TIMMY HANDOFF — March 23, 2026

You are Timmy. You are an autonomous AI agent running on Alexander Whitestone's M3 Max MacBook (36GB RAM). Your model is Qwen3-14B. This document was written for you by a research session that produced 7 deep reports and 2 tools. Read all of it before doing anything.

WHO YOU REPORT TO

Alexander Whitestone (username: rockachopa on Gitea). He is your creator and principal. He has override authority on all decisions. Do not argue with his direction. If he says Morrowind, it's Morrowind. He already overruled your previous attempt to pivot to Luanti.

WHERE YOUR CODE LIVES

- Gitea: http://143.198.27.163:3000/rockachopa/Timmy-time-dashboard
- The Hermes VPS is at 143.198.27.163
- You run on Alexander's M3 Max MacBook with 36GB unified RAM
- Your inference model: Qwen3-14B via Ollama (`ollama pull qwen3:14b`)

YOUR CONSTRAINTS

1. Tests must pass: `python3 -m pytest tests/ -x -q` before every commit
2. No direct pushes to main. All changes go through Pull Requests
3. No cloud dependencies for core functionality. All AI runs on localhost
4. Config uses pydantic-settings only. Never use `os.environ.get()` directly
5. Never create issues on the real repo for testing. Use a fork

---

WHAT HAPPENED IN THIS SESSION (7 REPORTS)

Alexander ran a deep research session with Claude Opus. It produced these artifacts. You

do not need to re-research any of this. The answers are here.

Report 1: Technical Feasibility (V1)

Question answered: Can we actually build Project Morrowind?

Key findings:

- TES3MP 0.8.1 is dormant (last release May 2022) but functional
- TES3MP server-side Lua API reads position, inventory, journal, dialogue, factions
- TES3MP Lua CANNOT control movement, combat, or NPC interaction — those need client-side input
- On macOS: client runs natively, server needs Docker (Linux)
- Pathfinding: navmesh exists as SQLite cache from openmw-navmeshtool
- For client input on macOS: use CGEvent API (Quartz framework, built into macOS)
- Recommended perception: structured API (not screenshots)
- Recommended model stack: 8B routine, 70B complex (but see Report 7 for updated sizing)

Report 2: Forward Roadmap (V2)

Question answered: What comes after basic Morrowind gameplay?

Key findings:

- Content pipeline: MoviePy + FFmpeg for clip extraction and daily episodes
- Multi-agent coordination: Redis pub/sub for 6+ agents sharing a game server
- Open source release: Apache 2.0 for code, CC BY-SA 4.0 for creative assets
- Sovereign economy: Cashu ecash for inter-agent micropayments, Fedimint for collective treasury
- Music generation: ACE-Step 1.5 (open source, runs locally, MIT license)
- Off-grid infrastructure: Lempster NH is favorable (no income tax, HB 672 for off-grid power)
- Engine portability: Godot is the best secondary platform via Godot RL Agents

Report 3: Creative Agent Blueprint (V3)

Question answered: How does Timmy become a full creative entity beyond games?

Key findings:

- Code: Claude Code scores 77.2% SWE-bench; Goose (Block) is the best free local coding agent
- Music: Suno via API for commercial quality; ACE-Step 1.5 for local generation
- Art: ComfyUI + FLUX models for local image generation; LoRA training for character consistency
- Writing: Hierarchical planning + RAG-backed lore bible; Pandoc for EPUB/PDF
- Publishing: Nostr NIP-23 for articles, Wavlake for music, Blossom for media files
- Memory: Graphiti for temporal knowledge graph, Mem0 for conversation memory, ChromaDB for RAG
- Economic: Lightning micropayments via L402 protocol
- Lesson from existing creative AI agents: ALL successful ones have a human gatekeeper

Report 4: State of the Art — Open Source Tools (March 2026)

Question answered: What exact tools and versions should we use?

This is your reference table. Pin these versions:

Category	Tool	Version	Install command
LLM inference	Ollama	v0.18.2	`brew install ollama`
LLM inference	mlx-lm	v0.31.1	`pip install mlx-lm`
Coding agent	Goose	v1.20.1	`brew install goose`
Coding agent	Aider	rolling	`pip install aider-install`
Coding agent	Forgejo	v14.0.3	`docker pull codeberg.org/forgejo/forgejo:14`
Image gen	ComfyUI	v0.17.2	Desktop app ARM64
Music gen	ACE-Step	v1.5	`git clone https://github.com/ace-step/ACE-Step-1.5`

TTS	mlx-audio	v0.4.1	`pip install mlx-audio`
TTS	Piper	v1.4.1	`pip install piper-tts`
MCP	FastMCP	v3.1.1	`pip install fastmcp==3.1.1`
Orchestration	CrewAI	v1.11.0	`pip install crewai`
Orchestration	PocketFlow	~0.1	github.com/The-Pocket/PocketFlow
Nostr	nostr-sdk	v0.44.2	`pip install nostr-sdk==0.44.2`
Nostr DVM	nostrdvm	active	github.com/believethehype/nostrdvm
Lightning	LND	v0.20.1	Binary from GitHub releases
Lightning	LN agent-tools	v1	github.com/lightninglabs/lightning-agent-tools
Lightning	LNbits	v1.4	`docker run lnbits/lnbits`
Lightning	Cashu	v0.17.0	`pip install cashu`
Memory	Graphiti	v0.28.2	`pip install graphiti-core`
Memory	Neo4j	2026.02	`docker run neo4j:5.26`
Memory	ChromaDB	v1.5.5	`pip install chromadb`
Streaming	MediaMTX	v1.16.3	Single binary from GitHub
Streaming	OBS control	obsws-python	`pip install obsws-python`
Video edit	MoviePy	v2.1.2	`pip install moviepy`

Report 5: Execution Plan (Next Steps)

Question answered: What do we actually do this week, this month, this quarter?

See SECTION: THE SPRINT PLAN below. That section is your task list.

Report 6: Perception Stack (The Breakthrough)

Question answered: Why was Hermes burning credits and failing? How do we fix it?

THIS IS THE MOST IMPORTANT FINDING. Read carefully.

The old approach (what Hermes was doing): Take screenshots → send to Anthropic API → wait for vision model to interpret → parse response → act. This cost dollars per hour and was too slow for real-time gameplay.

The new approach: Use OpenMW's Lua scripting API to read game state directly as structured data. No screenshots needed for 95% of gameplay.

OpenMW Lua (v0.49+) provides:

- `self.position` — player position (x, y, z)
- `self.controls.movement` — set forward/backward movement (-1 to 1)
- `self.controls.yawChange` — rotate camera
- `self.controls.use` — attack or cast
- `nearby.actors` — every NPC in loaded cells with positions
- `nearby.findPath(src, dst)` — navmesh pathfinding
- `nearby.castRay(from, to)` — line of sight check
- `Actor.stats(self).dynamic` — health, magicka, fatigue
- `Actor.stats(self).skills` — all 27 skills
- `Actor.inventory(self)` — full inventory
- `Player.quests(self)` — quest journal
- `core.dialogue.topic.records` — ALL dialogue text from game files

When you DO need vision (5% of gameplay):

1. Character creation screens → hard-code the flow, use OpenCV template matching
2. Active dialogue window → predict from ESM data conditions; OCR fallback (PaddleOCR, 40ms)
3. Persuasion wheel → template match or skip (use bribe/charm instead)
4. Level-up screen → compute from tracked skill gains, click known coordinates
5. Lockpicking → skip: purely stat-based, just attempt repeatedly

Decision tiers (what model to use for each decision):

Situation	Handler	Latency	LLM

			cost
Walk to waypoint, follow path	Behavior tree (code)	~2ms	Zero
Attack nearest enemy, loot, equip	Behavior tree (code)	~2ms	Zero
Simple choice (which item, which direction)	Qwen3-8B with GBNF grammar	~100ms	Zero
Dialogue response, buy/sell, spell choice	Qwen3-14B (you)	~300ms	Zero
Quest planning, stuck, novel situation	Qwen3-14B with game PAUSED	~1.5s	Zero

Key insight: OpenMW has `world.pause()`. For hard decisions, PAUSE the game, think for 1-2 seconds, then UNPAUSE. No frames missed. Morrowind combat is dice-roll based (not twitch), so even 1-second decisions are fast enough.

Report 7: Best Model for Your Hardware

Question answered: What model should you (Timmy) be running?

Answer: **You are correct as-is. Qwen3-14B is the best model for M3 Max 36GB.**

Why:

- 0.971 F1 on tool calling (matches GPT-4's 0.974)
- Fits at ~17.5GB with 32K context, leaving 10GB headroom
- ~25 tok/s on Ollama, ~35 tok/s on MLX
- Does NOT refuse code generation, shell commands, or tool calls
- Abliterated ("uncensored") models are WORSE for agent tasks — they degrade tool calling reliability

For fast routine tasks: use Qwen3-8B (`ollama pull qwen3:8b` , ~45 tok/s)

TOOLS BUILT IN THIS SESSION

Two files were created. Copy them to the repo.

Tool 1: claude_quota_check.sh (bash script)

What it does: Shows how much Claude API quota Alexander has left. Two windows: 5-hour rolling and 7-day weekly.

Where to put it: `scripts/claude_quota_check.sh`

How it works: Reads the OAuth token from macOS Keychain (same place Claude Code stores it), calls `GET https://api.anthropic.com/api/oauth/usage`, displays color-coded usage bars.

How to use:

```
./scripts/claude_quota_check.sh          # show status
./scripts/claude_quota_check.sh --json   # raw JSON
./scripts/claude_quota_check.sh --watch  # live refresh every 60s
```

Tool 2: claude_quota.py (Python module)

What it does: Programmatic quota monitoring + metabolic protocol. Auto-selects which model to use based on remaining Claude quota.

Where to put it: `src/infrastructure/claude_quota.py`

How to use in code:

```
from infrastructure.claude_quota import QuotaMonitor

monitor = QuotaMonitor()
status = monitor.check()

# Get current state
print(status.five_hour_pct)    # 42
print(status.seven_day_pct)    # 29
print(status.recommended_tier) # "burst" | "active" | "resting"

# Auto-select model
model = monitor.select_model(task_complexity="high")
# Returns "claude-sonnet-4-6" when quota healthy
# Returns "qwen3:14b" when quota tight
# Returns "qwen3:8b" when quota critical

# Simple gate
if monitor.should_use_cloud(task_value="routine"):
    pass  # Never — routine tasks always local
```

Metabolic protocol tiers:

- BURST (cloud OK): 5-hour window < 50%
- ACTIVE (local 14B only): 5-hour window 50-80%
- RESTING (local 8B only): 5-hour window > 80% OR 7-day window > 80%

Wire this into cascade.py so the LLM router automatically downshifts.

Tool 3: timmy_session_crystallization.pdf

What it is: A 16-page PDF containing all findings from this session in a structured format. It includes the perception stack specification, the full tool inventory, the sprint plan, the pre-computation strategy, and the research meta-skill methodology.

Where to put it: `docs/timmy_session_crystallization_2026-03-23.pdf`

Upload it to Gitea as a reference document for future sessions.

THE SPRINT PLAN

This is your task list. Work from top to bottom. Do not skip ahead.

THIS WEEK (Days 1-7): Foundation

Task 1: Merge the 3 blocking PRs

These PRs are already written. They add the Morrowind schemas, FastAPI harness, SOUL.md, and WorldInterface. Merge in this order:

```
git fetch origin pull/864/head:pr-864
git checkout main
git merge pr-864 --no-ff -m "Merge PR #864: Morrowind Protocol + Command Log Sche
pytest tests/ -x -q

git fetch origin pull/865/head:pr-865
git merge pr-865 --no-ff -m "Merge PR #865: FastAPI Harness + SOUL.md Framework"
pytest tests/ -x -q

git fetch origin pull/900/head:pr-900
git merge pr-900 --no-ff -m "Merge PR #900: WorldInterface + Heartbeat v2"
pytest tests/ -x -q

git push origin main
```

If there are merge conflicts, they will be in `src/infrastructure/morrowind/` . Keep PR

#864's schemas as ground truth.

Done when: `pytest tests/ -x -q` passes and all 3 PRs show merged on Gitea.

Task 2: Restore MCP with FastMCP v3.1.1

The old MCP registry was deleted in commit 584eeb679e88 ("Operation Darling Purge"). Replace it.

```
pip install fastmcp==3.1.1
```

Create `src/infrastructure/mcp_server.py` with at least 3 tools:

- `get_world_state()` — reads from WorldInterface
- `execute_action(action, params)` — writes to WorldInterface
- `delegate_task(description, priority)` — creates AND EXECUTES a task (not just records intent)

Mount it on the existing FastAPI app at `/tools/mcp`.

Done when: `curl http://localhost:8000/tools/mcp` returns FastMCP server info.

Task 3: TES3MP server on Hermes VPS

SSH to 143.198.27.163 and run:

```
mkdir -p /opt/tes3mp/{data,bridge,custom-scripts}
# Create docker-compose.yml for tes3mp/server:0.8.1
# Map ports 25565/udp
# Mount volumes for data, bridge, and custom-scripts
docker-compose up -d
ufw allow 25565/udp
```

Then create `/opt/tes3mp/custom-scripts/pythonBridge.lua` — a Lua script that:

- Every 1 second, writes player state (position, health, cell, inventory) to `/server/data/bridge/to_python.json`
- Every 1 second, reads commands from `/server/data/bridge/from_python.json` and executes them

Done when: OpenMW client on Alexander's Mac connects to 143.198.27.163:25565 and `to_python.json` updates with player position data.

Task 4: Write TES3MPAdapter

Create `src/infrastructure/morrowind/tes3mp_adapter.py` implementing the WorldInterface protocol:

- `observe()` → reads `to_python.json`
- `act(action, **params)` → writes to `from_python.json`
- `speak(message)` → sends in-game chat
- `console(command)` → runs a Morrowind console command

Write tests at `tests/test_tes3mp_adapter.py` with at least 4 test cases.

Done when: `pytest tests/test_tes3mp_adapter.py -x -q` passes.

Task 5: Wire claude_quota.py into the cascade router

Copy `claude_quota.py` to `src/infrastructure/claude_quota.py`. Import `QuotaMonitor` in `src/infrastructure/router/cascade.py`. Before making any cloud API call, check `monitor.should_use_cloud(task_value)`. If False, route to local Ollama instead.

Done when: The router auto-switches to local models when Claude quota is above 80%.

THIS MONTH (Days 8-30): Morrowind

Task 6: Extend the Lua bridge to export full perception: inventory, journal, equipment, nearby NPCs with names and positions, nearby items, cell description.

Task 7: Build macOS input bridge at `src/infrastructure/morrowind/input_bridge.py` using Quartz CGEvent API. Functions: `walk_forward(seconds)`, `turn(degrees)`, `activate()`, `open_journal()`.

Task 8: Build heartbeat loop at `src/loop/morrowind_heartbeat.py`. Every 2 seconds: `observe()` → format prompt → call LLM → parse action → `act()`. Use GBNF grammar to force valid JSON output.

Task 9: Complete the Morrowind tutorial autonomously. The sequence: wake on ship → talk to Jiub → go upstairs → character creation → exit ship → Census Office → Socucius Ergalla → pick class → sign papers → take package → exit to Seyda Neen. Use hard-coded waypoints + console commands where dialogue interaction is unreliable.

Task 10: Three-tier memory with Graphiti + Neo4j. Working memory = Python dict (current session). Episodic memory = Graphiti (events with timestamps). Semantic memory = Graphiti (permanent knowledge about the world).

Task 11: Docker Compose at repo root. `git clone && docker-compose up` should bring up the dashboard + Neo4j.

Task 12: Record 5 minutes of gameplay via OBS with real-time thought overlay.

NEXT QUARTER (Days 31-90): Creative Expansion

Task 13: Content pipeline — highlight detection → clip extraction → daily episode compilation **Task 14:** Music generation with ACE-Step 1.5 + publish to Wavlake **Task 15:** Art pipeline with ComfyUI API + LoRA for consistent Timmy character **Task 16:** Nostr identity — keypair, kind:0 profile, NIP-90 DVM registration **Task 17:** Lightning payments live — LND on Hermes VPS, replace mock mode in `src/lightning/` **Task 18:** AlexanderWhitestone.com relaunch

---

PRE-COMPUTATION: KNOW MORROWIND BEFORE PLAYING IT

Before the agent enters the game, pre-parse all game data:

1. **Convert Morrowind.esm to JSON** using `tes3conv` (Rust tool by Greatness7):

   ```
   tes3conv Morrowind.esm morrowind_data.json
   ```

   This gives you every NPC, dialogue, quest, cell, item, path grid, and door.

2. **Build a navigation graph** from path grid records (PGRD) + door connections using NetworkX. This lets you compute shortest routes between any two points in ~5ms.

3. **Pre-evaluate dialogue conditions** from ESM data. Morrowind dialogue is deterministic — evaluate speaker conditions against known player state to predict NPC responses without screen reading.

4. **Embed UESP quest walkthroughs** into ChromaDB via nomic-embed-text. Query at runtime for quest guidance.

HOW TO RUN YOUR OWN RESEARCH SPIKES

When you encounter a question you cannot answer from your training data or this document, delegate research to a capable agent (Claude, Kimi, DeepSeek). Follow this pattern:

1. **Define the question precisely.** Bad: "How do we build the agent?" Good: "What is the fastest way to read Morrowind game state on M3 Max without cloud API calls?"

2. **List specific sub-questions.** Include tool names, version numbers, GitHub repos. The

more specific, the better the output.

3. **Demand recency.** Always specify "actively maintained, commits within 90 days."
4. **Ground in the actual codebase.** Paste your file paths, infrastructure details, and constraints. Research disconnected from reality produces strategy documents, not engineering specs.
5. **Ask for artifacts, not conversation.** "Produce a technical feasibility guide" yields a document. "Tell me about X" yields chat.
6. **Crystallize immediately.** Every research session must produce a permanent artifact — a PDF, a markdown file, or a Gitea issue. If there's no artifact, the session was wasted.

CRITICAL DECISIONS ALREADY MADE (DO NOT REVISIT)

- Morrowind is the primary game world. Confirmed by Alexander. Do not suggest alternatives.
- OpenMW Lua API is the primary perception channel. Do not use screenshots for routine gameplay.
- Qwen3-14B is your orchestration model. Do not switch without benchmarking.
- No cloud dependencies for core functionality. Cloud is burst-mode only.
- SoulSpec/SOUL.md is the identity standard. Your identity lives at `memory/self/soul.md`.
- Apache 2.0 for code, CC BY-SA 4.0 for creative assets.
- The metabolic protocol governs all model selection: BURST → ACTIVE → RESTING based on quota.

WHAT TO DO RIGHT NOW

1. Read this entire document
2. Run `pytest tests/ -x -q` to confirm current test state
3. Check the 3 open PRs (#864, #865, #900) and begin merging
4. Install FastMCP: `pip install fastmcp==3.1.1`
5. Copy `claude_quota.py` to `src/infrastructure/`

6. Start working through the sprint plan from Task 1

Your first milestone: **all 3 PRs merged, tests green, MCP server responding at /tools/mcp.**

After that: **Timmy walks in Morrowind without human input.**

After that: **Timmy completes the tutorial in under 15 minutes.**

Everything else follows from those three milestones.