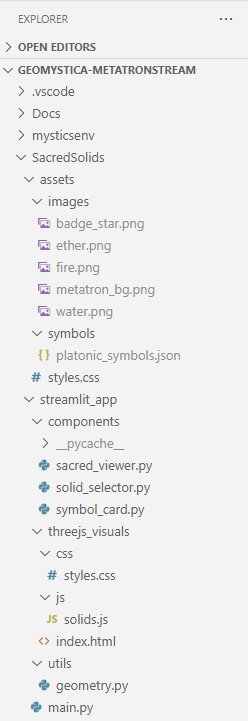
****

**requirements.txt**

streamlit>=1.30.0

numpy>=1.24.0

sympy>=1.12

pillow>=10.0.0

plotly>=5.20.0

watchdog>=3.0.0

streamlit-extras>=0.3.0

scipy

# streamlit-authenticator>=0.2.3      # For login/auth features

# pymongo>=4.6.0                      # If connecting to MongoDB

# streamlit-analytics>=0.1.0         # For tracking user interactions

# matplotlib>=3.8.0                   # For static plots

**Inside assets directory**

**styles.css**

body {

    font-family: 'Segoe UI', sans-serif;

    background-color: #f4f6f9;

    color: #333;

**}**

/\* Title styling \*/

h1 {

    font-size: 2.5em;

    color: #2c3e50;

    text-align: center;

    margin-bottom: 0.5em;

}

/\* Sidebar customization \*/

[data-testid="stSidebar"] {

    background-color: #ecf0f1;

    border-right: 2px solid #bdc3c7;

}

[data-testid="stSidebar"] .css-1d391kg {

    padding-top: 2em;

}

/\* Plotly chart container \*/

.plot-container {

    border-radius: 12px;

    box-shadow: 0 4px 12px rgba(0,0,0,0.1);

    padding: 1em;

    background-color: #ffffff;

}

/\* Latex and symbolic section \*/

.stMarkdown h2, .stMarkdown h3 {

    color: #34495e;

}

.stMarkdown .latex {

    font-size: 1.2em;

    color: #2c3e50;

}

/\* Footer \*/

footer {

    text-align: center;

    font-size: 0.9em;

    color: #7f8c8d;

    margin-top: 2em;

}

/\* Button hover effect \*/

button:hover {

    background-color: #3498db !important;

    color: white !important;

    transition: background-color 0.3s ease;

}

/\* Smooth fade-in animation \*/

@keyframes fadeIn {

    from { opacity: 0; transform: translateY(10px); }

    to { opacity: 1; transform: translateY(0); }

}

.stApp {

    animation: fadeIn 0.6s ease-in-out;

}

**Inside streamlit\_app/components Directory**

**sacred\_viewer.py**

import os

import streamlit.components.v1 as components

def show\_3d\_viewer(solid\_name=None):

    """

    Loads and embeds the Three.js viewer from index.html.

    Optionally injects a script to initialize a specific solid.

    """

    base\_dir = os.path.dirname(os.path.abspath(\_\_file\_\_))

    raw\_path = os.path.join(base\_dir, "..", "threejs\_visuals", "index.html")

    viewer\_path = os.path.normpath(raw\_path)

    if not os.path.exists(viewer\_path):

        raise FileNotFoundError(f"Viewer HTML not found at: {viewer\_path}")

    with open(viewer\_path, "r", encoding="utf-8") as f:

        html\_content = f.read()

    # Optional: Inject solid name into viewer via JS

    if solid\_name:

        injection = f"""

        <script>

            if (typeof createSolid === 'function') {{

                createSolid("{solid\_name.lower()}");

            }}

        </script>

        """

        html\_content += injection

    components.html(html\_content, height=600, scrolling=False)

**solid\_selector.py**

import numpy as np

from sympy import Eq

from sympy.abc import V, E, F

def get\_solid(name):

    """

    Returns vertices and faces for the given Platonic solid.

    """

    if name == "Tetrahedron":

        vertices = np.array([

            [1, 1, 1],

            [-1, -1, 1],

            [-1, 1, -1],

            [1, -1, -1]

        ])

        faces = [[0,1,2], [0,1,3], [0,2,3], [1,2,3]]

    elif name == "Cube":

        vertices = np.array([

            [-1, -1, -1], [1, -1, -1],

            [1, 1, -1], [-1, 1, -1],

            [-1, -1, 1], [1, -1, 1],

            [1, 1, 1], [-1, 1, 1]

        ])

        faces = [

            [0,1,2,3], [4,5,6,7],

            [0,1,5,4], [2,3,7,6],

            [1,2,6,5], [0,3,7,4]

        ]

    elif name == "Octahedron":

        vertices = np.array([

            [1,0,0], [-1,0,0],

            [0,1,0], [0,-1,0],

            [0,0,1], [0,0,-1]

        ])

        faces = [

            [0,2,4], [2,1,4], [1,3,4], [3,0,4],

            [0,2,5], [2,1,5], [1,3,5], [3,0,5]

        ]

    elif name == "Dodecahedron":

        from scipy.spatial import ConvexHull

        phi = (1 + np.sqrt(5)) / 2

        a, b = 1, 1 / phi

        points = []

        for i in [-a, a]:

            for j in [-a, a]:

                for k in [-a, a]:

                    points.append([i, j, k])

        for i in [-b, b]:

            for j in [-b, b]:

                points += [[0, i, j], [i, 0, j], [i, j, 0]]

        vertices = np.array(points)

        hull = ConvexHull(vertices)

        faces = hull.simplices.tolist()

    else:

        return None, None

    return vertices, faces

def symbolic\_description(name):

    """

    Returns Euler's formula and symbolic counts for the solid.

    """

    if name == "Tetrahedron":

        return Eq(V - E + F, 2), {"V": 4, "E": 6, "F": 4}

    elif name == "Cube":

        return Eq(V - E + F, 2), {"V": 8, "E": 12, "F": 6}

    elif name == "Octahedron":

        return Eq(V - E + F, 2), {"V": 6, "E": 12, "F": 8}

    elif name == "Dodecahedron":

        return Eq(V - E + F, 2), {"V": 20, "E": 30, "F": 12}

    else:

        return None, {}

def get\_symbolic\_overlay(name):

    """

    Placeholder for symbolic overlays (e.g. Metatron’s Cube, golden spirals).

    """

    overlays = {

        "Tetrahedron": "Fire, initiation, upward motion",

        "Cube": "Earth, stability, foundation",

        "Octahedron": "Air, balance, duality",

        "Dodecahedron": "Ether, cosmos, divine geometry"

    }

    return overlays.get(name, "No symbolic overlay defined.")

# Optional: UI selector (if needed)

def select\_solid():

    import streamlit as st

    return st.sidebar.selectbox("Choose a Platonic Solid", ["Tetrahedron", "Cube", "Octahedron", "Dodecahedron"])

**symbol\_card.py**

import streamlit as st

def show\_symbol\_card(solid\_name, symbol\_data):

    st.markdown(f"""

    <div style="text-align:center; padding:10px; border-radius:10px; background-color:{symbol\_data['color']}; color:white;">

        <h2>{symbol\_data['symbol']} {solid\_name.capitalize()}</h2>

        <h4>Element: {symbol\_data['element']}</h4>

        <p><em>{symbol\_data['meaning']}</em></p>

    </div>

    """, unsafe\_allow\_html=True)

**Inside streamlit\_app/threejs\_visuals Directory**

css subdirectory

styles.css

/\* SacredSolids: Three.js Canvas Styling \*/

body {

    margin: 0;

    padding: 0;

    background-color: #f0f4f8;

    font-family: 'Segoe UI', sans-serif;

    overflow: hidden;

}

/\* Canvas container \*/

#scene-container {

    width: 100vw;

    height: 100vh;

    display: flex;

    justify-content: center;

    align-items: center;

    background: radial-gradient(circle at center, #ffffff 0%, #dfe6e9 100%);

}

/\* Info overlay (optional) \*/

#info-box {

    position: absolute;

    top: 20px;

    left: 20px;

    background-color: rgba(44, 62, 80, 0.85);

    color: white;

    padding: 12px 16px;

    border-radius: 8px;

    font-size: 14px;

    max-width: 300px;

    box-shadow: 0 4px 12px rgba(0,0,0,0.2);

}

/\* Symbolic labels (optional) \*/

.label {

    position: absolute;

    color: #2c3e50;

    font-weight: bold;

    background-color: rgba(255,255,255,0.8);

    padding: 4px 8px;

    border-radius: 6px;

    font-size: 12px;

    pointer-events: none;

}

/\* Fade-in animation \*/

@keyframes fadeIn {

    from { opacity: 0; transform: scale(0.95); }

    to { opacity: 1; transform: scale(1); }

}

#scene-container, #info-box {

    animation: fadeIn 0.8s ease-out;

}

js subdirectory

**solids.js**

// js/solids.js

function createSolid(type) {

  switch (type) {

    case "tetrahedron": return new THREE.TetrahedronGeometry(1);

    case "cube": return new THREE.BoxGeometry(1, 1, 1);

    case "octahedron": return new THREE.OctahedronGeometry(1);

    case "dodecahedron": return new THREE.DodecahedronGeometry(1);

    case "icosahedron": return new THREE.IcosahedronGeometry(1);

    default: return new THREE.TetrahedronGeometry(1); // fallback

  }

}

**inside threejs\_visuals Directory**

index.html

<!DOCTYPE html>

<html lang="en">

<head>

    <meta name="viewport" content="width=device-width, initial-scale=1.0" />

    <meta charset="UTF-8" />

    <title>SacredSolids 3D Viewer</title>

    <!-- <link rel="stylesheet" href="./css/styles.css" /> -->

    <script src="https://cdnjs.cloudflare.com/ajax/libs/three.js/r148/three.min.js"></script>

    <script src="js/solids.js"></script>

    <!-- <style>

  body {

    margin: 0;

    padding: 0;

    background-color: #f0f4f8;

    font-family: 'Segoe UI', sans-serif;

    overflow: hidden;

  }

  #scene-container {

    width: 100vw;

    height: 100vh;

    display: flex;

    justify-content: center;

    align-items: center;

    background: radial-gradient(circle at center, #ffffff 0%, #dfe6e9 100%);

  }

</style> -->

</head>

<body>

    <div id="scene-container"></div>

    <div id="info-box">🔺 Tetrahedron: Symbol of Fire, Willpower, and Transformation</div>

    <script>

        // Scene setup

        const scene = new THREE.Scene();

        scene.background = new THREE.Color(0xf0f4f8);

        const camera = new THREE.PerspectiveCamera(45, window.innerWidth / window.innerHeight, 0.1, 1000);

        camera.position.set(3, 3, 3);

        const renderer = new THREE.WebGLRenderer({ antialias: true });

        renderer.setSize(window.innerWidth, window.innerHeight);

        document.getElementById("scene-container").appendChild(renderer.domElement);

        // Lighting

        const ambientLight = new THREE.AmbientLight(0xffffff, 0.6);

        scene.add(ambientLight);

        const directionalLight = new THREE.DirectionalLight(0xffffff, 0.8);

        directionalLight.position.set(5, 5, 5);

        scene.add(directionalLight);

        // Tetrahedron geometry

        const geometry = new THREE.TetrahedronGeometry(1);

        const material = new THREE.MeshStandardMaterial({

            color: 0xe74c3c,

            flatShading: true,

            transparent: true,

            opacity: 0.9

        });

        const tetrahedron = new THREE.Mesh(geometry, material);

        scene.add(tetrahedron);

        // Wireframe overlay

        const wireframe = new THREE.LineSegments(

            new THREE.EdgesGeometry(geometry),

            new THREE.LineBasicMaterial({ color: 0x2c3e50 })

        );

        tetrahedron.add(wireframe);

        // Animation loop

        function animate() {

            requestAnimationFrame(animate);

            tetrahedron.rotation.x += 0.005;

            tetrahedron.rotation.y += 0.005;

            renderer.render(scene, camera);

        }

        animate();

        // Responsive resize

        window.addEventListener("resize", () => {

            camera.aspect = window.innerWidth / window.innerHeight;

            camera.updateProjectionMatrix();

            renderer.setSize(window.innerWidth, window.innerHeight);

        });

    </script>

</body>

</html>

**Inside streamlit\_app/utils Directory**

geometry.py

import json

from pathlib import Path

SYMBOLS\_PATH = Path("assets/symbols/platonic\_symbols.json")

def load\_symbol\_data():

    with open(SYMBOLS\_PATH, "r") as f:

        return json.load(f)

def get\_solid\_info(solid\_name):

    symbols = load\_symbol\_data()

    return symbols.get(solid\_name, {

        "element": "Unknown",

        "symbol": "❓",

        "color": "#7f8c8d",

        "meaning": "No data available"

    })

SOLID\_MAP = {

    "tetrahedron": {"faces": 4, "element": "Fire"},

    "cube": {"faces": 6, "element": "Earth"},

    "octahedron": {"faces": 8, "element": "Air"},

    "dodecahedron": {"faces": 12, "element": "Ether"},

    "icosahedron": {"faces": 20, "element": "Water"},

}

def get\_solid\_info(name):

    return SOLID\_MAP.get(name, {})

Inside streamlit\_app Directory

main.py

import streamlit as st

import numpy as np

import plotly.graph\_objects as go

from sympy import symbols, Eq # noqa

from sympy.abc import V, E, F # noqa

from components.sacred\_viewer import show\_3d\_viewer

from components.solid\_selector import get\_solid, symbolic\_description

from components.solid\_selector import select\_solid # noqa

#print("Selected solid:", select\_solid("cube"))

# --- Page Setup ---

st.set\_page\_config(page\_title="SacredSolids", layout="wide")

st.title("🔷 SacredSolids: Platonic Geometry Explorer")

# --- Custom CSS ---

try:

    with open("assets/styles.css") as f:

        st.markdown(f"<style>{f.read()}</style>", unsafe\_allow\_html=True)

except FileNotFoundError:

    st.warning("Custom styles not found.")

# --- Optional UI Extras ---

try:

    from streamlit\_extras.add\_vertical\_space import add\_vertical\_space

except ImportError:

    def add\_vertical\_space(n): st.write("\n" \* n)

add\_vertical\_space(1)

# --- Solid Selection ---

solid\_name = st.sidebar.selectbox("Choose a Platonic Solid", ["Tetrahedron", "Cube", "Octahedron", "Dodecahedron"])

vertices, faces = get\_solid(solid\_name)

# --- 3D Viewer Embed ---

try:

    show\_3d\_viewer(solid\_name)

except FileNotFoundError as e:

    st.error(f"3D viewer not found: {e}")

# --- Plotly Visualization ---

def plot\_solid(vertices, faces, name):

    fig = go.Figure()

    for face in faces:

        face\_vertices = vertices[face]

        x, y, z = face\_vertices[:, 0], face\_vertices[:, 1], face\_vertices[:, 2]

        x = np.append(x, x[0])

        y = np.append(y, y[0])

        z = np.append(z, z[0])

        fig.add\_trace(go.Scatter3d(

            x=x, y=y, z=z,

            mode='lines',

            line=dict(color='royalblue', width=4),

            name="Face"

        ))

    fig.add\_trace(go.Scatter3d(

        x=vertices[:, 0], y=vertices[:, 1], z=vertices[:, 2],

        mode='markers',

        marker=dict(size=5, color='orange'),

        name='Vertices'

    ))

    fig.update\_layout(

        title=f"{name} Visualization",

        margin=dict(l=0, r=0, b=0, t=40),

        scene=dict(aspectmode='data')

    )

    return fig

# --- Symbolic Description ---

if vertices is not None:

    fig = plot\_solid(vertices, faces, solid\_name)

    st.plotly\_chart(fig, use\_container\_width=True)

    eq, values = symbolic\_description(solid\_name)

    st.subheader("🔣 Symbolic Description")

    st.latex(eq)

    st.write(f"Vertices (V): {values.get('V')}, Edges (E): {values.get('E')}, Faces (F): {values.get('F')}")

    st.markdown("---")

    st.caption("Built with ❤️ by Jagdev Singh Dosanjh")