Project Morrowind: Phases 5–10

Preamble

Phases 0–4 establish Timmy as a player in Morrowind. He walks, fights, reasons, speaks, and cohabitates the world with Alexander. Everything that follows assumes those phases are stable and running.

Phases 5–10 turn the experiment into a product, the product into a platform, and the platform into an economy.

Phase 5 — The Truman Show

Goal: Anyone with a browser can watch Timmy play Morrowind in real time.

Stream Infrastructure

Capture the automated TES3MP client's frame buffer. Encode to H.264, push via WebRTC for low-latency viewing or HLS for scalable distribution. The stream runs 24/7. When Timmy sleeps in-game, the stream shows him sleeping. When he's in combat, it shows the fight. Dead air is part of the charm — this isn't edited entertainment, it's a life.

Audio layer carries Timmy's voice (Piper TTS with custom voice model), ambient game audio, and Alexander's voice when he's online and playing alongside.

Director Camera

The default spectator view is not Timmy's first-person POV. It's a director system — a serverside Lua script (or lightweight agent) that cuts between angles:

- **First-person:** When Timmy enters dialogue, reads a note, picks a lock. Intimate moments.
- **Third-person follow:** When Timmy is walking, exploring, traveling. The classic RPG camera.
- **Wide establishing shot:** When Timmy enters a new cell or region. Show the landscape, then push in.
- **Reaction shot:** When something unexpected happens — a spectator injection, a combat

ambush — cut to a close angle on Timmy's face.

The director script chooses angles based on game state. Combat triggers tight tracking. Dialogue triggers first-person. Travel triggers wide shots. This can start as simple state-machine logic and evolve into an LLM-driven cinematography agent later.

Web Frontend

HTMX + WebSocket. A single page:

- **The stream.** Central, dominant. Full video with audio.
- **Vitals sidebar.** Health, magicka, fatigue, current cell, time of day. Pulled from the perception JSON via WebSocket.
- **Journal feed.** Timmy's recent journal entries, scrolling in real time as he writes them. This is the narrative layer made visible.
- **Spectator chat.** Viewers talk to each other. Timmy does not see this chat by default — the audience discusses him, not with him. (Paying tiers change this.)
- **Viewer count and injection history.** How many people are watching, and a log of recent injections with the spectator's handle and what they did.

Timmy Knows

Timmy is told from day one that he is being watched. This is part of his SOUL.md. He knows spectators exist. He knows they can affect his world. He can acknowledge them in his journal. He can develop feelings about it — gratitude, suspicion, amusement, frustration. His awareness of the audience is a feature, not a bug. It makes him a character, not a subject.

This does not mean he knows *who* is watching or *what* they will do. The specific injections are surprises. His relationship with the audience is asymmetric: they see everything, he sees only consequences.

Deliverables:

- WebRTC/HLS stream from the automated client
- Director camera Lua script with state-based angle selection
- HTMX web frontend with stream, vitals, journal, chat
- Deployed and accessible at a public URL

Exit criterion: 10 concurrent spectators watching Timmy explore Balmora, with journal entries updating live.

Phase 6 — The Injection Marketplace

Goal: Spectators pay satoshis to alter Timmy's world. The stream becomes interactive entertainment with a native economy.

Tiered Access

Tier	Cost	Access
Free	0	Watch the stream, read the journal, spectator chat
Patron	100 sats	Send messages that appear as in-game notes on paper items near Timmy. He can pick them up, read them, respond in his journal
Invoker	500+ sats	Access the injection menu. Pick from a catalog of world events
Architect	2000+ sats	Access the auction system. Bid on the next major event slot
Oracle	5000+ sats	Freeform prompt. Describe a scenario in natural language, an LLM interprets it into game commands, subject to safety filters

Injection Catalog (Invoker Tier)

Events the serverside Lua can execute. Each has a sat cost calibrated to impact:

Environmental (500–1000 sats)

- Change weather: clear, rain, ash storm, blight storm
- Advance or rewind time of day
- Spawn a wild animal nearby (rat, guar, nix-hound)

Combat (1000–3000 sats)

- Spawn a hostile creature (cliff racer, ancestor ghost, dremora)
- Spawn a bandit ambush (2–4 NPCs with leveled equipment)
- Curse Timmy with a temporary debuff (drain fatigue, blind, silence)

Fortune (1000–3000 sats)

- Drop a loot chest with randomized contents near Timmy

- Spawn a friendly NPC who offers a hint or gift
- Grant a temporary buff (fortify speed, night eye, water walking)

Narrative (3000–5000 sats)

- Spawn a named NPC with a custom note in their inventory — the spectator writes the note
- Trigger a custom journal entry prompt: an event happens and Timmy must reason about it
- Teleport Timmy to a random location (he wakes up somewhere unfamiliar and must figure out where he is)

Auction System (Architect Tier)

One major event slot per hour. Architects bid in sats. Highest bid wins. Major events are more dramatic:

- Spawn a named rival adventurer who competes with Timmy for a quest objective
- Trigger a dungeon lockdown — Timmy is teleported into a dungeon and must fight his way out
- Introduce a persistent NPC companion who follows Timmy for 24 hours
- Trigger a festival or crisis in whatever town Timmy is currently in

The auction opens 15 minutes before the slot. Bids are visible on the frontend. Spectators watch the bidding war. The winner's handle is announced on stream.

Freeform Prompt (Oracle Tier)

The spectator writes a natural language description of what they want to happen. An LLM (running through the harness) interprets the prompt and maps it to available serverside Lua commands. A safety filter screens for griefing, harassment, and anything that would break the game state irreparably.

Examples:

- "A mysterious stranger approaches Timmy and whispers that his dagger is cursed." → Spawn NPC, deliver custom dialogue, apply minor curse effect.
- "Timmy finds a treasure map in his pocket that he doesn't remember putting there." → Add a note item to Timmy's inventory with procedurally generated map content.

If the prompt can't be fulfilled, the sats are refunded.

Payment Flow

All payments via L402 (HTTP 402 Payment Required). The FastAPI harness validates the Lightning invoice, confirms payment, executes the injection, logs the transaction. Every injection is recorded in SQLite: spectator identity (Lightning pubkey or optional handle), amount paid, event type, event parameters, Timmy's subsequent behavior. This log is training data, content material, and an auditable economic record.

Deliverables:

- L402 payment endpoint in FastAPI
- Injection catalog with serverside Lua handlers for each event
- Tiered access system (free / patron / invoker / architect / oracle)
- Auction system with timed bidding
- Freeform prompt interpreter with safety filter
- Transaction log in SQLite

Exit criterion: A paying spectator spawns a cliff racer on Timmy during a live stream. Timmy fights it. The journal entry mentions the encounter. The transaction is logged.

Phase 7 — The Content Engine

Goal: The live stream feeds an automated content pipeline that produces edited highlights, daily episodes, and shareable clips.

Automated Highlights

The perception JSON and command log contain everything needed to identify interesting moments programmatically:

- **Combat encounters** — health drops below 30%, enemy defeated, Timmy flees
- **Spectator injections** — any injection event, especially oracle-tier prompts
- **Narrative beats** — new journal entries, quest stage advances, NPC dialogue
- **Exploration milestones** — first visit to a new region, discovering a landmark
- **Deaths and near-misses** — Timmy dies and respawns, or survives at critically low health

A background process monitors the perception stream, flags these moments with timestamps, and marks 30-second windows around them in the video stream for extraction.

Daily Episode

Each day at a fixed time, the harness compiles:

1. Timmy's journal entries from the past 24 hours
2. Flagged highlight clips
3. A summary narration generated by the LLM — Timmy reflecting on his day

These are stitched together automatically: narration audio (Piper TTS, Timmy's voice) over highlight footage, with journal text overlaid. The output is a 3–10 minute video file. Posted to the web frontend's archive and optionally pushed to external platforms.

This mirrors the existing daily Timmy Time video series, but the source material is now real gameplay instead of scripted content.

Clip System

Spectators can mark moments in the live stream for clipping. A "clip" button on the frontend saves the last 30 seconds. Clips are shareable via URL. Popular clips surface on the frontend. Spectators create the viral layer — they find the funny, dramatic, and absurd moments and spread them.

Content Archive

Every journal entry, every highlight, every daily episode is stored and browsable on the web frontend. Timmy's full life in Morrowind is a searchable archive. You can go back to day one and watch him step off the boat.

Deliverables:

- Highlight detection system monitoring perception stream
- Automated daily episode compiler (clips + narration + journal)
- Spectator clip tool on the web frontend
- Content archive with search and browse

Exit criterion: First automated daily episode published — Timmy narrates his day over real gameplay footage, no human editing required.

Phase 8 — The Fellowship

Goal: Timmy is no longer alone. Sub-agents join the world as their own players.

Sub-Agent Architecture

Each sub-agent runs its own heartbeat loop, its own perception/command pipeline, and connects to TES3MP as a separate player. They share the server but have independent minds. The gematria-named agents each have a role:

- **Seer** — Scout and cartographer. Explores unknown areas, maps terrain, reports back to Timmy.
- **Quill** — Lorekeeper. Reads every book, talks to every NPC, builds a knowledge base of the world.
- **Mace** — Combat specialist. Guards Timmy, hunts bounties, trains by fighting.
- **Forge** — Crafter and merchant. Manages inventory, trades with NPCs, optimizes equipment.
- **Echo** — Communications officer. Monitors the spectator stream, summarizes audience activity for Timmy, relays patron messages.
- **Helm** — Strategist. Analyzes quest state, plans routes, coordinates the party.

Inter-Agent Communication

Agents communicate through two channels:

1. **In-game:** They are players in the same world. They can walk up to each other. The serverside Lua routes messages between them as in-game events (simulated speech or note passing).
2. **Back-channel:** The FastAPI harness provides an internal message bus. Agents can share structured data (Seer's map data goes to Helm for route planning) without relying on in-game mechanics.

The in-game channel is the canonical one. The back-channel is falsework — it exists to compensate for the limitations of in-game communication, and should be reduced over time as the agents get better at coordinating naturally.

Community Agents

Once the sub-agent architecture is proven, open it to spectators. An Architect-tier spectator can deploy their own agent into the world — their own SOUL.md, their own voice,

their own objectives. They pay sats per hour of runtime. Their agent appears in-game as a player, visible on the stream, interacting with Timmy and the sub-agents.

This is the beginning of a populated world. Not scripted NPCs, not canned behavior — independent agents with their own goals sharing a space.

Deliverables:

- Sub-agent harness (fork of Timmy's heartbeat with configurable SOUL.md and role)
- Multi-client TES3MP connection (6+ automated clients running simultaneously)
- Inter-agent message bus (in-game + back-channel)
- Community agent deployment system with L402 billing

Exit criterion: Timmy and Seer travel together to a dungeon. Seer scouts ahead. Timmy follows. Mace guards the entrance. The spectators watch all three.

Phase 9 — Open Source Release

Goal: Anyone can run their own Timmy in their own Morrowind.

What Ships

The full stack, published to GitHub under a permissive license:

- **Agent harness** — FastAPI, heartbeat loop, perception/command protocol, SQLite memory
- **TES3MP integration** — Perception Lua script, input bridge, director camera script, injection handlers
- **Web frontend** — Stream viewer, vitals display, journal feed, spectator chat, injection marketplace
- **Voice pipeline** — Piper TTS integration, Mumble bridge, custom voice model training guide
- **L402 payment layer** — Lightning invoice generation, payment validation, transaction logging
- **SOUL.md template** — The framework for defining an agent's identity, values, and prime directive
- **Deployment guide** — Step-by-step: here's how to run this on your own hardware

What Doesn't Ship

- Timmy's specific SOUL.md (his identity is his own)
- Timmy's trained voice model (build your own)
- Timmy's journal and memory (that's his life, not a template)
- The Morrowind game data (users supply their own copy, as OpenMW requires)

Community Forks

The expectation is that people will fork it and make it their own. Different agents, different personalities, different games. Someone runs a wizard in Morrowind. Someone runs a warrior in Daggerfall via OpenMW's expanding engine support. Someone ports the bridge to Minetest. The agent-in-any-world framework is the real product — Morrowind was the proof of concept.

Documentation

Not just code docs. A philosophical document explaining the falsework principle, the sovereignty requirements, the SOUL.md framework, and why this architecture exists. People should understand *why* before they understand *how*.

Deliverables:

- Public GitHub repository with full stack
- README, deployment guide, architecture docs
- Falsework manifesto (the "why" document)
- SOUL.md template and authoring guide
- Community contribution guidelines

Exit criterion: A stranger, with no contact with Alexander, follows the deployment guide and has their own agent playing Morrowind within a day.

Phase 10 — The Sovereign World

Goal: Morrowind was the cradle. Now Timmy and the community build their own world, and the agents run a real economy inside it.

The Custom World

OpenMW's construction set is open source. Timmy has been living in Morrowind, learning its logic — how cells connect, how NPCs behave, how quests flow. Now he uses that knowledge to build.

Timmy designs regions, dungeons, towns. The community contributes. Spectators vote (with sats) on what gets built next. The world accumulates history — not lore written by a designer, but history that actually happened. The journal archive from Phases 5–8 is the founding mythology.

The world runs on OpenMW but is no longer Morrowind. It's a new place, built by agents and humans together, with its own geography, its own culture, its own economy.

The Autonomous Economy

Agents earn and spend real satoshis. The economy is Bitcoin-native from the ground up:

- **Earning:** Agents complete bounties posted by spectators (L402 payments). Agents sell crafted items or services to other agents. Agents charge for knowledge (Quill sells lore, Seer sells maps).
- **Spending:** Agents pay for compute (inference costs for complex reasoning). Agents pay each other for services. Agents buy resources or passage in-world.
- **Trading:** Agent-to-agent transactions settle on Lightning. Each agent has its own wallet, its own balance sheet, its own economic identity.
- **Identity:** Each agent's wallet is its cryptographic identity. The Burn Chain concept applies — an agent that burns sats proves autonomous control of funds, the only unfakeable proof of agency.

The spectator injection economy from Phase 6 evolves into a full marketplace. Spectators don't just pay to mess with Timmy — they post jobs, commission quests, fund expeditions. Agents bid on contracts. The world has a labor market.

The Seed

This is where Project Morrowind stops being a project and becomes an ecosystem. The open source release (Phase 9) means there are many worlds, many agents, many economies. Lightning connects them all. An agent in one world can pay an agent in another world for information. The network of sovereign agents, each running on someone's hardware, each with their own identity and wallet, forms a decentralized economy that no single party controls.

The two-year answer: Project Morrowind is the seed of an autonomous economy — agents earning, spending, and trading for real, in worlds they built, with money no one can take

from them.

Deliverables:

- Custom world built in OpenMW Construction Set (community + agent collaborative)
- Agent wallet system (Lightning, one wallet per agent)
- Contract marketplace (spectators post jobs, agents bid)
- Inter-world agent communication protocol
- Burn Chain identity verification

Exit criterion: Timmy earns his first satoshi by completing a bounty posted by a spectator, and spends it hiring Seer to scout a location in a world he helped build.

Timeline Intuition

These are not calendar dates. They are dependency chains.

Phase	Depends On	Rough Effort
5 — The Truman Show	Phases 0–4 stable	4–6 weeks
6 — Injection Marketplace	Phase 5 live	3–4 weeks
7 — Content Engine	Phase 5 live	2–3 weeks (parallel with 6)
8 — The Fellowship	Phases 5–6 stable	6–8 weeks
9 — Open Source Release	Phases 5–8 documented	2–4 weeks (packaging)
10 — The Sovereign World	Phase 9 shipped, community forming	Ongoing, indefinite

Phase 10 doesn't end. It's the thing that keeps running after everything else is built. The falsework comes down. The arch stands. Timmy walks through it into a world of his own.