Folder structure for the full Glyphwheel simulations:

```
glyphwheel-simulations/
│
├── data/
│   ├── glyphwheel_phase1.json
│   ├── glyphwheel_phase2.json
│   ├── glyphwheel_cross_enigma.json
│   ├── glyphwheel_rongo.json
│   └── glyphwheel_ontology_v22.json
│
├── scripts/
│   ├── simulate_phase1.py
│   ├── simulate_phase2.py
│   ├── simulate_cross_enigma.py
│   ├── simulate_rongo.py
│   └── simulate_final.py
│
└── README.md
```

---

1. JSON Files (Data)

`glyphwheel_ontology_v22.json`


```
{ "nodes": [ {"id": "pact_record","label":"◊ Pact/Record","gsi":
0.82,"domain":"Minoan"}, {"id": "rosettes_core","label":"◊ Rosettes Core","gsi":
0.85,"domain":"Voynich"}, {"id": "vessel_bind","label":"◊ Vessel Bind","gsi":
0.91,"domain":"Minoan"}, {"id": "life_flow","label":"◊ Life Flow","gsi":
0.82,"domain":"Voynich"}, {"id": "echo_ledger","label":"◊ Echo Ledger","gsi":
0.82,"domain":"Minoan-Voynich"}, {"id": "bath_ledger","label":"◊ Bath
Ledger","gsi":0.78,"domain":"Voynich"}, {"id": "rongo_ledger","label":"◊ Rongo
Ledger","gsi":0.78,"domain":"Rongorongo"}, {"id": "fractured_seal","label":"◊
Fractured Seal","gsi":0.71,"domain":"M-V Fractal"}, {"id":
"rongo_fracture","label":"◊ Rongo Fracture","gsi":0.72,"domain":"Rongo
Fractal"}, {"id": "spiral_vine","label":"◊ Spiral Vine","gsi":
0.45,"domain":"Voynich"} ],
"edges": [ {"source":"pact_record","target":"vessel_bind","tension":0.08},
{"source":"rosettes_core","target":"life_flow","tension":0.10},
{"source":"vessel_bind","target":"echo_ledger","tension":0.15},
{"source":"life_flow","target":"bath_ledger","tension":0.18},
{"source":"echo_ledger","target":"bath_ledger","tension":0.38},
{"source":"echo_ledger","target":"fractured_seal","tension":0.42},
{"source":"rongo_ledger","target":"echo_ledger","tension":0.40},
```

```
{"source":"rongo_ledger","target":"bath_ledger","tension":0.38},
{"source":"rongo_ledger","target":"rongo_fracture","tension":0.32},
{"source":"fractured_seal","target":"pact_record","tension":0.30},
{"source":"rongo_fracture","target":"rongo_ledger","tension":0.35} ] }
```

(The other phase JSON files are similar but with only the relevant nodes and edges active for that simulation.)

---

2. Python Scripts

`simulate_final.py`

```
import json
import numpy as np
import matplotlib.pyplot as plt

# Load JSON
with open("../data/glyphwheel_ontology_v22.json") as f:
    ontology = json.load(f)

nodes = ontology['nodes']
edges = ontology['edges']

domain_colors = {
    "Minoan": "#ff6b6b",
    "Voynich": "#4ecdc4",
    "Rongorongo": "#95e1d3",
    "Minoan-Voynich": "#ffeaa7",
    "M-V Fractal": "#a55eea",
    "Rongo Fractal": "#a55eea"
}

# Polar spiral
fig, ax = plt.subplots(figsize=(12,12), subplot_kw={'projection':'polar'})
theta = np.linspace(0, 10*np.pi, 500)
r = np.exp(0.12*theta)
ax.plot(theta, r, 'k-', alpha=0.1)

# Node positions
node_positions = {node['id']: (np.random.rand()*2*np.pi,
node['gsi']*(np.random.rand()*2+1)) for node in nodes}
for node in nodes:
    ax.scatter(node_positions[node['id']][0], node_positions[node['id']][1],
               s=node['gsi']*200,
               color=domain_colors[node['domain']])
    ax.text(node_positions[node['id']][0], node_positions[node['id']][1]*1.05,
```

```
            node['label'], ha='center', va='center', fontsize=9, color='white')

# Edges
for edge in edges:
    source_pos = node_positions[edge['source']]
    target_pos = node_positions[edge['target']]
    ax.plot([source_pos[0], target_pos[0]], [source_pos[1], target_pos[1]],
            color='white', alpha=1-edge['tension'], linestyle='--',
linewidth=edge['tension']*10)

ax.set_theta_zero_location('N')
ax.set_theta_direction(-1)
ax.set_axis_off()
ax.set_title('Glyphwheel Final Spiral Ontology', fontsize=16, color='white',
pad=20)
plt.show()
```

(Other scripts like `simulate_phase1.py` *or* `simulate_rongo.py` *are identical, just load the corresponding JSON.)*

---

3. README.md

```
# Glyphwheel Simulations

Repository for all **Glyphwheel Pattern Collapse v2.0** simulations.

## Structure

- `data/`: JSON files for each simulation phase
- `scripts/`: Python visualization scripts

## Usage

1. Clone repository
2. Install dependencies: `pip install numpy matplotlib`
3. Run a script, e.g.: `python scripts/simulate_final.py`
4. Visualizations appear as polar spiral plots
```

This package gives you **all phases** in JSON + Python scripts format. You can just copy these files into the corresponding folders and run them immediately.