Digital Constellation Generator

Artist Statement

Humans throughout history have looked at the stars and created stories from them. They have turned a few scattered points of light to become bears, hunters, lions and heroes. This has always fascinated me. It's amazing how easily the human mind turns something random into something familiar. This is what inspired me to start this project. I wanted to explore how this concept of turning familiar shapes into constellations would look in a more modern world, using digital tools rather than ancient storytelling. What would new constellations look like if anyone could create them on demand? I created an artwork that is a small interactive program that allows the viewer to generate a constellation from almost any word they choose. When a user types in a word such as “dog,” “cat,” or “bike (or 7000+ other objects), the program takes an image of that shape and transforms it into a constellation on a black night-sky background. It has elements such as bright anchor starts, faint connecting lines, and scattered background points. All of these elements are on a silhouette chosen by the viewer when they enter their word. The constellation is both playful and plausible, as if it could appear in an astronomy guide that has not been written yet. I wanted to capture both the wonder we associate with the cosmos and the creativity we bring to it when making this.

While the project is fully digital, the ideas behind it are astronomical. Constellations, historically, were not scientific diagrams but cultural artifacts. Before they were objects of scientific study, they were tools for navigation and storytelling. Today even, they occupy an interesting space between science and imagination. My artwork leans into this idea. My artwork uses real ideas from astronomy, such as patterns of major and minor stars, the appearance of a starfield, and the aesthetic of celestial charts. My artwork also highlights how open-ended constellation-making truly is. The stars do not “look like” the shapes we assign them; the shapes come from us. By allowing viewers to generate constellations from everyday objects, the project gently shows how flexible and subjective this act of interpretation can be.

When I was writing the code to create the visuals, I made some choices that reference real astronomical observations. I chose to make the background stars vary in brightness. This is similar to a true night sky where magnitude depends on the distance and luminosity of each star. Since real constellation diagrams rely on just a few imagined lines to hint at a figure, the connecting lines are faint and sparse. The “major” stars in each generated shape are larger and more luminous, inspired by the way constellations typically highlight a few dominant stars such as Betelgeuse or Polaris. Even the black background is intentional; it creates visual contrast while also grounding the artwork in the familiar experience of looking up at a dark sky. These choices help the generated constellations feel cohesive and believable, even though they originate from something as simple as a silhouette of a household object or animal.

I would say that this project is rooted in both curiosity and simplicity. I enjoy creating systems where a small input produces something unexpected. I find that there is something very satisfying about typing a single word and watching it then expand into a visual composition that feels larger than the sum of its parts. The program does not aim to be scientifically accurate or physically realistic; instead, it aims to spark the same imaginative leap that people have been making for thousands of years. It invites the viewer to briefly play the role of a myth-maker or chart-maker, inventing constellations for a universe that functions a little more like a sketchpad.

There is also a personal element in choosing this medium. I am not a traditional studio artist, so my creative process often begins with experimentation: “What would happen if I tried this?” In this case, the “this” was the idea of taking a digital shape and turning it into something cosmic. I wanted to create artwork that felt interactive and alive, something that responded to the viewer instead of sitting still. At the same time, I wanted the final visual to stand on its own as a piece of art, something that could be printed or displayed like a star map. Coding provided the flexibility I needed to blend these goals. It allowed me to design the system, the visual style, and the user experience all at once.

Ultimately, the project is about the connection between imagination and astronomy. It shows how easily we can project meaning onto the unknown, how quickly we turn points of light into something recognizable. It invites viewers to think about how constellations have always reflected culture, storytelling, and identity. By giving the user control over what becomes a constellation, the project turns that ancient practice into something playful and personal. My hope is that viewers walk away with a renewed appreciation for both the vastness of the night sky and the creativity we bring to it.

Figures

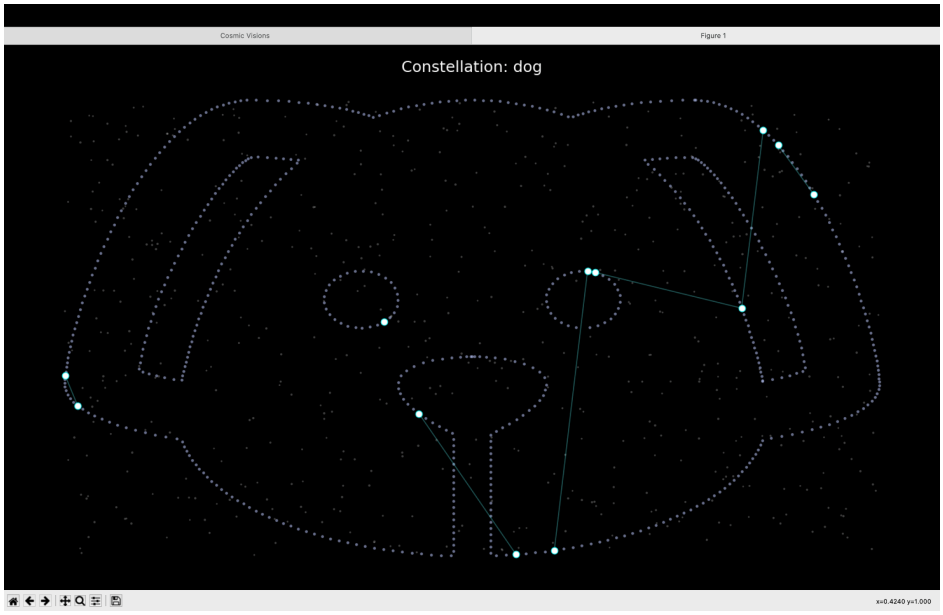

Figure 1. “Dog” constellation generated by the program.

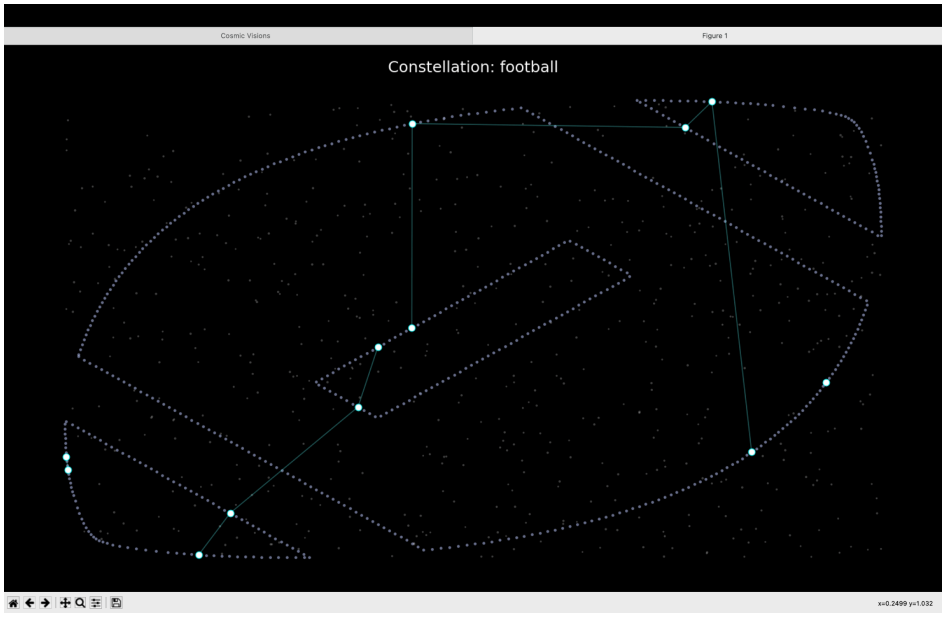

Figure 2. “Football” constellation generated by the program.

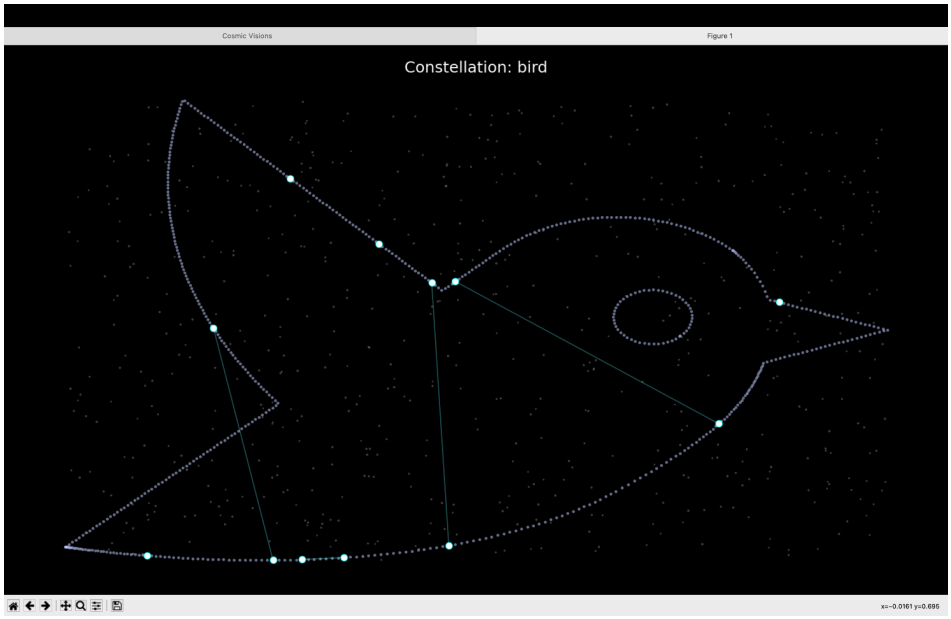

Figure 3. “Bird” constellation generated by the program.

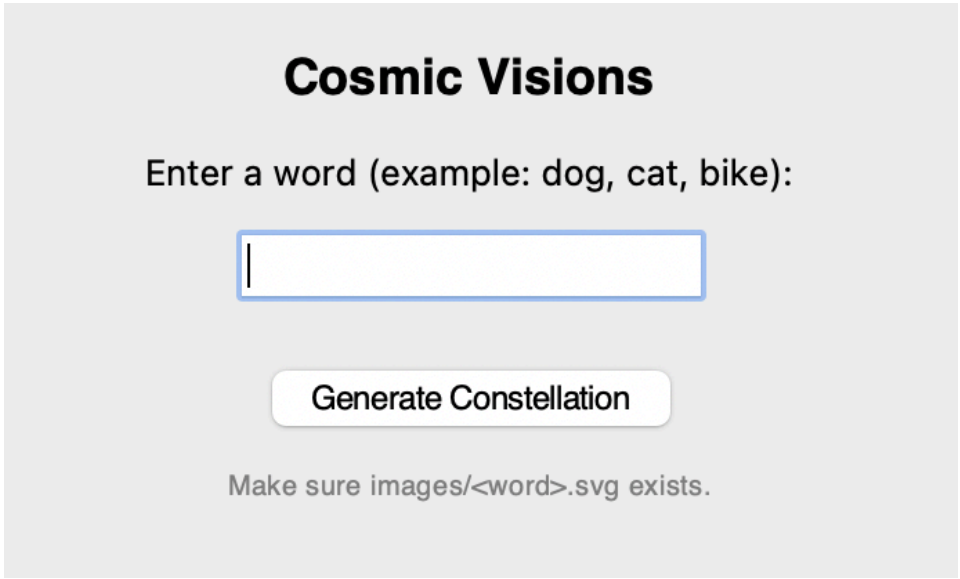

Figure 4. User interface for input.

Video Demonstration

https://drive.google.com/file/d/1hX6b4pjz5Ney0nue_xs-KZYDrd8gzWSc/view?usp=drive_link

Appendix: Program Code (Python)

```
import os

import numpy as np

import matplotlib.pyplot as plt

from svgpathtools import svg2paths

from pathlib import Path

import tkinter as tk

from tkinter import messagebox



# === Settings ===

IMAGE_DIR = "images"



def svg_to_points(svg_path, samples=600):

    """Load SVG and sample points along its paths."""

    paths, _ = svg2paths(svg_path)

    points = []



    for p in paths:

        for i in np.linspace(0, 1, max(2, samples // max(1, len(paths)))):

            try:

                pos = p.point(i)

                points.append((pos.real, pos.imag))
```

```
            except Exception:
                pass

    points = np.array(points)
    x = points[:, 0]
    y = points[:, 1]

    # Normalize
    x = (x - x.min()) / (x.max() - x.min())
    y = (y - y.min()) / (y.max() - y.min())

    return x, y


def draw_constellation(x, y, title):
    """Draw a realistic constellation-style star map."""

    # Background stars
    bg_x = np.random.rand(600)
    bg_y = np.random.rand(600)

    # Major stars
    num_major = max(10, len(x) // 50)
    idx = np.random.choice(len(x), num_major, replace=False)
```

```
major_x = x[idx]

major_y = y[idx]



# Black sky window

plt.figure(figsize=(7, 7), facecolor="black")

ax = plt.gca()

ax.set_facecolor("black")

plt.axis("off")



# Dim random stars

plt.scatter(

    bg_x, bg_y,

    s=np.random.uniform(1.5, 4, size=len(bg_x)),

    color=(1, 1, 1, 0.18),

    zorder=1

)



# Outline stars

plt.scatter(

    x, 1 - y,

    s=6,

    color=(0.75, 0.8, 1.0, 0.45),

    zorder=2

)
```

```
# Major bright stars
plt.scatter(
    major_x, 1 - major_y,
    s=60,
    color="white",
    edgecolors="cyan",
    linewidths=0.8,
    zorder=4
)

# Connect major stars
order = np.argsort(major_x)
cx = major_x[order]
cy = major_y[order]

for i in range(len(cx) - 1):
    if np.random.rand() < 0.65:
        plt.plot(
            [cx[i], cx[i+1]],
            [1 - cy[i], 1 - cy[i+1]],
            color=(0.4, 1.0, 1.0, 0.35),
            linewidth=1.2,
            zorder=3
```

```
            )


    plt.title(f"Constellation: {title}", color="white", fontsize=18)
    plt.tight_layout()
    plt.show()



def on_generate():
    """Triggered when user clicks the button."""
    word = entry.get().strip().lower()

    if not word:
        messagebox.showerror("Error", "Please enter a word.")
        return

    svg_path = Path(IMAGE_DIR) / f"{word}.svg"

    if not svg_path.exists():
        messagebox.showerror("Error", f"No SVG found at: {svg_path}")
        return

    try:
        x, y = svg_to_points(str(svg_path))
        draw_constellation(x, y, word)
```

```
    except Exception as e:
        messagebox.showerror("Error", f"Could not read SVG:\n{e}")



def main():
    root = tk.Tk()
    root.title("Cosmic Visions")

    title_label = tk.Label(root, text="Cosmic Visions", font=("Helvetica", 18, "bold"))
    title_label.pack(pady=(20, 10))

    prompt = tk.Label(root, text="Enter a word (example: dog, cat, bike):")
    prompt.pack()

    global entry
    entry = tk.Entry(root, font=("Helvetica", 14), width=20)
    entry.pack(pady=10)

    generate_btn = tk.Button(root, text="Generate Constellation", font=("Helvetica",
12),
                             command=on_generate)
    generate_btn.pack(pady=10)

    note = tk.Label(root, text="Make sure images/<word>.svg exists.",
```

```
                    font=("Helvetica", 10), fg="gray")

    note.pack(pady=(0, 20))



    root.mainloop()




if __name__ == "__main__":

    main()
```