Night Shift AI Studio — Optimized Full Talk Script (English)

Grey

Scene0 — Title / Claim (0:00–0:20)

Good morning everyone. My project is **Night Shift**.
The key idea is simple: **Difficulty equals computation budget**.
If you give the engine more budget, it sees further, becomes more stable, and avoids traps.

(One-line optional pun, only if it lands:)
I call it *Night Shift* because it's built with *knights* — and I built it mostly at night.

Scene1 — Freeze: Same board, different choice (0:20–1:20)

Let's start with a quick question.

Here is a position: **same board, same side to move**.
The opponent's queen is **hanging**.
Would you take the queen?

(*Pause 2 seconds.*)

Now watch what two levels do with different budgets.
On the **left**, the low-budget level grabs it immediately — it *looks* great.
But a few moves later, it walks into a tactical trap and gets punished.
On the **right**, the higher-budget level plays a different move.
It sees the trap first, and it chooses the safer line.

This difference is exactly what I mean by **controllable difficulty**:
same position, different behavior — because the budget is different.

Scene2 — Difficulty Dial: four levels as budget presets (1:20–2:05)

So how did I implement "difficulty"?

I built **four levels** by increasing computation budget and heuristic precision step by step. These levels are not just names — they are **budget presets**.

- **Level 1: Greedy (1-ply).** Fast, but short-sighted. It chooses the best move after a one-step lookahead.

- **Level 2: Minimax + Alpha-Beta, depth 3.** Uses a simplified *material-only* evaluation, and adds a small randomness tie-break among the top moves.
- **Level 3: Practical engine.** Alpha-beta plus **quiescence search** and a **transposition table** to cache explored positions. It is time-limited at **0.6 seconds per move**.
- **Ultimate.** Same structure as Level 3, but with a stronger, well-established evaluation function (**PeSTO**) and a larger budget: **1.2 seconds per move**.

Scene3 — Knobs: why budget changes behavior (2:05–3:05)

To make this explainable, I summarize difficulty into four knobs:

1. **Horizon**: how far the engine can see (depth / iterative deepening).
2. **Efficiency**: how many useless branches it can avoid (alpha-beta, move ordering, transposition table).
3. **Evaluation richness**: how well it can score a position (material-only $\rightarrow$ positional $\rightarrow$ PeSTO).
4. **Randomness**: controlled noise so low levels feel less "perfect" and more human.

And the key point is: **levels are just presets of these knobs**.
So difficulty becomes **controllable**, **repeatable**, and **explainable**.

Scene4 — Ladder: the behavior staircase (3:05–3:45)

You can think of the four levels as a ladder:

- L1 is fast and impulsive.
- L2 can see simple tactics, but misses deeper threats.
- L3 is a practical engine: deeper search, calmer decisions.
- Ultimate is similar to L3, but stronger evaluation and more time — so it converts advantages more reliably.

So the ladder is not just "stronger".
It is **more stable**, and it gets trapped less often.

Scene5 — Search X-Ray: internal mechanism (3:45–5:05)

Now, what changes *inside* when budget increases?

First, the engine follows a **principal variation** — the current best line it believes in.
Then, with **alpha-beta pruning**, many branches become irrelevant and are cut away.
With **iterative deepening**, it searches depth 1, then 2, then 3, and so on, always keeping the best move found so far.
And the key upgrade is **quiescence search**: it extends noisy tactical positions, so it reduces the horizon effect.

So, budget is not just "waiting longer".
Budget changes the **shape of the search tree**, which changes decisions.

Scene6 — Evaluation Harness: fair, reproducible, scalable (5:05–6:05)

To prove the levels really separate, I built an evaluation pipeline.

Everything is standardized:

- **Same rules**, same scoring: win $= 1$, draw $= 0.5$, loss $= 0$.
- **Colors reversed** to remove first-move bias.
- A max of **500 plies** to avoid endless games.

And I used three protocols:

- **M2M standard**: 100 games per pairing (50/50 colors).
- **H2M**: 10 participants, 5 games per level, with rating and feedback.
- **Time-scaled**: from 0.1s/move, step by +0.2s, and test each budget with enough games.

This makes the results comparable and reproducible.

Scene7 — Evidence Wall + TSB strip (6:05–8:20)

Now the evidence.

On the left is the round-robin scoreboard. The key pattern is clear:
levels separate in strength. For example, **L2 vs L1 reaches 0.85 score rate**, and **L3 vs L2 reaches 0.75**.
So the ladder is real.

On the right is the cost–strength curve. This is the second takeaway:
diminishing returns. Beyond a certain budget, strength keeps improving, but each extra millisecond gives less gain.

Finally, at the bottom, the **TSB film-strip** shows the same trend stays stable across different time budgets.
So the conclusion is robust: **budget controls difficulty, consistently**.

Scene8 — Future Work (8:20–9:00)

Future work has three clear directions:

- **Performance optimization**: faster search and better throughput.
- **Dynamic difficulty adjustment**: adapt the budget and evaluation aggressiveness based on recent outcomes.
- **Generalization**: apply the same multi-level framework to other board games by swapping move generation and evaluation.

Scene9 — Closing / Handoff (9:00–9:30)

To close, my main claim is: **Difficulty equals computation budget**.

I delivered three things:

1. A multi-level chess AI ladder (L1–L3 + Ultimate).
2. A reproducible evaluation harness (M2M / H2M / time-scaled).
3. Evidence of separation and diminishing returns.

Thank you — and now I can open the live demo. Any questions?