Meta-Experimental Infrastructure Implementation Guide

Overview

This document defines the practical implementation of a meta-experimental ecosystem built on Blender
It operationalizes the RSVP–TARTAN–CLIO theoretical stack through reproducible, headless experimenta

System Directory Structure

```
rsvp_meta/
■■■ experiments/
■   ■■■ Tier_I/ ... Tier_IV/
■■■ bpy_scripts/
■   ■■■ generate_experiment.py
■   ■■■ simulate_entropy_field.py
■   ■■■ render_snapshot.py
■■■ meta_ops/
■   ■■■ meta_operators.py
■   ■■■ meta_ops.py
■   ■■■ config.json
■■■ automation/
■   ■■■ run_all.sh
■   ■■■ run_meta.sh
■   ■■■ schedule.cron
■   ■■■ environment_setup.sh
■■■ logs/
```

Blender Python Templates

Three primary bpy templates define experiment generation:
1. generate_experiment.py – creates scalar/vector field datasets.
2. simulate_entropy_field.py – evolves temporal experiments.
3. render_snapshot.py – exports static renders or OBJ geometry.
Each script runs headlessly via `blender -b -P script.py -- args`.

Meta-Operators and Python Orchestration

The meta_operators.py module implements 13 analytic, morphic, and recursive meta-operators.
meta_ops.py serves as the CLI interface, and config.json stores paths and default operators.
Each operator takes experiment directories as input and outputs JSON summaries to /meta/.

Shell Automation Templates

Automation scripts control headless generation and analysis:
- run_all.sh → generates experiments and invokes meta-operators.
- run_meta.sh → performs nightly analytics.
- environment_setup.sh → installs dependencies and prepares runtime.
A cron job schedules regular meta-analysis at 03:00 daily.

Categorization Matrix

Layer	File Type	Function	Extension
Experiment	JSON	Scalar/vector logs	.json
Geometry	OBJ/PLY	Mesh data	.obj
Render	PNG	Visualization frames	.png

Meta-Operators	JSON	Aggregated summaries	.json
Automation	SH	Orchestration scripts	.sh
Configuration	JSON	Path/registry data	.json

Execution Order

1. ./automation/environment_setup.sh
2. ./automation/run_all.sh
3. python meta_ops/meta_ops.py omega_composer experiments/Tier_III/*
4. Review logs/meta_results/<date>/meta_pipeline_summary.json

Development Guidelines

- Headless-first execution for all processes.
- JSON-based interprocess communication.
- Reproducibility: no state persistence outside logs.
- Tiered refinement from data → fusion → analysis → synthesis.