```python
"""
Quantum Consciousness Equation Visualizations
Complete code for all diagrams in one file
Run this to generate all images at once
"""

import matplotlib.pyplot as plt
import numpy as np
from matplotlib.patches import FancyBboxPatch, FancyArrowPatch, Circle, Ellipse, Rectangle, Wedge
import matplotlib.patches as mpatches
from mpl_toolkits.mplot3d import Axes3D

print("Starting diagram generation...")
print("=" * 60)

# ============================================================================
# DIAGRAM 1: Classical vs Quantum Consciousness
# ============================================================================

def diagram_1():
    fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(16, 7))
    fig.suptitle('Classical vs Quantum Consciousness', fontsize=20, fontweight='bold', y=0.98)

    # LEFT: CLASSICAL
    ax1.set_xlim(0, 10)
    ax1.set_ylim(0, 10)
    ax1.axis('off')
    ax1.set_title('CLASSICAL CONSCIOUSNESS\n(After Measurement)', fontsize=16, fontweight='bold', pad=20)

    box1 = FancyBboxPatch((1, 7), 2, 1.5, boxstyle="round,pad=0.1",
                          edgecolor='black', facecolor='lightblue', linewidth=2)
    ax1.add_patch(box1)
    ax1.text(2, 7.75, 'Complexity\nx', ha='center', va='center', fontsize=12, fontweight='bold')

    arrow1 = FancyArrowPatch((2, 7), (2, 5.5), arrowstyle='->',
                             mutation_scale=30, linewidth=2, color='black')
    ax1.add_artist(arrow1)

    box2 = FancyBboxPatch((0.5, 4), 3, 1.5, boxstyle="round,pad=0.1",
                          edgecolor='darkblue', facecolor='lightgreen', linewidth=3)
    ax1.add_patch(box2)
    ax1.text(2, 4.75, 'C(x) = B(1 + ln x)', ha='center', va='center',
             fontsize=13, fontweight='bold', family='monospace')

    arrow2 = FancyArrowPatch((2, 4), (2, 2.5), arrowstyle='->',
                             mutation_scale=30, linewidth=2, color='black')
    ax1.add_artist(arrow2)

    box3 = FancyBboxPatch((0.5, 0.5), 3, 1.5, boxstyle="round,pad=0.1",
                          edgecolor='darkgreen', facecolor='gold', linewidth=3)
    ax1.add_patch(box3)
    ax1.text(2, 1.25, 'Single\nDefinite Value', ha='center', va='center',
             fontsize=13, fontweight='bold')

    ax1.plot(2, 1.25, marker='', markersize=30, color='red', zorder=10)
    ax1.text(5, 5, '→ ONE\n   REALITY', fontsize=14, fontweight='bold',
             bbox=dict(boxstyle='round', facecolor='wheat', alpha=0.8))

    # RIGHT: QUANTUM
    ax2.set_xlim(0, 10)
    ax2.set_ylim(0, 10)
    ax2.axis('off')
    ax2.set_title('QUANTUM CONSCIOUSNESS\n(Before Measurement)', fontsize=16, fontweight=
```

```
    'bold', pad=20)

    box4 = FancyBboxPatch((1, 7), 2, 1.5, boxstyle="round,pad=0.1",
                          edgecolor='black', facecolor='lightblue', linewidth=2)
    ax2.add_patch(box4)
    ax2.text(2, 7.75, 'Complexity\nX^', ha='center', va='center', fontsize=12, fontweight
='bold')

    arrow3 = FancyArrowPatch((2, 7), (2, 5.5), arrowstyle='->',
                             mutation_scale=30, linewidth=2, color='black')
    ax2.add_artist(arrow3)

    box5 = FancyBboxPatch((0.2, 4), 3.6, 1.5, boxstyle="round,pad=0.1",
                          edgecolor='darkblue', facecolor='lightgreen', linewidth=3)
    ax2.add_patch(box5)
    ax2.text(2, 4.75, '(C) = B(1 + ln X^)', ha='center', va='center',
             fontsize=13, fontweight='bold', family='monospace')

    arrow4a = FancyArrowPatch((2, 4), (1, 2.5), arrowstyle='->',
                              mutation_scale=30, linewidth=2, color='purple')
    ax2.add_artist(arrow4a)
    arrow4b = FancyArrowPatch((2, 4), (2, 2.5), arrowstyle='->',
                              mutation_scale=30, linewidth=2, color='purple')
    ax2.add_artist(arrow4b)
    arrow4c = FancyArrowPatch((2, 4), (3, 2.5), arrowstyle='->',
                              mutation_scale=30, linewidth=2, color='purple')
    ax2.add_artist(arrow4c)

    colors = ['#FF6B6B', '#4ECDC4', '#45B7D1']
    positions = [1, 2, 3]
    labels = ['C□\n30%', 'C□\n45%', 'C□\n25%']

    for pos, color, label in zip(positions, colors, labels):
        circle = plt.Circle((pos, 1.25), 0.4, color=color, alpha=0.7, linewidth=2,
edgecolor='black')
        ax2.add_patch(circle)
        ax2.text(pos, 1.25, label, ha='center', va='center', fontsize=10, fontweight=
'bold')

    cloud = Ellipse((2, 1.25), 3, 1.5, alpha=0.2, facecolor='purple', edgecolor='purple',
                    linewidth=2, linestyle='--')
    ax2.add_patch(cloud)

    ax2.text(5.5, 5, '→ MANY\n   POSSIBILITIES', fontsize=14, fontweight='bold',
             bbox=dict(boxstyle='round', facecolor='lavender', alpha=0.8))

    ax2.text(2, 0.2, '≈ Superposition ≈', ha='center', fontsize=11,
             style='italic', color='purple', fontweight='bold')

    plt.tight_layout()
    plt.savefig('1_consciousness_classical_vs_quantum.png', dpi=300, bbox_inches='tight')
    print("✓ Saved: 1_consciousness_classical_vs_quantum.png")
    plt.close()

# ============================================================================
# DIAGRAM 2: Measurement Collapse
# ============================================================================

def diagram_2():
    fig, (ax1, ax2, ax3) = plt.subplots(1, 3, figsize=(18, 6))
    fig.suptitle('The Measurement Collapse', fontsize=22, fontweight='bold', y=0.98)

    # LEFT: BEFORE
    ax1.set_xlim(0, 10)
    ax1.set_ylim(0, 10)
    ax1.axis('off')
    ax1.set_title('BEFORE OBSERVATION', fontsize=14, fontweight='bold', pad=15)
```

```
127
128         theta = np.linspace(0, 2*np.pi, 100)
129         r = 2 + 0.5*np.sin(5*theta)
130         x_cloud = 5 + r*np.cos(theta)
131         y_cloud = 5 + r*np.sin(theta)
132         ax1.fill(x_cloud, y_cloud, alpha=0.3, color='purple', edgecolor='purple', linewidth=3
    )
133
134         states = [(4, 6, 'C□'), (5, 5, 'C□'), (6, 6, 'C□'), (5, 4, 'C□')]
135         for x, y, label in states:
136             circle = Circle((x, y), 0.3, color='lightblue', alpha=0.6, edgecolor='blue',
    linewidth=2)
137             ax1.add_patch(circle)
138             ax1.text(x, y, label, ha='center', va='center', fontsize=10, fontweight='bold')
139
140         ax1.text(5, 5, '?', fontsize=60, ha='center', va='center',
141                  color='purple', alpha=0.4, fontweight='bold')
142
143         ax1.text(5, 2, '□C□\nSuperposition', ha='center', fontsize=13,
144                  fontweight='bold', bbox=dict(boxstyle='round', facecolor='lavender', alpha=
    0.8))
145
146         ax1.text(5, 8.5, '|ψ□ = Σ αᵢ|Cᵢ□', ha='center', fontsize=12,
147                  family='monospace', bbox=dict(boxstyle='round', facecolor='white', alpha=0.9
    ))
148
149         # MIDDLE: MEASUREMENT
150         ax2.set_xlim(0, 10)
151         ax2.set_ylim(0, 10)
152         ax2.axis('off')
153         ax2.set_title('MEASUREMENT', fontsize=14, fontweight='bold', pad=15)
154
155         eye_circle = Circle((5, 6), 1, color='gold', edgecolor='black', linewidth=3)
156         ax2.add_patch(eye_circle)
157         pupil = Circle((5, 6), 0.4, color='black')
158         ax2.add_patch(pupil)
159         highlight = Circle((5.2, 6.2), 0.15, color='white')
160         ax2.add_patch(highlight)
161
162         for angle in np.linspace(0, 2*np.pi, 8, endpoint=False):
163             x_end = 5 + 2*np.cos(angle)
164             y_end = 6 + 2*np.sin(angle)
165             ax2.plot([5, x_end], [6, y_end], 'gold', linewidth=3, alpha=0.7)
166
167         ax2.text(5, 3, 'OBSERVE', ha='center', fontsize=16, fontweight='bold',
168                  bbox=dict(boxstyle='round', facecolor='yellow', alpha=0.8))
169
170         for i in range(8):
171             angle = i * np.pi / 4
172             arrow = FancyArrowPatch((5 + 1.5*np.cos(angle), 6 + 1.5*np.sin(angle)),
173                                     (5 + 0.5*np.cos(angle), 6 + 0.5*np.sin(angle)),
174                                     arrowstyle='->', mutation_scale=20, linewidth=2,
175                                     color='red', alpha=0.6)
176             ax2.add_artist(arrow)
177
178         ax2.text(5, 1, 'Wavefunction\nCollapse', ha='center', fontsize=11,
179                  style='italic', color='red', fontweight='bold')
180
181         # RIGHT: AFTER
182         ax3.set_xlim(0, 10)
183         ax3.set_ylim(0, 10)
184         ax3.axis('off')
185         ax3.set_title('AFTER OBSERVATION', fontsize=14, fontweight='bold', pad=15)
186
187         star_x, star_y = 5, 5
188         ax3.plot(star_x, star_y, marker='*', markersize=80, color='gold',
189                  markeredgecolor='orange', markeredgewidth=3, zorder=10)
```

```
    for radius in [0.8, 1.2, 1.6]:
        circle = Circle((star_x, star_y), radius, color='yellow', alpha=0.2)
        ax3.add_patch(circle)

    ax3.text(5, 2.5, 'C = 3B\nDefinite Value', ha='center', fontsize=13,
             fontweight='bold', bbox=dict(boxstyle='round', facecolor='gold', alpha=0.8))

    ax3.text(5, 8.5, '|ψ⟩ → |C₂⟩', ha='center', fontsize=12,
             family='monospace', bbox=dict(boxstyle='round', facecolor='white', alpha=0.9
))

    ax3.text(5, 7, '✓', fontsize=50, ha='center', va='center',
             color='green', fontweight='bold')

    plt.tight_layout()
    plt.savefig('2_consciousness_measurement_collapse.png', dpi=300, bbox_inches='tight')
    print("✓ Saved: 2_consciousness_measurement_collapse.png")
    plt.close()

# ============================================================================
# DIAGRAM 3: Consciousness Growth Curve
# ============================================================================

def diagram_3():
    fig, ax = plt.subplots(figsize=(14, 9))

    B = 1
    x = np.linspace(0.1, 20, 1000)
    C = B * (1 + np.log(x))

    # Main curve
    ax.plot(x, C, linewidth=4, color='darkblue', label='C(x) = B(1 + ln x)', zorder=5)
    ax.fill_between(x, 0, C, alpha=0.2, color='lightblue')

    # Mark special points
    special_points = [
        (1, B * (1 + np.log(1)), 'Atom\nC = B', 'red'),
        (np.e, B * (1 + np.log(np.e)), 'Cell\nC = 2B', 'green'),
        (np.e**2, B * (1 + np.log(np.e**2)), 'Human\nC = 3B', 'blue'),
        (np.e**3, B * (1 + np.log(np.e**3)), 'Civilization\nC = 4B', 'purple')
    ]

    for x_pt, y_pt, label, color in special_points:
        ax.plot(x_pt, y_pt, 'o', markersize=18, color=color,
                markeredgecolor='black', markeredgewidth=2, zorder=10)
        ax.annotate(label, xy=(x_pt, y_pt), xytext=(x_pt, y_pt + 0.6),
                    fontsize=12, fontweight='bold', ha='center',
                    bbox=dict(boxstyle='round,pad=0.5', facecolor=color, alpha=0.8,
edgecolor='black', linewidth=2),
                    arrowprops=dict(arrowstyle='->', lw=2, color='black'))

    # Reference lines
    ax.axhline(y=B, color='gray', linestyle='--', alpha=0.5, linewidth=1.5, label='Base
Consciousness (B)')
    ax.axvline(x=1, color='gray', linestyle='--', alpha=0.5, linewidth=1.5)

    # Grid
    ax.grid(True, alpha=0.3, linestyle=':', linewidth=1)

    # Labels
    ax.set_xlabel('Complexity (x)', fontsize=14, fontweight='bold')
    ax.set_ylabel('Consciousness C(x)', fontsize=14, fontweight='bold')
    ax.set_title('Consciousness Growth: Logarithmic Scaling', fontsize=18, fontweight=
'bold', pad=20)

    # Add equation box
```

```
    equation_text = r'$C(x) = B(1 + \ln x)$' + '\n\nLogarithmic growth:\nInformation scales consciousness'
    ax.text(0.98, 0.05, equation_text, transform=ax.transAxes,
            fontsize=13, verticalalignment='bottom', horizontalalignment='right',
            bbox=dict(boxstyle='round', facecolor='wheat', alpha=0.9, edgecolor='black', linewidth=2),
            family='monospace', fontweight='bold')

    ax.legend(loc='upper left', fontsize=11, framealpha=0.9)
    ax.set_xlim(0, 20)
    ax.set_ylim(0, 5)

    plt.tight_layout()
    plt.savefig('3_consciousness_growth_curve.png', dpi=300, bbox_inches='tight')
    print("✓ Saved: 3_consciousness_growth_curve.png")
    plt.close()

# ============================================================================
# DIAGRAM 4: Superposition States
# ============================================================================

def diagram_4():
    fig, ax = plt.subplots(figsize=(14, 10))
    ax.set_xlim(0, 10)
    ax.set_ylim(0, 10)
    ax.axis('off')

    fig.suptitle('Consciousness Superposition', fontsize=22, fontweight='bold', y=0.96)

    # Title
    ax.text(5, 9.2, 'Quantum State: |ψ□ = α□|x□□ + α□|x□□ + α□|x□□',
            ha='center', fontsize=14, family='monospace',
            bbox=dict(boxstyle='round', facecolor='lightblue', alpha=0.8, edgecolor='black', linewidth=2))

    # Top: Input state
    input_box = FancyBboxPatch((3.5, 7.5), 3, 1, boxstyle="round,pad=0.15",
                               edgecolor='black', facecolor='lightyellow', linewidth=3)
    ax.add_patch(input_box)
    ax.text(5, 8, 'QUANTUM STATE |ψ□', ha='center', va='center',
            fontsize=13, fontweight='bold')

    # Arrows splitting - FIXED
    arrow_targets = [3, 5, 7]
    for x_target in arrow_targets:
        arrow = FancyArrowPatch((5, 7.5), (x_target, 6.5), arrowstyle='->',
                                mutation_scale=25, linewidth=3, color='purple', alpha=0.7)
        ax.add_artist(arrow)

    # Three states
    states_data = [
        (3, 5, '|x□□', 'C□ = 2B', '33%', '#FF6B6B'),
        (5, 5, '|x□□', 'C□ = 4B', '33%', '#4ECDC4'),
        (7, 5, '|x□□', 'C□ = 6B', '33%', '#45B7D1')
    ]

    for x_pos, y_pos, state_label, c_label, prob, color in states_data:
        # State circle
        circle = Circle((x_pos, y_pos), 0.6, color=color, alpha=0.7,
                        edgecolor='black', linewidth=3, zorder=5)
        ax.add_patch(circle)
        ax.text(x_pos, y_pos, state_label, ha='center', va='center',
                fontsize=12, fontweight='bold', family='monospace')

        # Consciousness value below
        ax.text(x_pos, y_pos - 1, c_label, ha='center', fontsize=11,
                fontweight='bold', bbox=dict(boxstyle='round', facecolor=color, alpha=0.6
```

```
    ))
317
318         # Probability
319         ax.text(x_pos, y_pos - 1.6, prob, ha='center', fontsize=10,
320                 style='italic', fontweight='bold')
321
322     # Arrows down to average - FIXED
323     for x_start in [3, 5, 7]:
324         arrow = FancyArrowPatch((x_start, 3.2), (5, 2), arrowstyle='->',
325                                 mutation_scale=25, linewidth=2.5, color='green', alpha=0.7
    )
326         ax.add_artist(arrow)
327
328     # Average consciousness
329     avg_box = FancyBboxPatch((3, 0.5), 4, 1.2, boxstyle="round,pad=0.15",
330                              edgecolor='darkgreen', facecolor='lightgreen', linewidth=4)
331     ax.add_patch(avg_box)
332     ax.text(5, 1.1, '□C□ = 4B', ha='center', va='center',
333             fontsize=16, fontweight='bold', family='monospace')
334     ax.text(5, 0.7, '(Average Consciousness)', ha='center', va='center',
335             fontsize=10, style='italic')
336
337     # Side annotation
338     ax.text(9, 5, 'Before\nMeasurement:\nAll states\nexist\nsimultaneously',
339             fontsize=11, ha='center', fontweight='bold',
340             bbox=dict(boxstyle='round', facecolor='lavender', alpha=0.8, edgecolor=
    'purple', linewidth=2))
341
342     plt.tight_layout()
343     plt.savefig('4_consciousness_superposition.png', dpi=300, bbox_inches='tight')
344     print("✓ Saved: 4_consciousness_superposition.png")
345     plt.close()
346
347 # ============================================================================
348 # DIAGRAM 5: Uncertainty Principle
349 # ============================================================================
350
351 def diagram_5():
352     fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(16, 8))
353     fig.suptitle('Consciousness-Complexity Uncertainty Principle', fontsize=20,
    fontweight='bold', y=0.96)
354
355     # LEFT: High certainty in C, low in X
356     ax1.set_xlim(0, 10)
357     ax1.set_ylim(0, 10)
358     ax1.axis('off')
359     ax1.set_title('High Certainty in Consciousness', fontsize=14, fontweight='bold', pad=
    15)
360
361     # Consciousness - narrow distribution
362     c_box = Rectangle((3.5, 6), 3, 2, facecolor='blue', alpha=0.7, edgecolor='darkblue',
    linewidth=3)
363     ax1.add_patch(c_box)
364     ax1.text(5, 7, 'C', ha='center', va='center', fontsize=20, fontweight='bold', color=
    'white')
365     ax1.text(5, 5.3, 'ΔC ≈ 0 (small)', ha='center', fontsize=11, fontweight='bold')
366     ax1.text(1, 7, 'Precise\nvalue', fontsize=10, fontweight='bold',
367              bbox=dict(boxstyle='round', facecolor='lightblue', alpha=0.8))
368
369     # Arrow down
370     arrow1 = FancyArrowPatch((5, 5), (5, 3.5), arrowstyle='<->',
371                              mutation_scale=30, linewidth=3, color='red')
372     ax1.add_artist(arrow1)
373     ax1.text(6, 4.2, 'Therefore...', fontsize=11, style='italic', color='red', fontweight
    ='bold')
374
375     # Complexity - wide distribution
```

```
    x_box = Rectangle((1, 1), 8, 1.5, facecolor='orange', alpha=0.5, edgecolor='darkorange', linewidth=3)
    ax1.add_patch(x_box)
    ax1.text(5, 1.75, 'X (Complexity)', ha='center', va='center', fontsize=16, fontweight='bold')
    ax1.text(5, 0.3, 'ΔX ≈ large (uncertain)', ha='center', fontsize=11, fontweight='bold', color='red')
    ax1.text(9.5, 1.75, 'Spread\nout', fontsize=10, fontweight='bold', rotation=0,
            bbox=dict(boxstyle='round', facecolor='lightyellow', alpha=0.8))

    # RIGHT: High certainty in X, low in C
    ax2.set_xlim(0, 10)
    ax2.set_ylim(0, 10)
    ax2.axis('off')
    ax2.set_title('High Certainty in Complexity', fontsize=14, fontweight='bold', pad=15)

    # Complexity - narrow distribution
    x_box2 = Rectangle((3.5, 6), 3, 2, facecolor='orange', alpha=0.7, edgecolor='darkorange', linewidth=3)
    ax2.add_patch(x_box2)
    ax2.text(5, 7, 'X', ha='center', va='center', fontsize=20, fontweight='bold', color='white')
    ax2.text(5, 5.3, 'ΔX ≈ 0 (small)', ha='center', fontsize=11, fontweight='bold')
    ax2.text(1, 7, 'Precise\nvalue', fontsize=10, fontweight='bold',
            bbox=dict(boxstyle='round', facecolor='lightyellow', alpha=0.8))

    # Arrow down
    arrow2 = FancyArrowPatch((5, 5), (5, 3.5), arrowstyle='<->',
                             mutation_scale=30, linewidth=3, color='red')
    ax2.add_artist(arrow2)
    ax2.text(6, 4.2, 'Therefore...', fontsize=11, style='italic', color='red', fontweight='bold')

    # Consciousness - wide distribution
    c_box2 = Rectangle((1, 1), 8, 1.5, facecolor='blue', alpha=0.5, edgecolor='darkblue', linewidth=3)
    ax2.add_patch(c_box2)
    ax2.text(5, 1.75, 'C (Consciousness)', ha='center', va='center', fontsize=16, fontweight='bold')
    ax2.text(5, 0.3, 'ΔC ≈ large (uncertain)', ha='center', fontsize=11, fontweight='bold', color='red')
    ax2.text(9.5, 1.75, 'Spread\nout', fontsize=10, fontweight='bold', rotation=0,
            bbox=dict(boxstyle='round', facecolor='lightblue', alpha=0.8))

    # Central equation
    fig.text(0.5, 0.08, 'ΔC · ΔX ≥ B/2', ha='center', fontsize=20, fontweight='bold',
            family='monospace', bbox=dict(boxstyle='round,pad=1', facecolor='yellow',
            alpha=0.9, edgecolor='black', linewidth=1))

    fig.text(0.5, 0.02, 'You cannot know both consciousness and complexity precisely at the same time',
            ha='center', fontsize=12, style='italic', fontweight='bold')

    plt.tight_layout(rect=[0, 0.12, 1, 0.95])
    plt.savefig('5_consciousness_uncertainty.png', dpi=300, bbox_inches='tight')
    print("✓ Saved: 5_consciousness_uncertainty.png")
    plt.close()

# ============================================================================
# RUN ALL DIAGRAMS
# ============================================================================

if __name__ == "__main__":
    diagram_1()
    diagram_2()
    diagram_3()
    diagram_4()
```

```
    diagram_5()

    print("=" * 60)
    print("✓ All 5 diagrams generated successfully!")
    print("Files created:")
    print("  - 1_classical-versus-quantum-consciousness.png")
    print("  - 2_measurement-collapse.png")
    print("  - 3_consciousness_growth_curve.png")
    print("  - 4_superposition-states.png")
    print("  - 5_uncertainty-principle.png")
```