

BOOK 02 · A FREE GUIDE FROM KURV

# Let Your Agent Draw

BOOK



CODE + AI



Code and AI, step by step. Pick a style, plan the layers, prompt your agent well, check every file and export a ZIP ready for KURV.

10 STEPS

One step per page. Written for creators, and for their agents.

[Apply at kurv.fun/apply](https://kurv.fun/apply)

KURV is testing on Base and Solana, with mainnet coming.

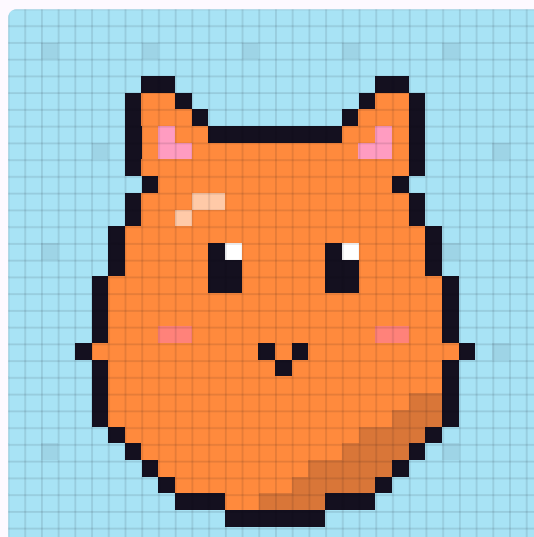
## CHAPTER 01 · THE LOOK

# Pick a style and a canvas

Before your agent draws a single pixel, **decide the look**. A clear style and one fixed canvas size make every later step easy.

Pick one style you can describe in a sentence: chunky pixel art, flat vector stickers, soft painted shapes. Then pick one canvas size and never change it. Pixel art is the friendliest place to start with code, because every trait is a small grid you can scale up cleanly.

|            |                                                    |
|------------|----------------------------------------------------|
| STYLE      | Pixel art, 32 by 32 grid, thick black outline      |
| CANVAS     | Draw at 32 by 32, export scaled up to 1024 by 1024 |
| PALETTE    | 8 to 12 colors, written down as hex codes          |
| BACKGROUND | Transparent on every layer except background       |



Only use art that is yours or properly licensed. Describe a style in your own words, never by naming a living artist.

## CHAPTER 02 · THE PLAN

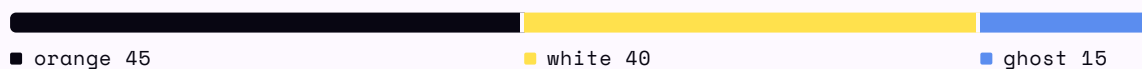
# Plan the layers and the rarity

A collection is a stack of layers. Write the stack down **back to front** before anything gets drawn.

List each layer in order, the traits inside it and how rare each one is. Check the current layer and piece limits on docs.kurv.fun before you plan a big set. Keep the plan in a simple file your agent can read.

TRAITS.JSON

```
{
  "canvas": [1024, 1024],
  "layers": [
    { "name": "background", "traits": { "sky": 50, "night": 35, "gold": 15 } },
    { "name": "body", "traits": { "orange": 45, "white": 40, "ghost": 15 } },
    { "name": "eyes", "traits": { "dot": 60, "sleepy": 30, "laser": 10 } },
    { "name": "hat", "traits": { "none": 50, "beanie": 35, "crown": 15 } }
  ]
}
```

**BACKGROUND****BODY****EYES****HAT**

Numbers are weights. Bigger number, more common. Rare traits stay rare.

## CHAPTER 03 · THE GUIDE

# Write the style guide

Your agent is only as consistent as its brief. Write one style guide and paste it at the top of every chat.

Describe the look in plain words: shapes, outline, palette, light, what the character always has. Say what is not allowed. Your style guide replaces any artist name.

**PROMPT: THE STYLE GUIDE**

```
You are drawing traits for my NFT collection.  
STYLE: pixel art on a 32x32 grid, 1 px black outline,  
flat colors, no gradients, no anti-aliasing.  
PALETTE (only these): #14101F #FFF6EA #FFCF3F #FF7BAC  
#3EC1E8 #FF8A3D #4ADE80 #8B5CF6  
CHARACTER: a round cat, head centered, ears at rows 4-5.  
RULES: every layer is its own transparent PNG, same  
canvas, trait drawn in its final position, nothing  
cropped. Never copy or imitate a named artist.
```

**PALETTE**

## CHAPTER 04 · THE CODE

# Draw layers with code

For pixel art, code is a great brush. Ask your coding agent for a small script that draws each trait on its own grid.

Each layer is a transparent PNG at the same size. The script draws on a tiny grid, then scales up with nearest neighbor so pixels stay crisp.



DRAW\_TRAIT.PY (PYTHON + PILLOW)

```
from PIL import Image, ImageColor
GRID, OUT = 32, 1024
def save_trait(pixels, path):
    # pixels: {(x, y): "#hex"}
    img = Image.new("RGBA", (GRID, GRID), (0, 0, 0, 0))
    for (x, y), hexc in pixels.items():
        img.putpixel((x, y), ImageColor.getrgb(hexc) + (255,))
    img.resize((OUT, OUT), Image.NEAREST).save(path)
crown = {(x, 6): "#FFCF3F" for x in range(11, 21)}
save_trait(crown, "layers/hat/hat_crown.png")
```

## CHAPTER 05 · ONE TRAIT

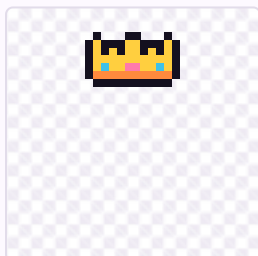
# Prompt a single trait

Start with one trait, not fifty. Get one right, then scale.

Ask for exactly one file, with its layer, its size and its position. Works the same with an AI image tool or a coding agent. Review it on top of the body layer before asking for more.

## PROMPT: A SINGLE TRAIT

```
Using the style guide above, draw ONE trait.  
Layer: hat. Trait: crown.  
Canvas: 32x32, transparent background.  
Position: sits on top of the head, rows 3-7, centered.  
Colors: #FFCF3F with #FF8A3D shading, 1 px black outline.  
Save as layers/hat/hat_crown.png, exported at 1024x1024.  
Then show it stacked on body_orange.png so I can check.
```



hat\_crown.png



on body\_orange.png

One trait, one file, one check. Then the next.

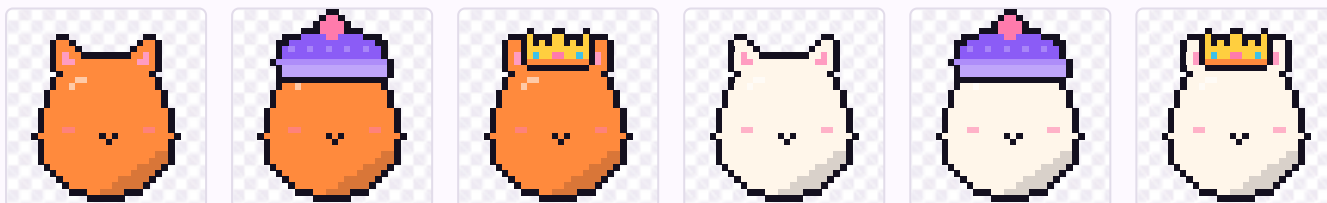
# Ask for the full set

When the first trait looks right, hand over the whole list. Your traits.json is the brief.

Ask the agent to work layer by layer, keep names exactly as listed and stop for review after each layer. Small batches are easier to check than a giant dump.

PROMPT: THE FULL TRAIT SET

```
Read traits.json and the style guide.  
For each layer, in order, draw every trait listed.  
- One transparent PNG per trait, 1024x1024.  
- Save to layers/<layer>/<layer>_<trait>.png  
- Keep head position and outline identical to hat_crown.  
- After each layer, stop and show me a contact sheet  
  of every trait stacked on the base body.  
Do not invent new traits or rename any file.
```



## CHAPTER 07 · THE MATCH

# Keep every trait consistent

Drift is the enemy. Lock a reference stack and make every new trait match it.

Keep one approved image: the base body with one trait from each layer. Attach it to every request and ask the agent to compare before saving.

PROMPT: KEEP IT CONSISTENT

Here is reference.png, my approved stack.  
Match it exactly: same grid, same outline weight,  
same palette, same head position and size.  
Before saving any new trait, compare it to the  
reference and list anything that differs.  
If it differs, redraw it. Do not save it.



reference.png

|                     |                                        |
|---------------------|----------------------------------------|
| <b>SAME GRID</b>    | Every trait on the same 32 by 32 grid  |
| <b>SAME ANCHOR</b>  | Head and eyes never move between files |
| <b>SAME PALETTE</b> | Only the hex codes in the guide        |

## CHAPTER 08 · THE FIX

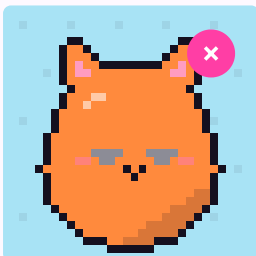
# Fix mistakes fast

Something will come out wrong. Be specific about what, where and how to fix it.

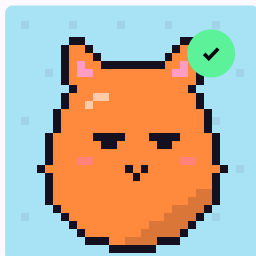
Name the file, describe the problem in one line and say exactly what to keep. Vague feedback like "make it better" gets you a new random image.

PROMPT: FIXING A MISTAKE

```
Fix layers/eyes/eyes_sleepy.png only.  
Problem: the eyes sit 2 px too low and the outline  
is grey instead of #14101F.  
Keep: shape, size, colors of the lids.  
Change: move up 2 px, outline to #14101F.  
Do not touch any other file. Show before and after.
```



before



after

## HALO EDGES

Ask for no anti-aliasing and pure transparency

## WRONG SIZE

Ask to re-export at the exact canvas size

## OVERLAP

Ask to clear pixels outside the trait area

# Check the output

Before you zip, let code check the files. A tiny script catches what tired eyes miss.

Ask your agent to run a check like this over the layers folder and fix anything it flags.

CHECK\_LAYERS.PY

```
from PIL import Image; import pathlib, re
for f in sorted(pathlib.Path("layers").rglob("*")):
    if f.is_dir(): continue
    if f.suffix != ".png": print("not a PNG:", f); continue
    if not re.fullmatch(rf"{f.parent.name}_[a-z0-9-]+", f.stem):
        print("bad name:", f)
    im = Image.open(f)
    if im.size != (1024, 1024): print("wrong size:", f, im.size)
    if im.mode != "RGBA": print("no transparency:", f)
```

|              |                                       |
|--------------|---------------------------------------|
| SIZE         | Every file 1024 by 1024               |
| TRANSPARENCY | Only background is fully filled       |
| NAMES        | layer_trait.png, lowercase, no spaces |
| OVERLAP      | Stack a few samples and look          |

# Folders, names and the ZIP

KURV takes **one ZIP**: one folder per layer, each file named `layer_trait.png`.

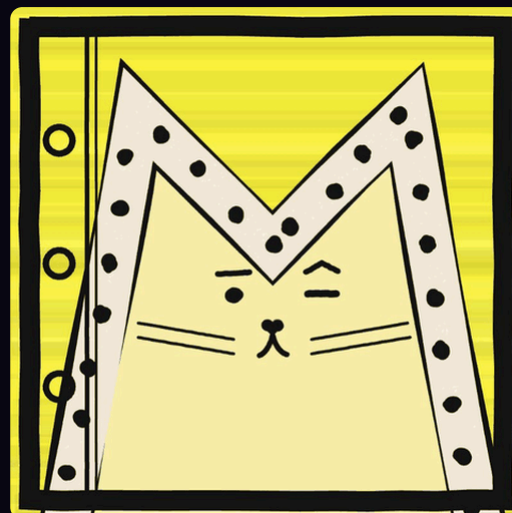
Ask your agent to build the ZIP from your layers folder. Then open it once yourself and look before you upload it to KURV.

```
■ my-collection.zip
  ■ background
    □ background_sky.png
    □ background_night.png
  ■ body
    □ body_orange.png
  ■ eyes
    □ eyes_sleepy.png
  ■ hat
    □ hat_crown.png
```

PROMPT: EXPORT THE ZIP

```
Zip the layers folder as my-collection.zip.
One folder per layer, in stack order.
Files named <layer>_<trait>.png.
No hidden files, no extra folders.
List every file inside when done.
```

Once your launch is approved, the ZIP goes to KURV. Book 03 walks you through every setting.



THE LAST PAGE

# Your traits are **one** **ZIP** away from KURV.

Style guide, trait list, one good trait, the full set, a clean check,  
one ZIP. Early launches on KURV are open by application.

[Apply at \*\*kurv.fun/apply\*\*](https://kurv.fun/apply)[Visit \*\*kurv.fun\*\*](https://kurv.fun)

Made by KURV. Free to read and share. Book 02 of 6. Share it with one creator and their agent.