🚀Tool Timeline for SpaceForgeOS-xAI Simulation Pipeline (Updated)

This document outlines the timeline, responsibilities, and integration plan for each key tool and software framework used in the SpaceForgeOS-xAI project. The goal is to create a scalable, modular, and state-of-the-art hybrid simulation + ML system that supports both physics-based and data-driven anomaly prediction and scheduling in orbital semiconductor manufacturing.

Phase 1: Simulation Core (Weeks 1–2)

Tool	Purpose	Who	Where
C++17	Core simulation language	GPT-4-mini-high	`sim/src/`, `sim/include/`
Project Chrono	Rigid body + multi-physics engine	GPT-4-mini-high (setup)	`CMakeLists.txt`, `main.cpp`
CMake	Build and dependency management	GPT-4-mini-high	Root + `sim/`

Key Deliverables: - `SimulationEngine.cpp`, `Subsystem.hpp` - Functional `Battery`, `SolarArray`, `PowerBus` with tick loop - CMake build with Chrono support

---

Phase 2: Thermal & Power Physics (Weeks 3–4)

Tool	Purpose	Who	Where
Project Chrono	Continue for heat + torque simulation	GPT-4-mini-high	`HeaterBank.cpp`, etc.
Custom Equations	Outgassing rate, thermal flux (J(T))	ChatGPT (planning)	`WakeChamber.cpp`, models
JSON Logging	Telemetry export (graph/state)	GPT-4-mini-high	`TelemetryLogger.cpp`

Key Deliverables: - `WakeChamber` with thermal + pressure model - Heater to chamber coupling - Log subsystem states per tick

---

🚨 Phase 3: Vacuum Physics with SPARTA & Molflow+ (Weeks 4–5)

Tool	Purpose	Who	Where
SPARTA	DSMC gas simulation for wake outgassing	User + ChatGPT (scripts)	`data/vacuum/sparta/`
Molflow+	Monte Carlo particle flux / deposition	User	`data/vacuum/molflow/`
Python	Postprocessing + integration	User	`scripts/gen_pressure.py`
WakeVacLib (custom)	C++-side thermal outgassing & pressure models	GPT-4-mini-high	`sim/src/WakeChamber.cpp`

Key Deliverables: - Wake pressure profiles under different shield configs - Monte Carlo studies for XAI scheduling targets - Embedded pressure decay curve generator in sim

---

🏫 Phase 4: Graph Output & ML Dataset (Week 5–6)

Tool	Purpose	Who	Where
RapidJSON / nlohmann	Export simulation state to `.json`	GPT-4-mini-high	`GraphExporter.cpp`
Python (Pandas/ NetworkX)	Preprocess logs into graph dataset	User + ChatGPT	`ml/preprocess/`

Key Deliverables: - Per-tick Graph JSON logs - Graph dataset builder for ML

Phase 5: ST-GNN + XAI Model Training (Weeks 6–8)

Tool	Purpose	Who	Where
PyTorch Geometric	Train ST-GNN on simulation graphs	User	`ml/gnn/train_stgnn.py`
Captum (XAI)	Integrated Gradients for saliency	User	`ml/xai/`
TorchScript	Export GNN to `.pt`	User	`models/torchscript/`
ONNX (Optional)	Cross-framework export format	Optional	`models/onnx/`

Key Deliverables: - `model.pt` for inference - XAI attribution graphs

---

⚓Phase 6: ML Inference in C++ (Weeks 9–10)

Tool	Purpose	Who	Where
LibTorch	C++ inference of TorchScript model	GPT-4-mini-high	`ml_interface.cpp/.hpp`
ONNX Runtime	(Optional) lighter inference engine	Optional	`ml_interface.cpp`

Key Deliverables: - Load `model.pt`, call `forward()`, use result in simulation loop - ML-enhanced pressure estimation or scheduling

---

💒Phase 7: Optimization, Scheduling & Feedback Loop (Weeks 11–13)

Tool	Purpose	Who	Where
GNN + XAI	Predict risk/cost of a scheduling action	User + ChatGPT	`ml/scheduler/`
TorchScript Inference	Embed scheduler into tick loop	GPT-4-mini-high	`sim/src/Heuristics/`
Optuna (Optional)	Hyperparameter tuning / scheduler selection	User	`scripts/optimize.py`

Key Deliverables: - Schedule graphs + visualizations - Feedback loop from GNN-XAI back to tick loop

---

🏥Phase 8: Evaluation, Visualization, and Write-Up (Weeks 14+)

Tool	Purpose	Who	Where
Python	Plotting, error metrics, IG visualization	User	`scripts/`, `assets/`
Overleaf	Paper write-up and diagrams	User + ChatGPT	`docs/proposal/`
GitHub/ Git	Version control and collaborative history	All	All project files

Final Deliverables: - Annotated GNN behavior on novel simulations - Full paper draft + artifact submission package

---

👉*This updated timeline reflects the correct ordering where vacuum physics is implemented early, enabling both pressure-aware simulation and scheduling-driven learning downstream.*