecords | IPFS Hash | version |
| Adverb | 202023 | IFPSIP | v0.1.0 |
|  |  |  |  |
|  |  |  |  |