Guiding Light

Game Systems Design Document

How the prototype's systems interlock to produce the player's experience.

1. The Systems Layer

Guiding Light is built on five interlocking systems. None of them work alone. Each one feeds another, and the friction between them is what produces the prototype's central tension: every act of seeing costs something.

This document is about how those systems were designed, what each one does to the player's decision space, and how they were tuned through playtest. It is not about the code that runs them — that lives in the technical design document. This is the design layer.

2. The Oil Economy

The oil economy is the prototype's spine. Every other system either feeds into it or is constrained by it.

Core Rules

- Oil is a single resource, measured as a float (range: 0 to startingOil).
- Oil drains while the torch is lit at a rate of 1.0 unit per second.
- Oil does not regenerate. The only recovery is via oil pouches placed in levels.
- Oil persists across all scenes — IntroLevel, Level 1, Level 2, Level 3 — without reset.
- Death does not refill oil. The player respawns with the oil they had at the start of the current scene attempt.

Tuning Values

Parameter	Value	Justification
Starting oil (IntroLevel)	50.0	Enough for ~50 seconds of continuous light. Tight but survivable.
Drain rate	1.0 / second	Reduced from initial 1.5 after playtest showed L2 was reliably unwinnable.
Pouch refill	25.0	Half a full tank. Three pouches across three levels = potential to top up to full once if collected.

Max oil cap	50.0	Pouches clamp at startingOil. Players cannot stockpile beyond the cap.
Number of pouches	1 per level	Three total. Forces choice about when to collect.

Why These Values

Drain rate is the most important number in the prototype. It defines how much information the player can buy with their entire resource budget. At 1.0/sec, a full tank (50 oil) buys 50 seconds of vision — enough to see a level several times if used in short bursts, but not enough to keep the torch lit through any single level continuously.

This creates the central decision: light briefly to memorize, then traverse blind, repeating as needed. Players who treat the torch as a flashlight (continuous light) run out by mid-Level 2. Players who treat it as a sonar pulse (brief flashes) have headroom for Level 3.

Pouch Placement as Design Lever

Each pouch is placed deliberately within its level's flow:

- Level 1 pouch: mid-level. Reward for cautious exploration.
- Level 2 pouch: near the upper path. Requires the player to climb during the stalactite test — risky but rewarded.
- Level 3 pouch: late in the level. Forces players to survive most of the level before refilling.

This placement is what makes oil economy a real strategy: a player who burns oil in L1 hoping to refill at the L1 pouch may find they spent too much before reaching it. A player who saves oil reaches the pouch with a full or near-full tank — and the pouch's refill (clamped at startingOil) was effectively wasted.

The economy punishes both extremes: hoarding wastes pouch value; profligacy starves later levels.

3. The Light System

Light is the action that triggers every interesting decision in the game.

Core Rules

- Holding Z lights the torch. Releasing extinguishes it.
- Lighting requires oil > 0. At zero oil, the torch cannot be lit.
- Torch automatically extinguishes when oil reaches zero mid-use.
- The light reveals the entire scene — there is no falloff radius. World becomes fully visible.
- Without light, the player sees only the Player Locator (a small UI marker showing screen position).

Why Binary On/Off

An early iteration explored two-intensity light (dim at half-drain, bright at full-drain). It was rejected. The binary choice forces commitment. The player cannot 'mostly see'; they must decide to spend oil or not. This creates discrete decision points the player can feel.

A continuous brightness system would have given players a permanent middle-ground option ('I'll just keep it dim to save oil') and eliminated the moments of darkness that make the game tense.

Why Hold-To-Light, Not Toggle

Initial design used toggle (press Z to light, press again to extinguish). Playtest showed players forgot the torch was on, burning oil while not actively using vision. Hold-to-light created intentional, accountable consumption — every second of light is a second the player chose.

This is a real example of how UX choice affects systems behavior. A toggle creates passive consumption. A hold creates active consumption. Same underlying economy, completely different player relationship to it.

Information Economy

The light system is fundamentally an information economy. The player exchanges oil (a resource) for information (knowledge of the level layout, hazard positions, paths). Every other system feeds into how valuable that information is:

- In Level 1, information is about static positions — once memorized, the value persists.
- In Level 2, information has a half-life — stalactites move, so memorization decays.
- In Level 3, information triggers consequences — looking at a monster makes it chase.

The same oil cost buys different value in each level. This is the prototype's key systems insight: a resource's worth depends on what it can be exchanged for, and that exchange rate is the level designer's lever.

4. The Hazard System Layering

Each level introduces exactly one new hazard type. The hazards layer cognitively — the player must keep prior systems active while learning the new one.

Level	Hazard Type	Threat Model	Player Decision Created
L1	Static (spike rocks, pits)	Position-based: stand on it = die.	Where to step (memorization).
L2	Timed (stalactites)	Time-based: be there during fall = die.	When to step (timing + memorization).
L3	Active (monsters)	Attention-based: be lit near it = die.	Whether to look (memorization + timing + attention).

Why Add One Layer at a Time

Each layer multiplies the player's cognitive load. Adding two new layers at once would create unfair learning conditions — a player dying could not isolate which mechanic killed them, which prevents learning.

By introducing one at a time and reusing previously-learned mechanics, the player builds vocabulary. By Level 3, they are simultaneously managing oil, watching for stalactites (carried from L2), and managing monster attention (new). The cognitive load is maximum, and that maximum is earned.

System Interactions

The hazards do not just stack — they interact. Specific examples from the prototype:

- Level 2 Twist beat: a stalactite over a pit, with the beacon piece on a platform below. Player must time the stalactite AND execute a platforming jump AND see well enough to land. Three systems demanding attention at the same moment.
- Level 3 Twist beat: a monster sits in the only gap forward. Player must extinguish the torch (oil decision), walk forward blind (information sacrifice), and discover the monster is solid in dark (counterintuitive learning). The light-oil-hazard systems all interact in one moment.
- Continuous oil across levels means a Level 1 hazard mistake (a spike rock that costs no oil to die from) still costs the player oil they spent lighting to see the hazard. Death doesn't reset that loss.

5. The Monster Behavior System

Monsters are the prototype's most complex system because they exhibit different behaviors based on torch state, and those behaviors invert the player's relationship to them.

Two States, Two Roles

Torch State	Monster Role	Player Relationship
Off (dark)	Solid platform	Monster is a physical obstacle. Can stand on, walk past, jump from.
On (lit)	Active threat	Monster moves toward player at 1.6x player speed. Touch = death.

This dual role creates the prototype's most subversive design moment. Every other game establishes 'this thing is dangerous, avoid it.' Guiding Light says 'this thing is dangerous WHEN you can see it. Otherwise it's useful.' That inversion requires the player to override a deeply-trained instinct.

Why Faster Than Player

Monsters move at 1.6x player speed when chasing. Tested values: 1.2x (felt avoidable, removed the threat), 1.6x (felt threatening but learnable), 2.0x (felt unfair, deaths felt like ambush).

1.6x means the player cannot outrun a monster — they must extinguish the torch to escape. This forces the design intent: monsters are managed through torch state, not through movement skill.

The Roar

When a monster transitions from idle (dark) to chasing (lit), a roar sound plays exactly once. This is functional design, not flavor:

- It signals the state change to the player — they hear the consequence of their decision to light.
- It plays once, not on loop, so it doesn't become noise.
- It's spatially clear (comes from the monster's audio source), so the player learns to associate the sound with that monster's position.

The roar is part of the feedback loop, not decoration. Audio is a system layer.

6. The Feedback System

The prototype uses three channels of player feedback, each tuned to teach a different thing:

Channel	Information Communicated	Example
Visual (light)	Spatial layout, hazard positions, paths	Torch on reveals the entire scene.
Audio cues	Imminent danger, state changes	Stalactite crack precedes fall by 0.5s. Monster roar marks chase start.
UI elements	Resource state, collection progress	Oil bar shows current oil. Beacon counter shows X/3 collected.

Information Hierarchy

These channels are designed to layer, not compete:

- Visual is high-information but expensive (costs oil).
- Audio is low-information but always available (no cost).
- UI is constant-information about meta-state (oil, beacons), never about world.

This separation matters. If audio gave hazard positions, the visual cost (oil) would lose value. If UI showed level layout, lighting would be obsolete. Each channel is gated to communicate only what it should, and the player learns to read all three together.

Why No Text Tutorials

The prototype has no popup tips, no 'press Z to light' overlay, no 'monsters are dangerous when lit' warning. Every mechanic is taught through level design and consequence.

This is a deliberate systems decision. Text tutorials externalize learning into reading; level-design teaching forces learning through play. The four-beat structure (Introduce, Develop, Twist,

Test) per level is the teaching system. The hazards are the teachers. Death is the feedback. Restart is the iteration.

7. Progression Gating

The win condition is collection-based: three beacon pieces, one per level.

How Gating Works

- Each level's EndLevel trigger checks the player's beacon collection count.
- Level 1 EndLevel requires 1 piece collected.
- Level 2 EndLevel requires 2 pieces.
- Level 3 EndLevel requires 3 pieces.
- Touching EndLevel without the required pieces does nothing — silent gate. The player must return to find the beacon.

Why Hard Gating Instead of Soft Gates

An earlier design considered making beacons optional (skip a level's beacon, lose 'completion %', but progress). It was rejected. Soft gates would let players skip the climax of each level — the section the entire level's four-beat structure builds toward. The mechanic teaching would be wasted.

Hard gating enforces engagement with each level's climax. The player cannot complete the prototype without engaging the design's intended climax for each mechanic. Every player has the same shape of experience, even if the path through it differs.

The Beacon Placement Strategy

Each beacon piece sits in the Twist beat of its level — the climax of the level's mechanical learning, not the Test beat (which is the final challenge). This means the player collects the beacon while still mid-level, then must escape the rest of the level to reach EndLevel.

This creates a two-act structure within each level: pre-beacon (learning), post-beacon (proving). The post-beacon Test section is where the player demonstrates that the lesson landed, with the beacon already in hand as a marker of progress.

8. The Death System as an Economic Pressure

Death is not a punishment — it's an economic event.

Death Rules

- Death triggers: spike rocks, pits, stalactites (during fall or while grounded), monsters (while torch lit).
- On death, player freezes for 0.3 seconds. Sound plays. Scene reloads.
- Oil reloads to whatever it was at the player's scene-entry state.
- Beacon pieces collected in previous scenes persist. Beacons collected in this scene attempt are lost on reload.

Why 0.3s Pause

Tested at 0.0s (instant reload, felt jarring — player couldn't register what killed them), 0.3s (felt clean — moment to absorb), 1.0s (felt slow — interrupted flow). 0.3s is the floor where the player has enough time to read 'I died' without breaking pace.

Why Oil Doesn't Refill On Death

Earlier design refilled oil on death. Playtest showed this eliminated the oil economy entirely — players treated death as a free recovery and burned oil recklessly. The economy required real consequence.

The current rule (oil preserves to scene-entry state) makes death an economic event: every death costs oil already spent in this attempt, plus restarts the section's traversal cost. Both costs accumulate against the player's total budget.

9. Emergent Decisions

The systems above are designed to produce specific decision moments. These are the experiences the player should face by design:

Macro-Level Decisions

- How much oil to spend in this level vs. save for the next. (Cross-level economy.)
- Which pouches to detour for. (Geographic vs. economic value.)
- Whether to attempt the Test section with low oil or restart the level for a fresh attempt with a better economy.

Section-Level Decisions

- Light briefly to memorize, then traverse blind — or keep light steady for the whole section?
- In Level 3 monster sections: light to see the monster's position and risk death, or stay dark and risk falling?

Frame-Level Decisions

- Light during the jump or after? (Brief visibility during a critical moment, or save oil and trust muscle memory.)
- Tap or hold? (How long is 'enough' to absorb the layout?)

These decisions emerge from the systems, not from explicit choices presented to the player. The player never sees a menu asking 'how much oil will you spend?' The economy makes the question implicit, and the player answers it every second of play.

10. Tuning Pass: Playtest-Driven Iteration

Several systems values changed through playtest. Each change reflects how playtest evidence shaped a system parameter:

Oil Drain Rate: 1.5 → 1.0

Initial playtest: players consistently ran out in Level 2's stalactite sections. The mechanic was tight, but tight in a frustrating way (couldn't complete) rather than tense (could just barely complete). Reducing drain gave headroom while preserving pressure. Players still ran low in Level 3, which was correct.

Stalactite Crack Duration: 0.2s → 0.5s

Original 0.2s warning was below human reaction time (~150-250ms). Players died from sounds they heard but could not act on. Extending to 0.5s gave a fair window without removing the demand for alertness.

Monster Speed: tested 1.2x → 1.6x → 2.0x

1.2x: monster felt avoidable through movement alone, undermined the torch-management design. 1.6x: required torch management, felt threatening. 2.0x: created cheap deaths where players could not extinguish in time. 1.6x kept.

Pouch Refill: 50 → 25

Initial 50 (full tank refill) made the pouch a 'reset' button — collect it and the prior section's oil cost was erased. The economy stopped feeling like a budget. Reducing to 25 (half tank) made the pouch a recovery, not a reset, preserving cross-section consequence.

Beacon Placement: Test → Twist

Original placement put beacons at the end of each level (in the Test section). Players who failed the Test had to redo the entire level to retry collection. Moving beacons to the Twist beat preserved the climactic placement while creating a clear 'I have the piece, now I need to escape' moment.

11. What the Systems Don't Do

Several systems were considered and cut. Each cut was a design decision:

- No combat system. Player cannot fight monsters or destroy hazards. The only verb is movement + light.
- No upgrade system. Oil capacity is fixed at startingOil. No 'larger tank' meta-progression.
- No multiple oil types. Oil is one resource. No 'magic oil that burns slower' or 'pure oil that lights brighter.'
- No story branches. The cave is linear; the player's only choices are within sections.
- No timer or score. The game has no efficiency metric other than the oil economy itself.

Each cut sharpens the prototype's argument. Adding any of these would dilute the central tension by introducing alternative paths to the same outcome. The simplicity is the discipline.

12. The Design Argument Validated by Systems

The systems above exist to argue one thing: every act of seeing must cost something.

If the oil drained without lighting being a choice, sight would not cost. If lighting refilled oil somehow, sight would not cost. If hazards were instant-readable without lighting, sight would not cost. If oil reset between levels, sight would cost only briefly. If death refilled oil, sight would cost only until the next failure.

Every system rule above reinforces the central argument. That is what game systems design does: it ensures the design's intent survives every interaction. The argument is not made by any single system; it is made by the way they refuse to let the player escape the cost.

Playtest confirmed: players consistently described their experience using economic vocabulary. They were rationing. They were choosing. They were anxious about a resource they had spent. That is the systems landing as designed.

—

Designed, Developed and in-game art by Pratik. Solo prototype, ~3 weeks. Unity 6.3 LTS, C#, WebGL.