Tutorial: Create Generative NFT Art with Rarities

Rounak Banik Follow
Sep 27 · 10 min read

Cryptopunks by Larva Labs

Introduction

Marquee NFT projects like *Cryptopunks* and *Bored Ape Yacht Club* have generated hundreds of millions of dollars in revenue, and have made several of their owners millionaires.

What the aforementioned projects (and most other successful NFT projects today) have been in common is that they are *PFP projects*. This means that they usually are a collection of 10,000+ avatars where each avatar is unique and has a set of traits.

In this tutorial, we will show you how to generate a collection like this with custom rarities. We will be using a library created by the *Scrappy Squirrels* team to accomplish this. At the end of this tutorial, you would have generated your own custom avatar collection with associated metadata.

Pre-Requisites

Python and pip installed on your computer

Our library is written in Python so you will need to have this installed on your computer. You will also need pip which will install important packages for us.

Go to this website and download the latest version of Python. **You do not need to know how to program (in Python or otherwise) to follow this tutorial.**

An Artist (preferred but not required)

You will also need an artist who knows their way around digital art to create your own custom collection. However, this is not required to follow this tutorial. We will be providing you with certain test images to play around with.

The Scrappy Squirrels Collection

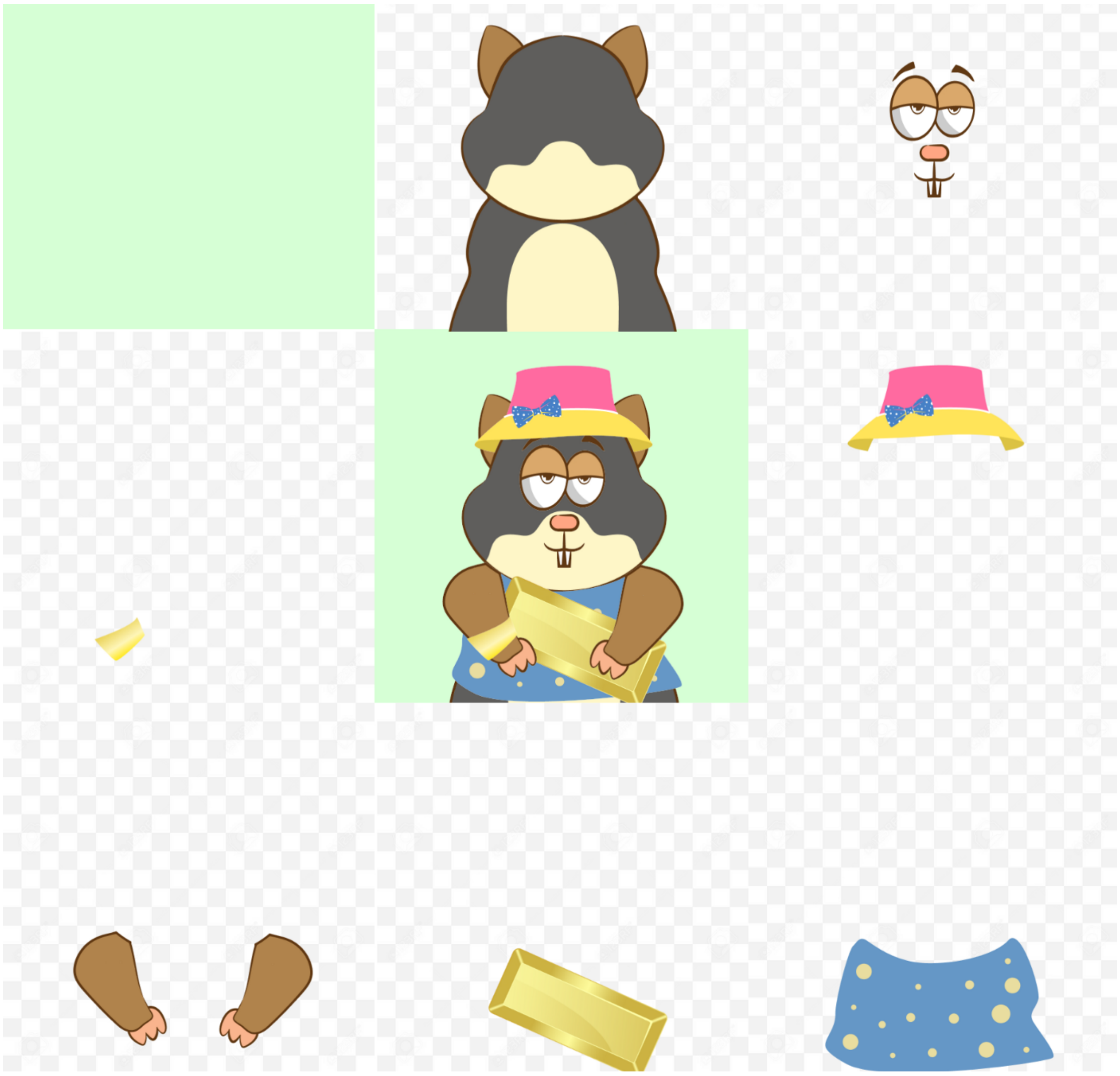
As part of this tutorial, we will walk you through the process of creating the *Scrappy Squirrels* NFTs, a real project that we have launched. This tutorial (and every subsequent one) has been created as part of our roadmap goals to make NFTs and blockchains more accessible to people. Do check out our Discord for more details. (Go on, we will wait :))

The squirrels have been generated using over 85 traits. Here are a few samples.

Sample Scrappy Squirrels

The Generation Process

The squirrels that you see above were generated by stacking PNG images on top of one another. Although no blue-chip NFT projects describe how they generate their art, we are certain that this is what they do too. Almost every NFT avatar that you see is a set of stacked PNG images (which makes the claims that they are *just JPEGs* false. Checkmate, NFT critics).


Stacking Trait Images to generate the final squirrel

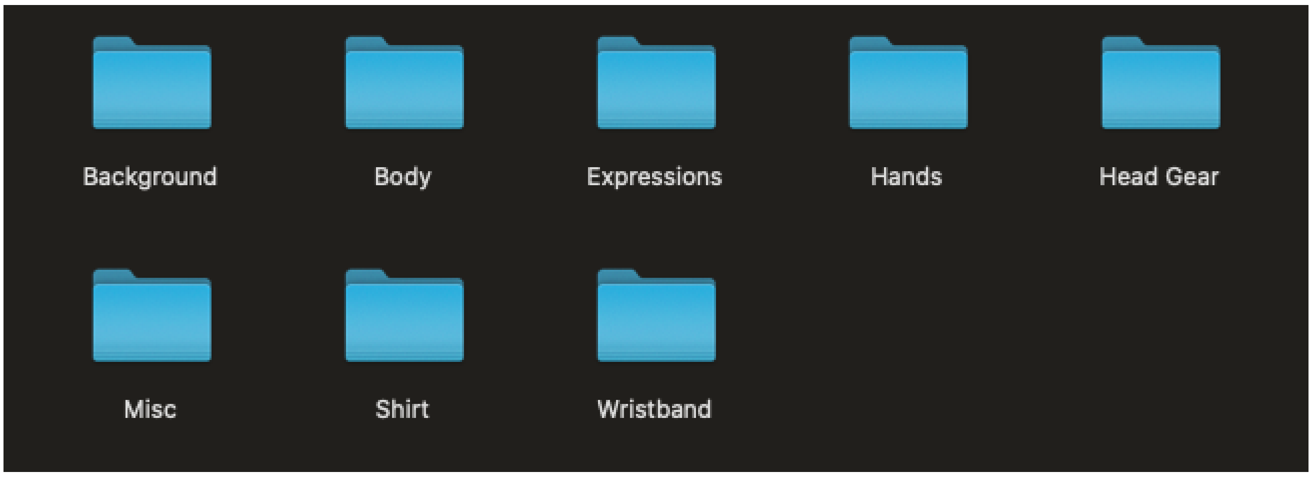
Starting from the top right, if you stack every trait image clockwise, one after the other, you will end up with the image in the center. Here are few things to note:

1. Each trait image (and the final squirrel avatar) is of exactly the same dimension.
2. Apart from the background trait (which is the first trait), every other trait image has a transparent background.
3. The trait images must be stacked in order to get the correct squirrel avatar (i.e clockwise from top-right).
4. The trait images are drawn in such a way that their positioning makes sense with respect to all other traits.

5. We can swap any trait with another trait of the same category (for instance, a red shirt for a blue shirt). Therefore, in this case, if we had 10 traits for each category of trait, we could theoretically produce 100 million distinct squirrels.

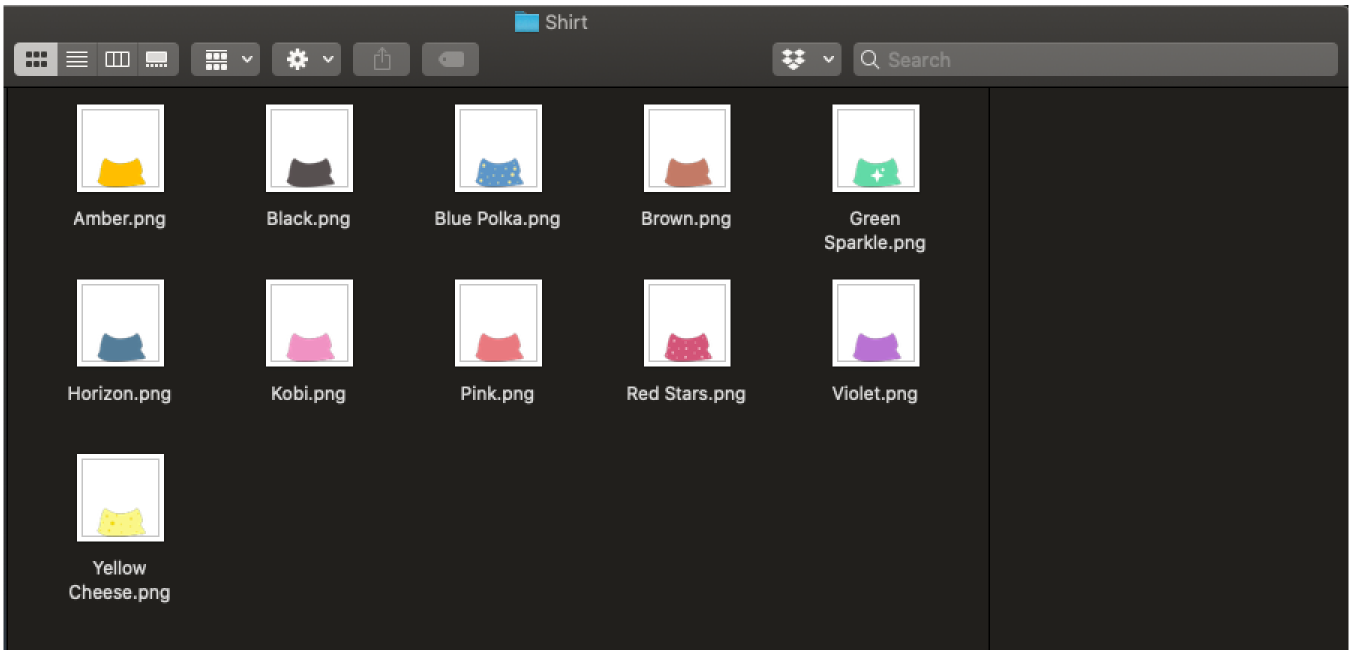
Therefore, the artist's job is to create multiple images of various trait categories. You can have as many or as few trait categories as you want. Do keep in mind though that the number of possible combinations increases exponentially with the number of traits categories.

In the *Scrappy Squirrels* project, we created 8 trait categories.



Trait Categories

Each trait category had a varying number of trait images. For instance, we had 11 different shirts to work with.



Now, it's your turn. You will need to decide on trait categories that you want to work with and generate trait images for each category. Make sure they satisfy the conditions mentioned above (should be of the same dimension, should be correctly positioned, etc). Also, make sure you name the trait images appropriately. What you name your image is what will appear in the metadata file.

Once you are done with this, we are now ready to use the library to generate our collection automatically! If you are not an artist (or do not have access to one), don't worry! We have some sample images that you can play around with.

NOTE:
At present, the library is only capable of handling PNG images. We will be adding support for other media types soon.

Download repository and install required packages

Our generative art library is available for free on GitHub. Go ahead and download it.

Once you've downloaded the repository, open your Terminal or Command Prompt, and run the following command:

```
pip install Pillow pandas progressbar2
```

Running this command will install three important Python packages that our library depends on:

1. **Pillow:** An image-processing library that will help us stack trait images.
2. **Pandas:** A data analysis library that will help us in generating and saving our image metadata.
3. **Progressbar:** A library that will tell us about the progress when the image generation takes place.

Add your custom assets

In the *generative-art-nft* repository that you downloaded, you will see that there is an *assets* folder. If you have your custom trait artwork available with you, go ahead and replace the contents of this folder with your assets. In our case, our *assets* folder had 8 subfolders representing categories named appropriately (see above), and each subfolder had trait images of that particular category.

If you do not have custom artwork, leave the default *assets* folder as is.

Configure the config.py file

This is the last (and perhaps, the most important step) before we can generate our avatar collection. Open the *config.py* file and fill it up according to the instructions below.

The config file consists of a single Python variable called CONFIG. CONFIG is a Python list (encapsulated by []). It contains a list of trait categories in the order that they need to be stacked. **The order here is extremely important.** Here is a sample configuration.

```
CONFIG = [
    {
        'id': 1,
        'name': 'background',
        'directory': 'Background',
        'required': True,
        'rarity_weights': None,
    },
    {
        'id': 2,
        'name': 'body',
        'directory': 'Body',
        'required': True,
        'rarity_weights': 'random'
    },
    {
        'id': 3,
        'name': 'eyes',
        'directory': 'Expressions',
        'required': True,
        'rarity_weights': None
    },
    {
        'id': 4,
        'name': 'head_gear',
        'directory': 'Head Gear',
        'required': False,
        'rarity_weights': None
    },
    {
        'id': 5,
        'name': 'clothes',
        'directory': 'Shirt',
        'required': False,
        'rarity_weights': None
    },
    {
        'id': 6,
        'name': 'held_item',
        'directory': 'Misc',
        'required': True,
        'rarity_weights': None,
    },
    {
        'id': 7,
        'name': 'hands',
        'directory': 'Hands',
        'required': True,
        'rarity_weights': None,
    },
    {
        'id': 8,
        'name': 'wristband',
        'directory': 'Wristband',
        'required': False,
        'rarity_weights': [100, 5, 5, 15, 5, 5, 15, 15, 5, 1]
    },
]
```

Each trait category is represented as a Python dictionary (encapsulated by {}). All that needs to be done is define these trait category dictionaries in order in the CONFIG list.

A trait category dictionary has 5 keys that it needs. These are *id, name, directory, required, and rarity_weights*. When creating a new layer (or replacing an existing one), make sure all these keys are defined.

This is how you go about assigning value to each key.

1. **id:** The layer number. For instance, if the body is the second trait category (or layer) that needs to be stacked, it will have an id of 2. Please note that layers must still be defined in the correct order.

2. **name:** The name of the trait category. This can be anything you choose it to be. It will appear in the metadata.

3. **directory:** The name of the folder inside *assets* that contain images of that particular trait category.

4. **required:** If this category is required for every image. Certain trait categories (like background, body, and eyes) must appear in every avatar whereas certain other categories (like headgear, wrist band, or clothes) can be optional. We strongly recommend that you set the first layer's *required* value to true.

5. **rarity_weights:** This category will determine how common (or rare) your traits are going to be. Check the next section for more details.

Configuring rarity weights

The *rarity_weights* key can take three values: None, 'random', or a Python list. Let's explore each value one by one.

None

If you set the rarity_weights value to *None,* each trait will be assigned an equal weight. Therefore, if you have 5 traits, each trait will appear in roughly 20% of the avatars.

In case *required* is False, it will be equally likely to not get that particular trait at all. In the previous case, if the *required* property was set to false, then each trait would appear in roughly 16.6% of the avatars. Another 16.6% of avatars would not have that particular trait at all.

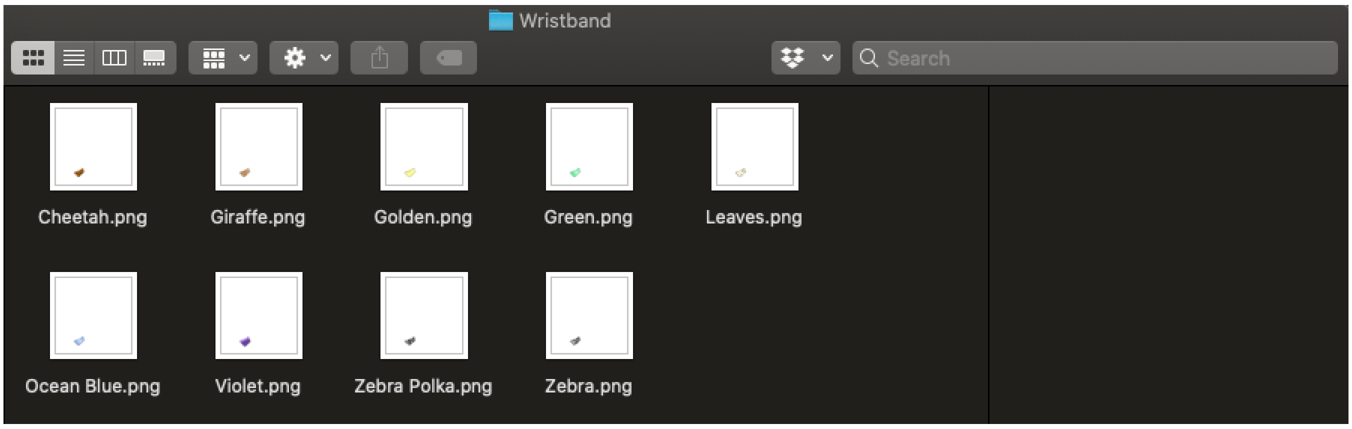
'random'

Setting *rarity_weights* to 'random' (note the parenthesis) would randomly assign weights to each category. **We strongly recommend you do not use this feature. Always resort to either equal or custom user-defined rarity.**

Python List

This is probably the most common way of assigning rarity weights.

The first thing to do is to go to your trait category folders and sort the trait images by Name. For instance, sorting the *Wristbands* folder will yield this for us:



You can see that we have 9 different kinds of wristbands. Now, we need to define a Python list (encapsulated by []) where each number represents a weight assigned to a particular trait in ascending order.

If *required* is set to True, then the number of weights should be equal to the number of traits for that category. If *required* is set to False, then the number of weights should be equal to the number of traits plus one.

In our case, if wristbands were required, we would define nine weights in the list and if it wasn't required, we would define ten weights. In the latter case, the first weight would be the weight associated with not having the wristband at all.

Let's take a look at the *rarity_weights* we defined for Wristbands.

```
[100, 5, 5, 15, 5, 5, 15, 15, 5, 1]
```

Since wristbands aren't required, we have set ten weights (nine plus one). The first weight is the weight associated with not having a wristband at all. The second weight is associated with the *Cheetah* band, the third weight is associated with the *Giraffe* band, and so on. Note the alphabetical order here.

The higher the weight, the more common a particular trait is. For instance, *Cheetah* has a weight of 5, and not having a band has a weight of 100. This means that having a *Cheetah* band is 20 times rarer than not having a band at all.

Generating the collection

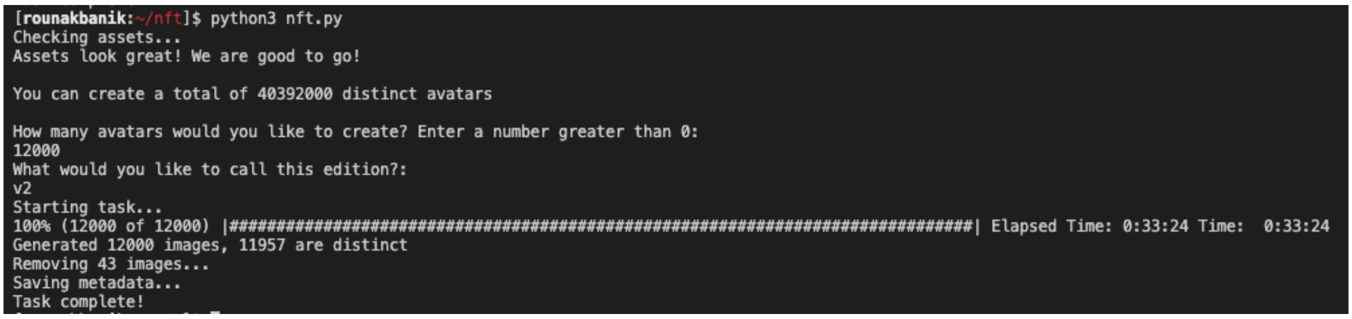
Once you've configured the config.py file, it is now time to generate your collection. Open up your Terminal (or Command Prompt) and navigate to the *generative-art-nft* folder (using the *cd* command).

Now, run the following command:

```
python nft.py
```

Running this command will initiate the image generation program. It will first check that the config.py file is valid. Next, it will tell you about the total number of distinct possible combinations.

It will then ask you how many avatars you'd like to create. We suggest creating 20% more than what you want to create so you have plenty left over even after the removal of duplicates. In our case, we chose to create 12,000 avatars although we wanted 10,000. It will then ask you to name the collection, and will then begin the generation process.



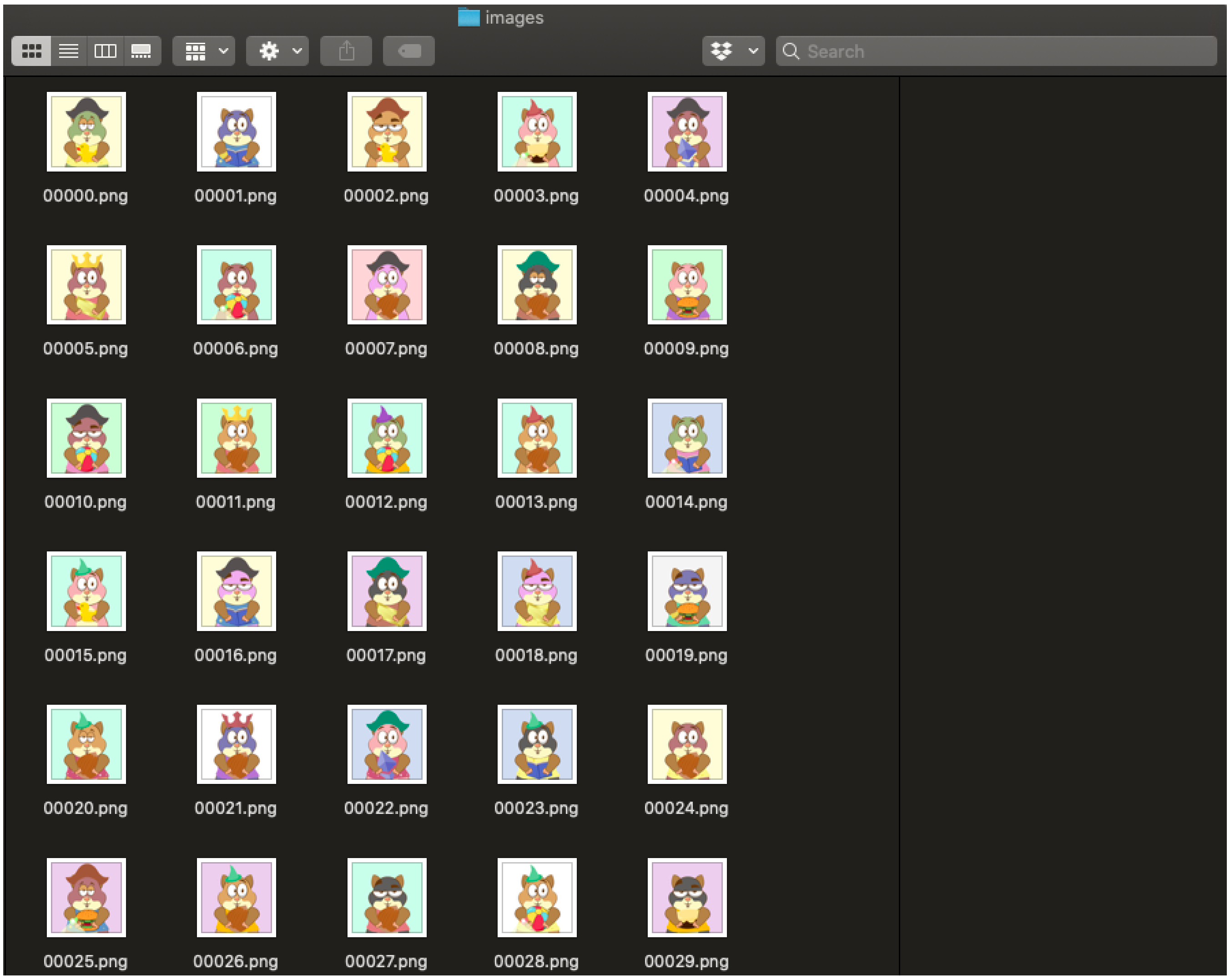
```
[rounakbanik:~/nft]$ python3 nft.py
Checking assets...
Assets look great! We are good to go!

You can create a total of 40392000 distinct avatars

How many avatars would you like to create? Enter a number greater than 0:
12000
What would you like to call this edition?:
v2
Starting task...
100% (12000 of 12000) |################################################################################| Elapsed Time: 0:33:24 Time:  0:33:24
Generated 12000 images, 11957 are distinct
Removing 43 images...
Saving metadata...
Task complete!
```

It took us approximately 30 minutes to generate 11,957 avatars (after removing duplicates). The images and their related metadata will be available in the output folder.

The images folder will look something like this (note that this is only a sample and not the final squirrels that we generated).



The metadata file is a CSV file that you can import into Excel and analyze (for things like which trait is the rarest, which trait combination is the most common, avatar rarity ranking, etc.)

Conclusion

And there you have it! You have generated your very own avatar collection.

So are we now ready to launch the next big NFT project? Not quite. You will need to upload these images to IPFS, allow your users to mint them into NFTs, and create community and buzz around your project.

Some of these words sounding alien? Don't worry. We will be launching tutorials on the aforementioned topics very soon. **Do join our Discord for the latest updates.**

If you have any questions or would like us to add additional features to this library, please reach out to us on our Discord server, or drop them in the comments below. We will try to address as many of them as possible.

Until next time!

About Scrappy Squirrels

Scrappy Squirrels is a collection of 10,000+ randomly generated NFTs on the Ethereum Blockchain. Scrappy Squirrels are meant for buyers, creators, and developers who are completely new to the NFT ecosystem.

The community is built around learning about the NFT revolution, exploring its current use cases, discovering new applications, and finding members to collaborate on exciting projects with.

Join our community here: https://discord.com/invite/5WeaXPjDrb

Nft Blockchain Art Tutorial Python