BLUNDERMIND

Bot Controls — Technical Project Brief

Move Selection Mechanics, Formulas & Architecture

All controls described are available in the bot-control-panel.html sidebar. Formulas reference src/50-bot-engine.js and src/60-bot-ui.js. This brief reflects the state of the dev branch as of June 2026.

1. Architecture Overview — Move Selection Pipeline

Every bot move passes through six sequential stages. Each stage can veto, modify, or short-circuit to skip later stages. The stages run in strict order so that think time — computed in Stage 2 — feeds into every downstream degradation calculation.

Stage	Name	What happens
1	Opening Book	ECO table lookup (preferred) or Lichess Masters API (mainline). If a matching move is found, it is played immediately after a brief book
2	Think Time	Actual think time for this move computed from timing mode + Hustle attractor + clock pressure. Result fed as thinkSec into all downstrea
3	Engine Query	Rough thinkSec estimate used to pick Maia ELO (curve A). Maia3 or LCSF queried for candidate move probabilities.
4	Distribution Cut	Curve B sets upper percentile cutoff of candidate list. Luck attractor shifts the sampling window up or down.
5	Attractor Reweighting	Each candidate move's probability is multiplied by exp(logBoost) where logBoost is the sum of all active strategic attractor signals.
6	Temperature Sampling	Final temperature applied (base T × phase/pressure modifiers). One move sampled from reweighted distribution.

Key architectural principle: Think time is computed first (Stage 2), and the resulting thinkSec value drives Stages 3, 4, and 5 via the degradation curves. This means a Coffeehouse Hustler thinking 0.3 s on a move sees heavy ELO and distribution degradation on that specific move — not an average derived from the remaining clock.

2. Move Source Layer

2a. Opening Book

Two independent book modes are available. Either can be active simultaneously; each deactivates permanently the first time it fails to find a candidate move.

Mode	Source	Behaviour
Preferred	In-memory ECO table (~3 000 named openings)	Matches game history against ECO lines whose ECO code or family prefix is in the bot's slot list. Plays the ne
Mainline	Lichess Masters API (live)	Queries masters database for the most popular continuation from the current position. Falls back to Lichess d

Book delay: 400–1200 ms random pause to simulate human reading time.

2b. Maia3 Neural Network

Maia3 is a series of neural networks (600–2600 ELO) trained to predict human moves at each rating level. The bot queries the appropriate ELO model (determined by degradation curve A) and receives a probability distribution over all legal moves.

- Model selection: effective ELO = pressureEffectiveMaiaEloByThink(thinkSec)
- ELO clamped to [600, 2600].
- Probability distribution filtered by quality range (§8), then reweighted by attractors (§6).

2c. Stockfish (SF) Engine

Used when the bot engine mode is set to Stockfish. Skill level 1–20 controlled by sfEffectiveLevel(). The engine is queried via WebWorker; MultiPV probes provide complexity scores used by the Chaos attractor.

```
sfEffectiveLevel: floor = max(blunderFloor, pressureFloor) blunderFloor = max(1, round(startLevel × (1 − blunderLimitCp / 400))) pressureFloor = max(1, startLevel − round(timePressureMaxDrop / 50))
```

Time degradation path (highest priority wins):

- Weaponizer active AND bot is ahead → return floor immediately
- Curve A present → spline interpolation in log-time space → lerp level to floor
- Fallback: linear ramp from startLevel at 30 s → floor at 0 s

2d. Level-Controlled Stockfish (LCSF)

LCSF uses Stockfish with controlled skill variation rather than Maia probability distributions. sfPickLevel() distributes calls around the target level:

```
p(offset) distribution: −2: var2/2 (e.g. 10% at ±2 → 5% chance) −1: var1/2 (e.g. 30% at ±1 → 15% chance) 0: 1 − var1 − var2 (remaining probability) +1: var1/2 +2: var2/2 Level clamped to [1, 20]
```

var1 (±1 spread): 0–50%. **var2** (±2 spread): 0–20%. Setting both to 0 uses the exact target level every move.

3. Think Time Calculation

Think time is calculated in botThinkTime(moveProbs, clockMs) before the engine is queried. The result (thinkSec) drives all downstream degradation curves, so attractors that shorten think time (Hustler) directly increase move-quality degradation for that specific move.

3a. Timing Modes

Mode	Formula / Behaviour
Instant	Returns 0 ms. No delay whatsoever.
Fixed	thinkMs = botFixedDelayMs. Capped at min(fixed, clockMs × 0.08) when clock < 30 s. Capped at clockMs – 3 s when flagging is disabled.
Pace (default)	baseSec = (5 × 60) / pace. complexity = min(1 + entropy × 0.35, 2.5). thinkMs = baseSec × complexity × 1000. Pace slider: 5–120 (moves per 5
Complexity	cplx = sfCplxScore if available, else min(1, entropy / 4). mult = botCplxMin + cplx × (botCplxMax – botCplxMin). thinkMs = botCplxBase × mult ×
Mirror	avg = rolling average of human move timestamps. thinkMs = avg × jitter(0.8–1.2) × (1 + mirrorOffsetPct/100). Falls through to Complexity/Pace v

3b. Think Time Modifiers (applied in order)

Modifier	Condition	Effect
Weaponizer shortcut	Enabled AND bot clock > opp clock + lead ms	thinkMs = 0 ms immediately (skip all delays)
Move blink	botBehavBlink AND entropy < 0.5 (forced/obvious position)	thinkMs = 200–500 ms random; skips further calculation
Clock-pressure mirroring	botBehavClockMirror AND oppClock < botClockMs	thinkMs ×= 0.5
Hustle attractor	value v ∈ [–5, +5]	thinkMs ×= (1 – v × 0.15). At +5: ×0.25 (very fast). At –5: ×1.75 (very slow).
Reconsideration	botBehavReconsider, 15% random chance	thinkMs ×= 1.5 to 2.5 (hesitation pause)
Base jitter	Always applied	thinkMs ×= uniform(0.8, 1.2)
Clock cap (low clock)	clockMs < 30 000 ms	thinkMs = min(thinkMs, clockMs × 0.08)
Flagging guard	botCanFlag = false	thinkMs = min(thinkMs, max(200, clockMs – 3000))
Global cap	Always applied	thinkMs = max(200, min(botThinkCapMs(), thinkMs)). Cap = max(6 s, min(45 s, startClock × 0.02)) –

Position entropy is Shannon entropy over Maia move probabilities: H = –Σ p·log■(p). Higher entropy = more complex / branching position.

4. Time Pressure Play Degradation

Two independent user-editable spline curves control how the bot's play quality degrades as think time per move decreases. Both curves use log-scale interpolation on the X axis (think time in seconds) so that the visually equal spacing on the panel matches the mathematical interpolation.

```
Curve interpolation (log-x space): pts sorted by x. For xSec in [pts[i].x, pts[i+1].x]: t = (ln(xSec) − ln(pts[i].x)) / (ln(pts[i+1].x) − ln(pts[i].x)) y = pts[i].y + t × (pts[i+1].y − pts[i].y)
```

Curve A — ELO Degradation

Maps think time (s) → effective Maia ELO. At full think time the bot queries its configured ELO tier. As think time shrinks the ELO drops, producing weaker move distributions.

```
effectiveELO = pressureEffectiveMaiaEloByThink(thinkSec) = evalPressureCurve(curveA, thinkSec) clamped to [600, 2600]
```

- X axis: think time per move in seconds (log scale, typically 0.1 s – 30 s)
- Y axis: Maia ELO (600 – 2600)
- Default: flat at configured ELO (no degradation) unless curve is set

The rough thinkSec estimate (computed before the Maia query) is used for ELO selection. The precise thinkSec (after the engine responds) is used for curve B and temperature.

Curve B — Distribution Cutoff

Maps think time (s) → upper percentile cutoff of the move candidate list. 100% = full distribution (top candidate down to last). As think time falls the cutoff drops — forcing the bot to pick from a narrower top slice of the distribution (higher-probability, safer-looking moves).

```
effectiveUpperPct = pressureEffectiveDayUpperByThink(thinkSec) = min(botDayUpper, evalPressureCurve(curveB, thinkSec))
```

- X axis: think time per move in seconds (log scale)
- Y axis: upper percentile 0–100%
- Combined with the Move Quality Range lower bound and Luck shift (§8)

Time Pressure Temperature Boost

In addition to ELO and distribution effects, low think time raises the sampling temperature, making move selection more random under pressure:

```
timePressureTempByThink(baseTemp, thinkSec): boost = { steady: 0.0, normal: 1.0, panicky: 2.5 }[botTimePressure] if curveB present: fraction = 1 − evalPressureCurve(curveB, thinkSec) / 100 else: fraction = max(0, 1 − thinkSec / 30) # linear fallback effectiveTemp = baseTemp + fraction × boost
```

botTimePressure modes: "steady" (no boost), "normal" (+1.0 max), "panicky" (+2.5 max).

5. Temperature & Move Sampling

After attractor reweighting, one move is sampled from the distribution using temperature-scaled probabilities. Temperature T controls how peaked or flat the distribution is before sampling.

```
sampleFromProbs(moveProbs, T): scaled[m] = prob[m] ^ (1/T) total = Σ scaled[m] sample r ~ Uniform(0, total) return first m where cumulative sum ≥ r
```

At T=1.0 the original Maia probabilities are used directly. T < 1 sharpens the distribution (top move more likely). T > 1 flattens it (unlikely moves become relatively more likely). T is clamped to ≥ 0.1 to prevent division issues.

5a. Base Temperature Presets

Preset	T value	Behaviour
Focused	0.4	Strongly favours top 1–2 Maia moves
Neutral	1.0	Samples Maia's full distribution as-is
Varied	1.8	Wider sample — more surprise choices
Wild	3.0	Even low-probability moves get meaningful weight

5b. Hustler Phase Temperature

When the Coffeehouse Hustler personality is active (personalityId = "hustler"), the base temperature is replaced by a phase-sensitive curve:

```
hustlerPhaseTemp(): piecesLeft = count of non-pawn, non-king pieces on board fraction = clamp(1 − (piecesLeft − 4) / 10, 0, 1) # 0 at 14 pieces (opening), 1 at ≤4 pieces (endgame) T = 5.0 + fraction × (0.6 − 5.0) = 5.0 in opening → 0.6 in endgame
```

14 non-pawn/non-king pieces = full material. At 4 or fewer such pieces, the endgame temperature floor of 0.6 is reached.

5c. Complexity-Adjusted Temperature

For Maia3 mode, the base temperature can optionally be scaled by position complexity (MultiPV evaluation spread):

```
complexityAdjustedTemp(baseT): if sfCplxScore is null → return baseT cplxFactor = 0.8 + sfCplxScore × 0.4 # 0.8 at simple, 1.2 at complex return baseT × cplxFactor
```

sfCplxScore is the normalized std. deviation of the top-N Stockfish evaluations, ranging 0 (all moves equal) to 1 (widely spread evaluations).

6. Strategic Attractors (Style Gauge)

Strategic attractors reweight each candidate move's probability by multiplying it by exp(logBoost). The total logBoost for a move is the sum of all active attractor contributions. Attractors are zero at centre and scale linearly with slider value (–5 to +5).

6a. CP Budget and Scale Factor

The CP (centipawn) budget (0–300) is a single master gain knob for all attractors. It controls how strongly attractors push the distribution, regardless of the individual slider positions.

```
totalAbs = Σ |v| for all attractor and piece sliders scale = cpBudget / (totalAbs × 150) if cpBudget > 0 and totalAbs > 0 = 0 otherwise For each move: prob_new = prob × exp(logBoost) where logBoost = Σ (attractor_v × scale × signal) At cpBudget=150 with one slider at 1 and all others at 0: scale = 1/150, signal = 1 → logBoost = 1 → exp(1) ≈ 2.7× boost
```

Allocating 150 cp of budget on a single maximally-set attractor produces roughly a 2.7× probability boost on moves that fully satisfy that attractor.

6b. Attractor Formulas

Attractor	Left (–) / Right (+)	Signal formula	Notes
Chaos Agent / Simplifier	– Simplifier → low-σ moves + Chaos Agent → high-σ moves	logBoost += chaosVal × scale × signal (signal from MultiPV σ)	Requires MultiPV Stockfish probe. σ = std. dev. of top-N eval scores.
Complicate / Simplify winning	– Simplify when winning + Complicate when winning	Active when eval ≥ +100 cp. Boosts moves that increase or decrease MultiPV variance.	Only fires in clearly winning positions. Combined with Chaos Agent for fu
Space Cadet / Space Waster	– Space Waster → cede space + Space Cadet → reduce weak sq	delta = currentWeakSq – simWeakSq logBoost += v × scale × tanh(delta/5)	Full board attack map computed on simulated position. Captures discove
Fort Knox / Glass Cannon	– Glass Cannon → exposed pieces + Fort Knox → more defenders	delta = simTotalDefs – currentDefs logBoost += v × scale × tanh(delta/3)	Counts sum of defensive coverage across all bot pieces. Full board attac
Gambito / Gambit Shy	– Gambit Shy → avoid sacrifices + Gambito → follow gambit lines	Opening only (< 20 half-moves). ECO: logBoost += v × scale if move is ECO gambit continuation. Fallback: pawn to attacked, undefended sq.	ECO gambit line match takes priority. Structural fallback if ECO data not
Trade Seeker / Trade Avoider	– Trade Avoider → no captures + Trade Seeker → captures	Capture: logBoost += v × scale Threat: logBoost += v × scale × tanh(newThreats / 2)	Non-captures scored by number of new opponent-piece threats created
Rigid / Loose Pawn Structure	– Loose → open mobile structure + Rigid → tighter structure	Own pawn moves only. delta = currentPenalty – simPenalty logBoost += v × scale × tanh(delta)	Penalty = pawn islands + doubled pawns + isolated pawns. delta > 0 = m
Coffeehouse Hustler / Overthinker	– Overthinker → longer think time + Hustler → shorter think time	Applied to think time (not prob): thinkMs ×= (1 – v × 0.15)	At +5: ×0.25 (4× faster). At –5: ×1.75 (75% slower). Feeds into pressure
Luck / Bad Day	– Good day → top of distribution + Bad day → bottom of distribution	Shifts quality range window: lo = botDayLower – v × 4 hi = _pressureUpper – v × 4	See §8 Move Quality Range for full description.
Panicky / Calm under pressure	– Calm → no temperature boost + Panicky → larger T boost under pressure	Modifies timePressure boost mode. (steady / normal / panicky)	See §4 Pressure Temperature Boost.

Attacker / Peacemaker	– Peacemaker → quiet position + Attacker → maximize threats	totalThreats = Σ attacks on each opp. piece logBoost += v × scale × tanh(threats/6)	Full board attack map on simulated position. Counts attack coverage acr

All attractor signals that use tanh map their input to a smooth –1..+1 range, preventing any single move from receiving an unbounded boost.

7. Piece-Type Attractors

Six independent sliders (–5 to +5) bias the bot toward or away from moving each piece type. The formula is the same for all six:

```
logBoost += pieceVals[pieceType] × scale
```

The piece type is determined from the moving piece before the move. The signal is 1 (or 0 for no boost) — there is no continuous signal function as with most strategic attractors. Scale is shared with the strategic attractors so the CP budget controls piece attractor strength alongside everything else.

Piece	Left (–) behaviour	Right (+) behaviour	Best combined with
Pawn ■	Suppress pawn moves	Prioritise pawn advances	Rigid structure (connected chains)
Knight ■	De-prioritise knight hops	Knight manoeuvres first	Rigid structure (closed positions)
Bishop ■	Keep bishops passive	Diagonal play, bishop pair	Loose structure (open diagonals)
Rook ■	Passive rooks, avoid early trades	File seizure, rook lifts	Open files, trade-seeker
Queen ■	Conservative queen, avoid exposure	Active queen, queen-led attacks	Chaos Agent (keeps complications)
King ■	King safety, stay castled	King activity, endgame aggression	Endgame positions, low piece count

8. Move Quality Range & Luck

The Move Quality Range dual slider selects which segment of the Maia probability-ranked candidate list the bot samples from. Candidates outside the [lower, upper] percentile window are excluded before temperature sampling and attractor reweighting.

8a. Percentile Band Filtering

```
sorted candidates by probability (highest first) cumulative probability sum = total band.lo = (lo/100) × total band.hi = (hi/100) × total Include move m if: cumulative_end(m) > band.lo AND cumulative_start(m) < band.hi
```

Default: lo = 0%, hi = 100% (full distribution). Setting lo = 20%, hi = 60% samples only the middle tier — avoiding the top moves (too accurate) and the worst moves (random blunders).

8b. Luck Attractor Shift

The Luck attractor shifts both bounds of the quality window simultaneously:

```
lo_effective = botDayLower − luckVal × 4 hi_effective = pressureUpper − luckVal × 4 luckVal > 0 (Bad day): window shifts down → samples lower-ranked moves → worse play luckVal < 0 (Good day): window shifts up → samples top-ranked moves → sharper play
```

pressureUpper is the curve-B degraded upper bound, so luck shift compounds with time pressure effects.

8c. Blunder Limit and Min-Probability Filter

Two additional filters prune the candidate list before the quality range is applied:

```
relFloor = bestProb × exp(−blunderLimitCp / 100) absFloor = minProbPct / 100 threshold = max(absFloor, relFloor) keep only moves with prob ≥ threshold
```

- blunderLimitCp (0–400 cp): sets how far below the best move a candidate can fall. 50 cp → tight; 400 cp → everything allowed.
- minProbPct (0–10%): absolute probability floor; removes near-zero candidates.

9. Preset Personalities

Each preset personality is a named collection of attractor values. Loading a personality sets all attractor sliders to the preset values. The values use the same –5 to +5 scale as manual sliders. The table below shows each personality's non-zero attractor settings.

Personality	Tag	Notable attractor values	Character
Captain Entropy ■	Chaos	chaos:+4, compwin:+4, gambito:+3 trade:+2, spacecadet:+2 hustle:–3, structure:–3	Seeks complications, gambit lines, and tactical chaos. Thinks longer per move (h
Norm ■	Order	structure:+4, fortkx:+3, spacecadet:0 chaos:–4, compwin:–4 gambito:–2, hustle:+2	Closes positions, defends solidly. Simplifies when winning. Slightly faster pace.
Attacky McTackerson ■	Captures	trade:+4, attacker:+3, chaos:+2 hustle:–2, structure:–1	Favours captures and threats. Slightly deliberate (hustle –2).
Overthinker ■	Methodical	structure:+3, spacecadet:+3, fortkx:+2 hustle:–4, pressure:–3 chaos:–3, compwin:–2	Maximises board control and structure. Thinks long on every move (hustle –4). Li
Coffeehouse Hustler ■	Fast	hustle:+5, pressure:+2, luck:+2 gambito:+2, hustle:+5 structure:–2	Plays fast and loose. Phase temperature: T=5 opening → T=0.6 endgame. At hu
The Blunderer ■	Human	luck:+3, pressure:+5	Plays normally but cracks unpredictably under pressure. Luck shifts toward lower
The Hoarder ■	Material	gambito:–3, trade:–3, fortkx:+2 chaos:–2, hustle:+2, luck:–2	Never sacrifices material. Avoids trades and gambits. Slight faster pace.
Pawn Chain Gang ■	Pawns	structure:+5, fortkx:+1, hustle:+1 chaos:–1, compwin:–2, gambito:–1	Maximises pawn structure. Slightly faster pace.
Spite Check †	Checks	chaos:+3, compwin:+2, attacker:+3 trade:+2, hustle:–2	Always prefers checking moves. Builds up attacks. Slower deliberate pace (hustl
Clock Watcher ■	Clock	compwin:–2, trade:–1, pressure:–3	Conservative when ahead on time, reckless when behind. Calm under pressure (
Grandmaster Bad Day ■	Human	luck:+5, pressure:+2	Maia 2400 ELO but samples deep in the distribution — strong player having an o

Hustle attractor sign convention: +5 = Coffeehouse Hustler (fast, thinkMs ×0.25). –5 = Overthinker (slow, thinkMs ×1.75).

10. Time Controls

Time controls are defined in TIME_CONTROLS and selected from the bot panel grid. The grid is divided into Blitz (≤5 min), Rapid (10–30 min), Classical/Correspondence (60 min+), Infinite (untimed), and a special 90+30 column for FIDE/Candidates format.

Format	Base time	Increment	Bonus	Bonus at	Inc from
Bullet 1+0	1 min	0 s	—	—	—
Blitz 3+2	3 min	2 s	—	—	—
Blitz 5+0	5 min	0 s	—	—	—
Rapid 10+0	10 min	0 s	—	—	—
Rapid 15+10	15 min	10 s	—	—	—
Classical 30+0	30 min	0 s	—	—	—
Classical 60+0	60 min	0 s	—	—	—
Classical 90+0	90 min	0 s	—	—	—
FIDE 90+30 *	90 min	30 s	+30 min	Move 40	Move 41
Untimed	∞	0 s	—	—	—

** 90+30 (FIDE / Candidates): 90 minutes for first 40 moves, then +30 minutes added to both clocks at move 40. 30-second increment applies from move 41 onward. The clockBonusApplied flag prevents the bonus from being awarded twice.*

Bonus Time Implementation

```
clockAfterMove(): fullMoveNum = ceil(gameMovesAlgebraic.length / 2) if tc.bonusSecs AND fullMoveNum ≥ tc.bonusAtMove AND NOT clockBonusApplied: clockTimeW += bonusSecs (capped at 59940 s) clockTimeB += bonusSecs clockBonusApplied = true if clockInc > 0 AND fullMoveNum ≥ tc.incFromMove: add increment to side that just moved
```

11. Human Behaviour Flags

Human behaviour flags add realistic timing irregularities and clock-pressure responses. Each flag is independently toggleable. They modify think time or move selection in specific situations.

Flag	When active	Effect
Move Blink (botBehavBlink)	Maia3 mode only. Position entropy < 0.5 (forced or obvious move)	Returns 200–500 ms random pause immediately, skipping full think time calculation. Simulates a human instant
Reconsider (botBehavReconsider)	15% random chance per move	Multiplies computed think time by 1.5–2.5×. Simulates the human hesitation of starting to play a move then reco
Clock Mirror (botBehavClockMirror)	Opponent clock < bot clock × 0.6	Halves think time when the opponent is significantly lower on time. Simulates a human speeding up to maintain
Clock Weaponizer (botWeaponizerEnabled)	Bot clock ahead of opp by more than leadMs	Returns 0 ms think time AND drops Stockfish to floor level. Maximises time pressure on the opponent by playing
Can Flag (botCanFlag)	Always on or off	When OFF: think time capped at max(200 ms, clockMs – 3000 ms). Prevents the bot from flagging itself. When

End of brief. For implementation details see src/50-bot-engine.js (engine), src/60-bot-ui.js (config application), and bot-control-panel.html (UI and presets).

Generated June 2026 — Blundermind Bot Controls v1.0